使用手冊

EtherCAT Library API

86Duino Coding IDE 501

EtherCAT 函式庫

(版本 3.2)

修訂

日期	版本	描述
2023/05/05	Version1.0	Initial release.
2023/10/24	Version2.0	Updated API.
2024/08/01	Version2.2	Updated API.
2025/03/20	Version3.0	• Updated API information for 86Duino IDE 501. • Change master to MDevice. • Change Slave to SubDevice. • Add errGetCableBrokenLocation1() figure. • Adjust The Ecoder Reading Object of The Three-Axis Stepper Motor Controller SubDevice (Affects encoder.read): 1. 60e4.0 ==> 502a.1 2. 68e4.0 ==> 502a.2 3. 70e4.0 ==> 502a.3 • Added functions for 3-axis stepper SubDevice.
2025/04/18	Version3.1	• Corrected typos in multiple API comments (Pages 34, 89, 162, 176, 182, 248–280) • Fixed duplication and grammar in function descriptions. • Updated UART-related notes for CIO Series. • Confirmed isSupportDC() applies to SubDevice. • Fixed incorrect type: int writeSII16 to int writeSII32. • Renamed CIO series model from QEC-R11DFFG to QEC-R11CFFG
2025/04/29	Version3.2	• Merged duplicate descriptions of common API functions (e.g., attach(), detach()) into shared references.

版權

版本 3.2，2025 年 4 月

商標聲明

如需更多詳細資訊，或對 ICOP 其他產品有興趣，請造訪我們的官方網站：

- 全球網站：www.icop.com.tw
- 美國網站：www.icoptech.com
- 日本網站：www.icop.co.jp
- 歐洲網站：www.icoptech.eu
- 中國網站：www.icop.com.cn

如需技術支援或驅動程式下載，請造訪：

- https://www.icop.com.tw/resource_entrance

若需 EtherCAT 解決方案服務、支援或教學，以及 86Duino Coding IDE 500+ 的介紹、函式、語言與函式庫說明，請造訪 QEC 官方網站：

- QEC：https://www.qec.tw/

本手冊適用於 QEC 系列產品。

安全資訊

- 請仔細閱讀本安全說明。
- 搬運設備時請雙手托持，並小心處理。
- 電源輸入電壓為 +19 至 +50VDC (典型值：+24VDC)。
- 在將設備接上電源插座之前，請確認電源電壓是否適當。
- 為防止 QEC 裝置發生電擊或火災危險，請保持設備乾燥，遠離水源與潮濕環境。
- 操作溫度範圍為 -20 至 +70°C (選配為 -40 至 +85°C)。
- 若使用外接儲存裝置作為作業系統主儲存媒介，請在連接或拆除前確認設備已關機。
- 未切斷電源轉接器前，請勿觸碰未絕緣的端子或電線。
- 請將裝置安置於靠近插座的位置，以便操作並避免手臂纏繞於周圍電纜而造成拉扯。
- 若裝置長時間不使用，請務必切斷電源，以避免瞬間過電壓損壞設備。

警告！

請勿嘗試打開或拆解本產品的機殼。
如需維修服務，請聯繫您的經銷商，由合格技術人員處理。

內容

章節. 1 介紹 1
1.1 關於 QEC EtherCAT 主站 3
1.1.1 什麼是 86Duino IDE? 3
1.1.2 QEC EtherCAT 主站架構 4
1.1.3 硬體平台 5
1.1.4 雙系統同步機制 6
1.2 功能特點 7
1.2.1 功能表 7
1.3 功能擴充套件 10
1.3.1 電纜冗餘 (Cable Redundancy) 10
1.4 效能評估 (Benchmark) 14
1.4.1 系統變數 14
1.4.2 測量功能 15
1.4.3 測量結果 17
1.4.4 範例程式碼 20
1.4.5 EtherCAT 主站週期封包的抖動（Jitter） 22
1.5 同步 23
1.5.1 Free Run 模式 24
1.5.2 SM-Synchronous 模式 25
1.5.3 DC-Synchronous 模式 26
章節. 2 功能函式 28
2.1 EtherCAT MDevice 30
2.1.1 初始化函式 38
2.1.2 控制函式 48
2.1.3 Callback 函式 61
2.1.4 從站裝置資訊函式 78
2.2 EtherCAT SubDevice 87
2.2.1 _EthercatDevice_CommonDriver 88
2.2.2 EthercatDevice_Generic 164
2.2.3 EthercatDevice_CiA402 169
2.3 QEC-Series SubDevice 174
2.3.1 _EthercatDevice_DmpCommonDriver 176
2.3.2 EthercatDevice_DmpDIQ_Generic 199
2.3.3 EthercatDevice_DmpAIQ_Generic 212
2.3.4 EthercatDevice_DmpCIQ_Generic 227

2.3.5 EthercatDevice_DmpHID_Generic 284
2.3.6 EthercatDevice_DmpLCD_Generic 365
2.3.7 EthercatDevice_DmpStepper_Generic 431
章節. 3 範例 544
3.1 從站裝置資訊範例 545
3.1.1 範例 1：使用 EthercatMaster 類別 545
3.1.2 範例 2：使用 EthercatDevice_Generic 類別 547
3.1.3 範例 3：使用 EthercatDevice_CiA402 類別 549
3.2 SDO 上傳/下載範例 551
3.2.1 範例 1：使用 sdoUpload8() 進行 SDO 讀取 551
3.2.2 範例 2：使用 sdoUpload() 進行 SDO 讀取 552
3.2.3 範例 3：使用 sdoUpload() 進行 SDO 讀取並處理中止碼 553
3.2.4 範例 4：使用 sdoDownload8() 進行 SDO 寫入 554
3.2.5 範例 5：使用 sdoDownload() 進行 SDO 寫入 555
3.2.6 範例 6：使用 sdoDownload() 進行 SDO 寫入並處理中止碼 556
3.2.7 範例 7：列印出 PDO 映射設定 557
3.2.8 範例 8：更改 PDO 映射設定 559
3.3 PDO 讀/寫範例 561
3.3.1 範例 1：使用 pdoBitRead() 讀取輸入 PDO 的 1 位元資料 561
3.3.2 範例 2：使用 pdoRead8() 讀取輸入 PDO 的 1 Byte 資料 562
3.3.3 範例 3：使用 pdoRead() 從輸入 PDO 讀取資料 563
3.3.4 範例 4：使用 pdoBitWrite() 寫入輸出 PDO 的 1 位元資料 564
3.3.5 範例 5：使用 pdoWrite8() 寫入輸出 PDO 的 1 Byte 資料 565
3.3.6 範例 6：使用 pdoWrite() 寫入輸出 PDO 資料 566
3.4 Callback 範例 567
3.4.1 範例 1：Cyclic callback 567
3.4.2 範例 2：啟用 FPU 的 Cyclic callback 569
3.4.3 範例 3：Error callback 571
3.4.4 範例 4：Event callback 573
3.5 DC 範例 574
3.5.1 範例 1：啟用 DC 同步 574
附錄 576
A.1 錯誤代碼 577
A.2 錯誤說明與修正措施 580
A.3 Error Callback Code 605
A.4 Event Callback Code 606
A.5 SDO Abort Code 607
A.6 資料型別 609

A.7 EtherCAT Network Information.....611
保固.....614

章節. 1

介紹

EtherCAT（Ethernet for Control Automation Technology）是一種專為工業控制應用設計的通訊協定。它是一項高效能、即時性的通訊技術，具有極低的通訊延遲與高頻寬，非常適用於各類工業自動化與控制系統。

EtherCAT 的主要特點包括：

- **高速即時性**：EtherCAT 能以極高的速度傳輸資料，同時保持極低的通訊延遲，實現高等級的即時效能，適用於對控制反應要求嚴苛的應用場景。
- **靈活的擴展性**：EtherCAT 網路架構具有高度可擴展性，能夠輕鬆支援大量節點與複雜的網路拓撲，適用於不同規模與需求的應用情境。
- **開放標準**：EtherCAT 採用開放標準，獲得廣泛的產業支援與應用，並擁有豐富的函式庫與工具，可輕鬆整合至現有的工業自動化系統中。
- **高成本效益**：EtherCAT 使用標準的乙太網路硬體，無需額外專用設備，能有效降低成本與部署複雜度。

EtherCAT 廣泛應用於工業自動化、機器人控制、運動控制、嵌入式系統以及即時資料擷取等領域，為這些應用提供高效能且穩定可靠的通訊解決方案。

1.1 關於 QEC EtherCAT 主站

QEC（Quicker, Easier Control with EtherCAT）是由 ICOP 所開發的 EtherCAT 解決方案，具備高同步性、即時性與易於開發等特點。以下將介紹 QEC 的架構、特色與效能。
QEC 主站是一款相容於 86Duino Coding IDE 500+ 的 EtherCAT 主站系統，支援 EtherCAT 主站與多個 EtherCAT 從站之間的即時通訊。除了在 86Duino IDE 中提供的 EtherCAT 函式庫外，QEC 主站也支援 Modbus、Ethernet TCP/IP、CAN bus 等多種工業通訊協定，並可透過高階的 C/C++ 程式語言快速開發應用程式。

1.1.1 什麼是 86Duino IDE?

86Duino 整合式開發環境（IDE）可輕鬆撰寫並上傳程式碼至 86Duino 開發板與 QEC MDevice。該環境支援 Windows、Mac OS X 及 Linux 系統，使用 Java 編寫，並基於 Arduino IDE、Processing、DJGPP 及其他開源軟體開發而成，可於以下網址下載：
https://www.qec.tw/software/。

QEC 主站所搭配的 86Duino IDE，也提供一套名為 **86EVA** 的圖形化設定工具，方便使用者進行 EtherCAT 網路的參數設定，其主要功能包括：

- 掃描 EtherCAT 從站
- 匯入 ENI 檔案
- 設定 EtherCAT 主站
- 配置 EtherCAT 從站

更多詳細功能，請參閱 86EVA 使用手冊。

1.1.2 QEC EtherCAT 主站架構

EtherCAT 主站軟體僅能運行於由 DM&P 所生產的 Vortex86EX2 處理器上，該處理器具備雙系統架構。EtherCAT 主站軟體主要分為兩個部分，分別在 Vortex86EX2 的兩個系統上執行。

這兩個部分的職責如下：

- **EtherCAT 主站函式庫**
    - 提供 C/C++ 應用介面，包括：
        - 初始化介面
        - 組態設定介面
        - 處理資料（PDO）存取介面
        - 基於 EtherCAT 的 CAN 應用協議（CoE）存取介面
        - 基於 EtherCAT 的檔案存取（FoE）介面
        - 子裝置資訊（SII）存取介面
        - 分散式時鐘（DC）存取介面
- **EtherCAT 主站韌體**
    - 執行 EtherCAT 主戰核心程式
    - 控制主/副乙太網路驅動程式，傳送 EtherCAT 封包

這些程式運行於 FreeDOS 作業系統，並使用 DJGPP 開發環境中提供的 GCC 編譯器進行編譯。

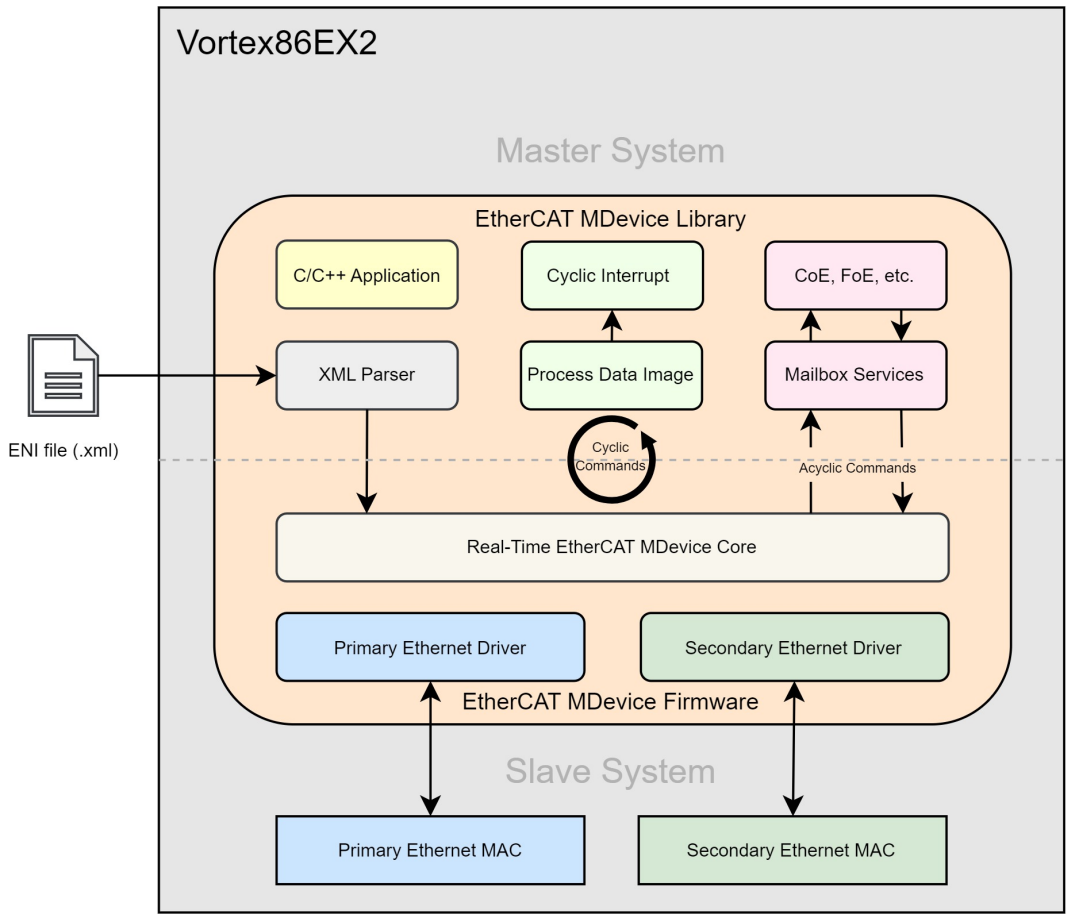
1.1.3 硬體平台

EtherCAT 主站軟體僅能運行於由 DM&P 所生產的 Vortex86EX2 處理器上，該處理器具備雙系統架構，分為 **主系統 (Master System)** 與 **從系統 (Slave System)**，各自運行獨立的作業系統，並透過 **雙埠記憶體 (Dual-Port RAM)** 與 **事件中斷 (Event Interrupts)** 進行系統間通訊。

各系統負責的任務如下：

- **主系統（Master System）**
  - 執行使用者的 EtherCAT 應用程式
  - 執行使用者的 HMI 人機介面程式
  - 執行使用者的乙太網路應用程式
  - 其他應用程式等
- **從系統（Slave System）**
  - 專責執行 EtherCAT 主站韌體

由於大多數應用程式皆在主系統中執行，使得從系統上的 EtherCAT 主站韌體能避免受到其他應用的干擾，專注於執行 EtherCAT 主站核心，從而確保 EtherCAT 的高同步性與即時效能。



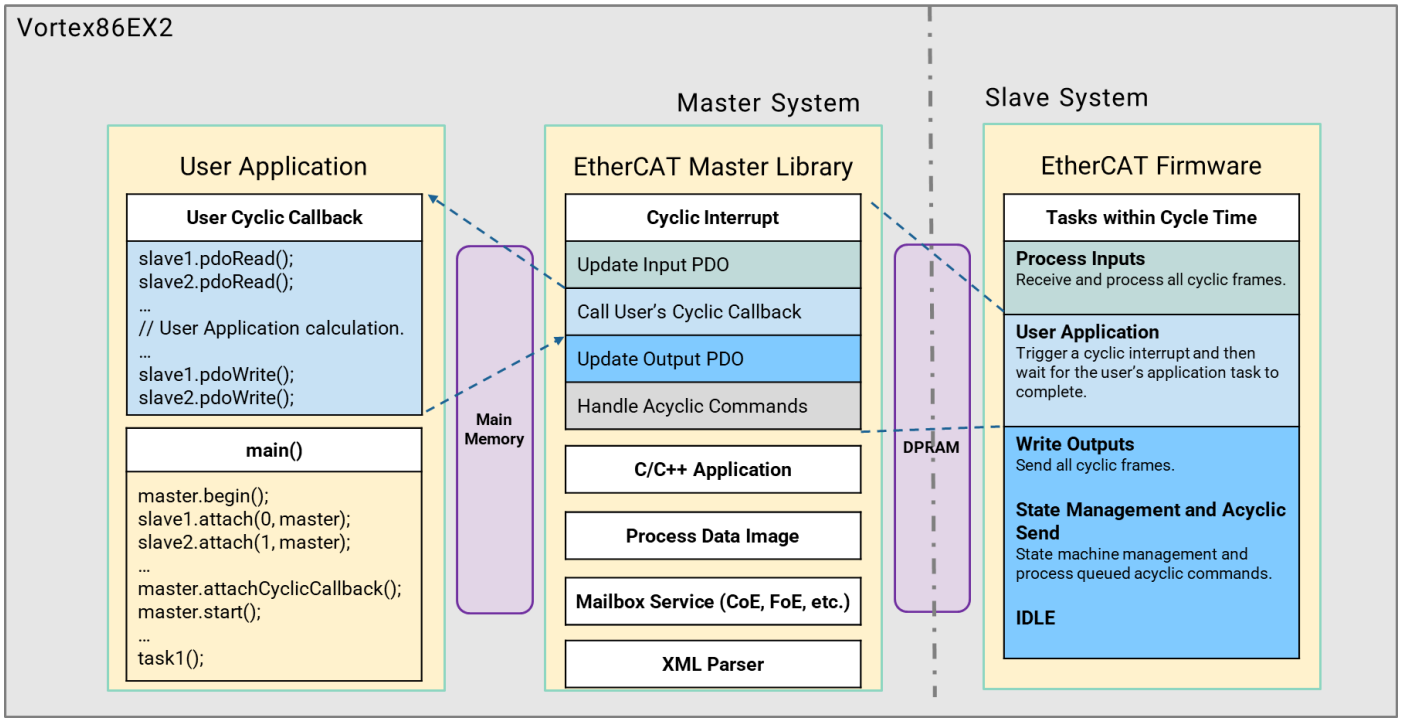
1.1.4 雙系統同步機制

本節的主要重點是雙系統 PDO 資料的同步。如圖所示，**使用者應用程式**與 **EtherCAT 主站函式庫模組**運行於主系統，而**即時 EtherCAT 主站核心**運行於從系統。
當 EtherCAT 主站核心執行至 **Process Inputs (處理輸入)** 階段時，它會從乙太網路驅動接收所有週期性封包，並將輸入 PDO 資料複製至 DPRAM (雙埠記憶體)。

當進入 **User Application (使用者應用程式)** 階段時，EtherCAT 主站核心會觸發一個 **週期性中斷 (Cyclic Interrupt)** 給主系統。主系統收到中斷後，會執行 EtherCAT 主站函式庫的中斷處理流程：它會將輸入 PDO 資料從 DPRAM 移至主記憶體，呼叫使用者註冊的 **Cyclic Callback**，回呼完成後再將輸出 PDO 資料從主記憶體移回 DPRAM，處理非週期命令，並結束中斷處理流程。

此時，**EtherCAT 主站核心的使用者應用程式**與**中斷處理流程**同步完成。
當 EtherCAT 主站核心執行至 **Write Outputs (寫入輸出)** 階段時，會將輸出 PDO 資料從 DPRAM 複製至乙太網路驅動的 DMA，並發送封包。

這些任務以週期方式循環執行，依照上述步驟進行，確保雙系統 PDO 資料的同步性。



1.2 功能特點

EtherCAT 技術協會在規範文件 ETG.1500 中，定義了兩種 EtherCAT 主站軟體實作的類別。此規範針對主站的功能性，制定了明確的分類標準。

為簡化設計，僅定義了兩種 主站類別：

- **Class A**：標準 EtherCAT 主站
- **Class B**：最小化 EtherCAT 主站

以下將說明 **Class A**、**Class B**，以及我們的 **QEC 主站**之間的功能比較。

1.2.1 功能表

用詞說明：

- **shall** 表示「**必須**」
- **should** 表示「**建議**」
- **may** 表示「**允許**」
- **O** 表示「**支援**」

功能名稱	簡要說明	Class A	Class B	QEC 主站
Basic Features				
Service Commands	Support of all commands	shall if ENI import support		O
IRQ field in datagram	Use IRQ information from SubDevice in datagram header	should	should	--
SubDevices with Device Emulation	Support SubDevices with and without application controller	shall	shall	O
EtherCAT State Machine	Support of ESM special behavior	shall	shall	O
Error Handling	Checking of network or slave errors, e.g. Working Counter	shall	shall	O
VLAN	Support VLAN Tagging	may	may	--
EtherCAT Frame Types	Support EtherCAT Frames	shall	shall	O
UDP Frame Types	Support UDP Frames	may	may	--

Process Data Exchange				
Cyclic PDO	Cyclic process data exchange	shall	shall	O
Multiple Tasks	Different cycle tasks. Multiple update rates for PDO	may	may	--
Frame repetition	Send cyclic frames multiple times to increase immunity	may	may	--
Network Configuration				
Online scanning	Network configuration functionality included in EtherCAT MDevice	at least one of them		O
Reading ENI	Network Configuration taken from ENI file	at least one of them		O
Compare Network configuration	Compare configured and existing network configuration during boot-up	shall	shall	O
Explicit Device Identification	Identification used for Hot Connect and prevention against cable swapping	should	should	O
Station Alias Addressing	Support configured station alias in SubDevice	may	may	--
Access to EEPROM	Support routines to access EEPROM via ESC register	Read shall / Write may		O
Mailbox Support				
Support Mailbox	Main functionality for mailbox transfer	shall	shall	O
Mailbox Resilient Layer	Support underlying resilient layer	shall	shall	O
Multiple Mailbox channels	Simultaneous Mailbox protocol transfer to one device	may	may	--
Mailbox polling	Polling Mailbox state in SubDevices	shall	shall	--
CAN application layer over EtherCAT (CoE)				
SDO Up/Download	Normal and expedited transfer	shall	shall	O
Segmented Transfer	Segmented transfer	shall	should	O
Complete Access	Transfer the entire object (with all sub-indices) at once	shall	should (shall if ENI Import supported)	O
SDO Info service	Services to read object dictionary	shall	should	O
Emergency Message	Receive Emergency messages	shall	shall	--
PDO in CoE	PDO services transmitted via CoE	may	may	--
EoE				
EoE protocol	Services for tunneling Ethernet frames. includes all specified EoE services	shall	shall if EoE support	--

Virtual Switch	Virtual Switch functionality	shall	shall if EoE support	--
EoE Endpoint to Operation Systems	Interface to the Operation System on top of the EoE layer	should	should if EoE support	--
FoE				
FoE Protocol	Support FoE Protocol	shall	shall if FoE support	O
Firmware Up/Download	Password, FileName should be given by the application	shall	should	O
Boot State	Support Boot-State for Firmware Up/Download	shall	shall if FW UP/Download	--
SoE				
SoE Services	Support SoE Services	shall	shall if SoE support	--
AoE				
AoE Protocol	Support AoE Protocol	should	should	--
VoE				
VoE Protocol	External Connectivity supported	may	may	--
Synchronization with Distributed Clock (DC)				
DC support	Support of Distributed Clock	shall	shall if DC support	O
Continuous Propagation Delay compensation	Continuous Calculation of the propagation delay	should	should	--
Sync window monitoring	Continuous monitoring of the Synchronization difference in the SubDevices	should	should	--
SubDevice-to-SubDevice Communication				
via MDevice	Information is given in ENI file or can be part of any other network configuration Copying of the data can be handled by MDevice stack or MDevice's application	shall	shall	--
MDevice information				
MDevice Object Dictionary	Support of MDevice Object Dictionary (ETG.5001 MDP sub profile 1100)	should	may	--

1.3 功能擴充套件

1.3.1 電纜冗餘 (Cable Redundancy)

EtherCAT 電纜冗餘是指 EtherCAT 通訊系統即使在電纜故障時也能保持連續可靠通訊的能力。電纜冗餘採用環形拓撲結構，可雙向運作。如果一條電纜發生故障或斷開，另一條電纜路徑仍可運作以確保不間斷通訊。電纜冗餘增強了 EtherCAT 網路的容錯能力，最大限度地減少了停機時間並提高了整體系統的可靠性。

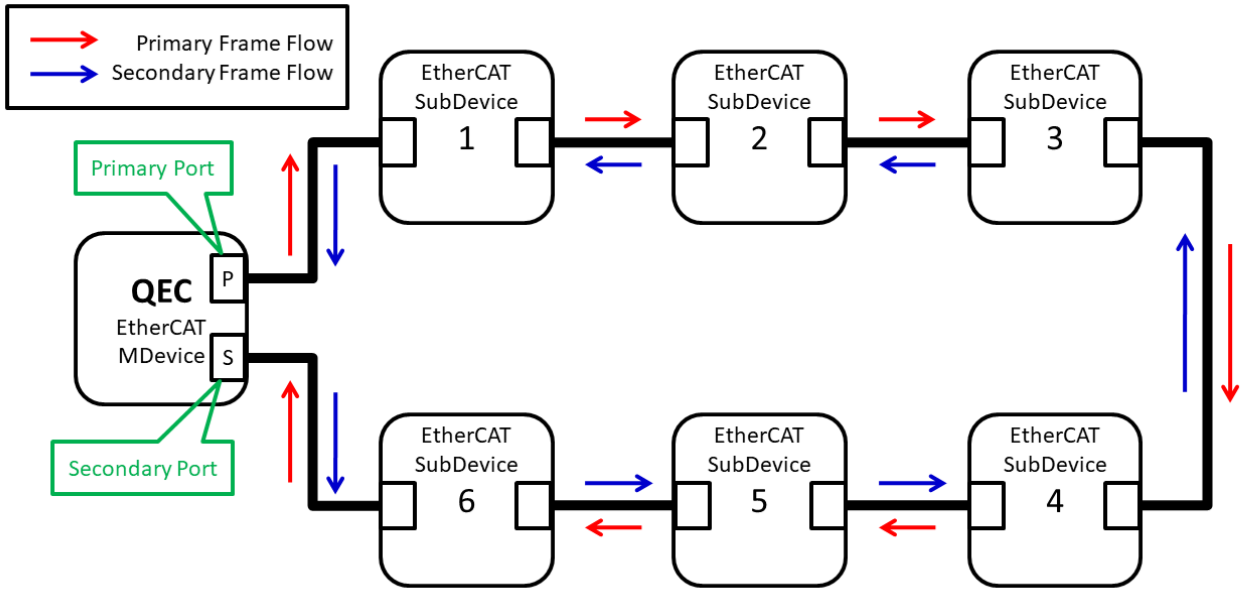
以下列出三種情境，說明在備援模式下網路是否發生斷線。接下來您將看到如何實現電纜冗餘，以及這些場景之間 EtherCAT 主站控制器的差異。

- 無網路斷線
- 從站裝置之間的網路線斷線
- 主站與從站裝置之間的網路線斷線

為了便於說明，本文採用以下假設條件：

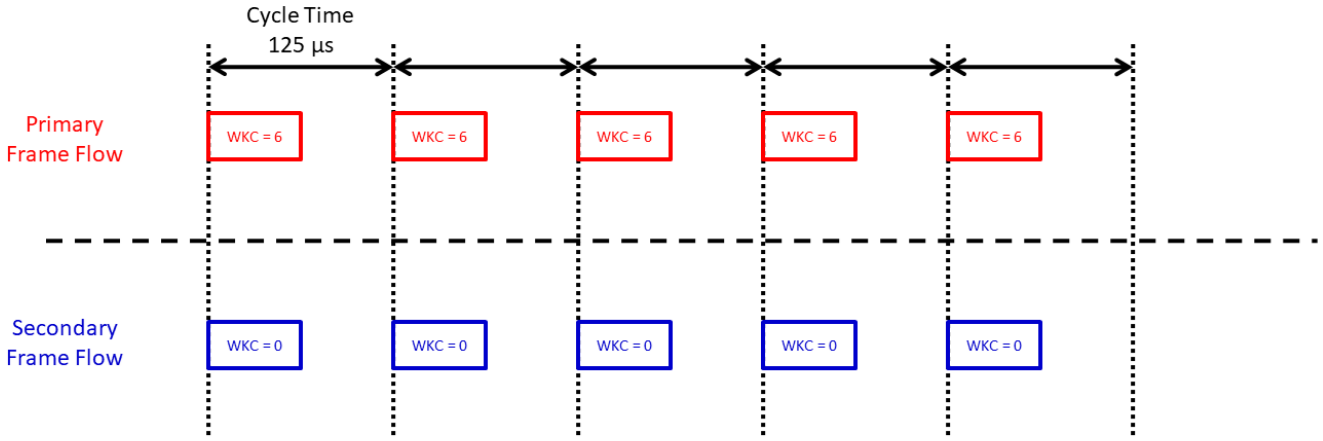
- 假設通訊週期時間 (Cycle Time) 為 125 微秒 (μs)。
- 假設所有從站裝置僅具備輸入 PDO (無輸出 PDO)，每當封包通過一個從站裝置時，其處理資料封包中的工作計數器 (WKC) 將增加 1。
- 假設 EtherCAT 網路中共有 6 個從站裝置，因此期望工作計數器 (EWKC) 為 6。
- 主通訊埠 (Primary Port) 與備援通訊埠 (Secondary Port) 會在每個週期中**同時發送**處理資料封包。

情境一：無網路線斷線



無網路線斷線

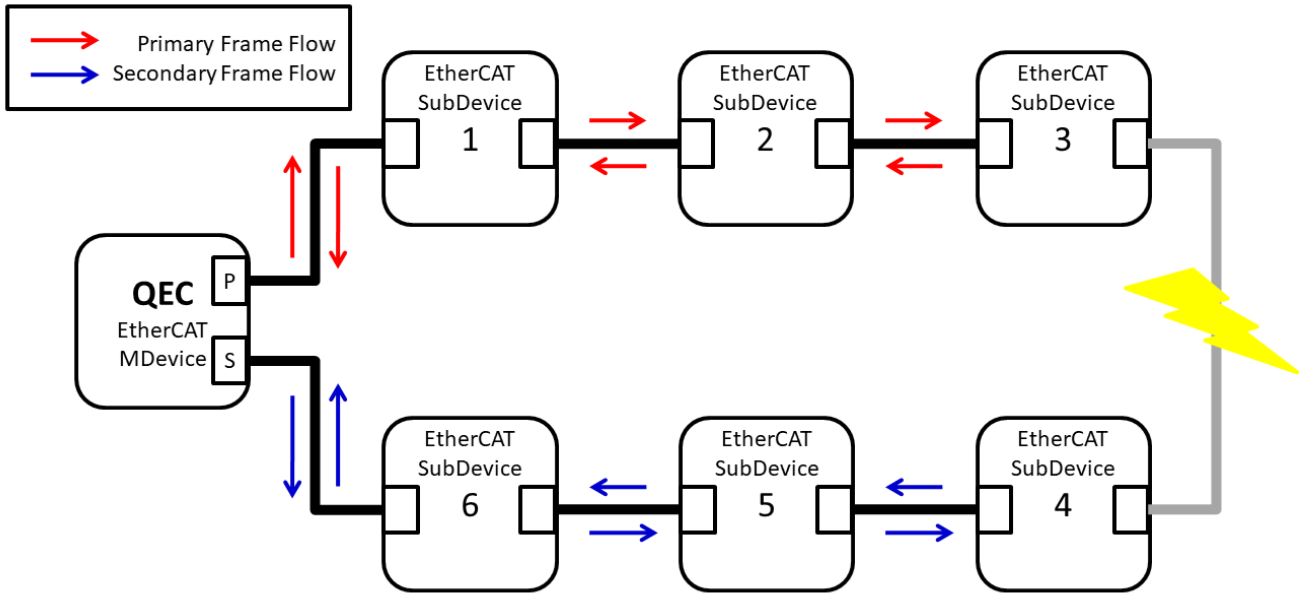
- 從主通訊埠發出的主封包，將會被從備援通訊埠接收，此封包的工作計數器為 6。
- 從備援通訊埠發出的備援封包，將會被從主通訊埠接收，此封包的工作計數器為 0。
- 因為主封包的工作計數器等於期望工作計數器，且備援封包的工作計數器為 0，這代表網路線未斷線，因此主站將捨棄備援封包。



情境二：從站裝置之間的網路線斷線

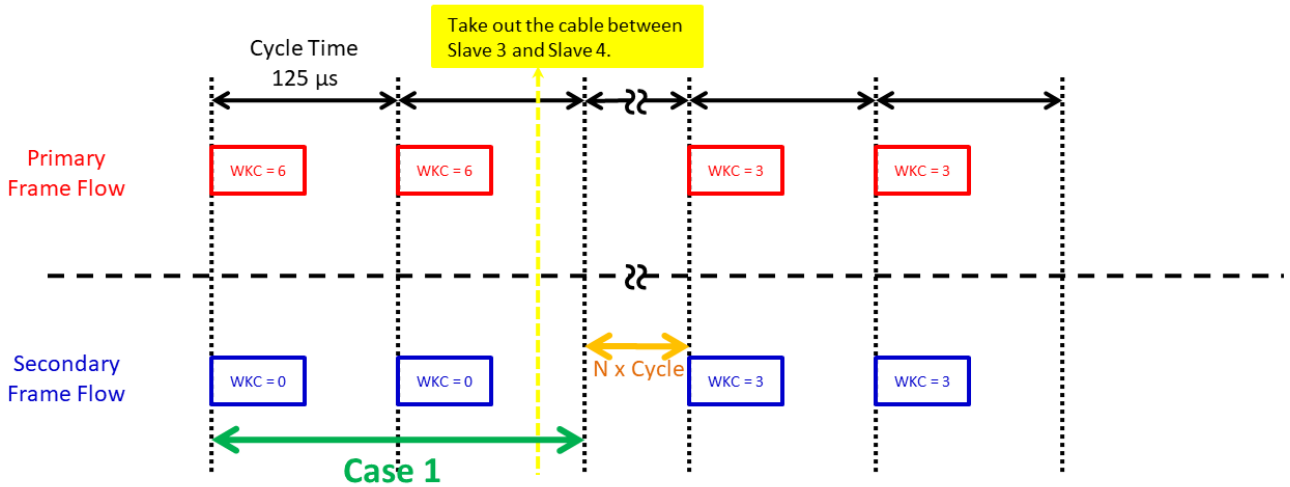
在從站裝置 3 與從站裝置 4 之間拔除網路線：

- 若您手動拔除網路線，可能會產生干擾，該干擾可能持續數個週期 (N = 0 ~ 多個週期)。
- 若干擾持續一段時間，部分子裝置可能會因 SyncManager Watchdog 偵測異常而進入 SAFEOP 安全操作模式。



從站裝置 3 與從站裝置 4 之間的網路線斷線：

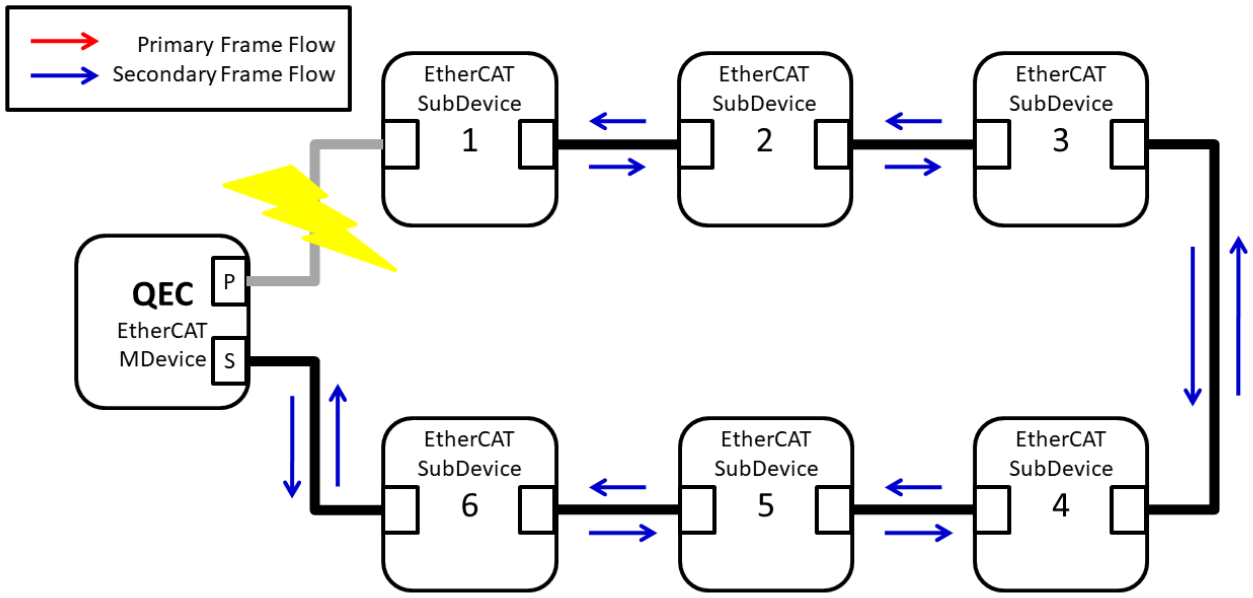
- 從主通訊埠發出的主封包會從主通訊埠接收，該封包的工作計數器為 3。
- 從備援通訊埠發出的備援封包會從備援通訊埠接收，該封包的工作計數器也為 3。
- 因為主封包的工作計數器小於期望工作計數器，而備援封包的工作計數器大於 0，這代表兩個子裝置之間發生了網路線中斷事件，因此主站將主封包與備援封包進行合併處理。



情境三：主站與 從站之間的網路線斷線

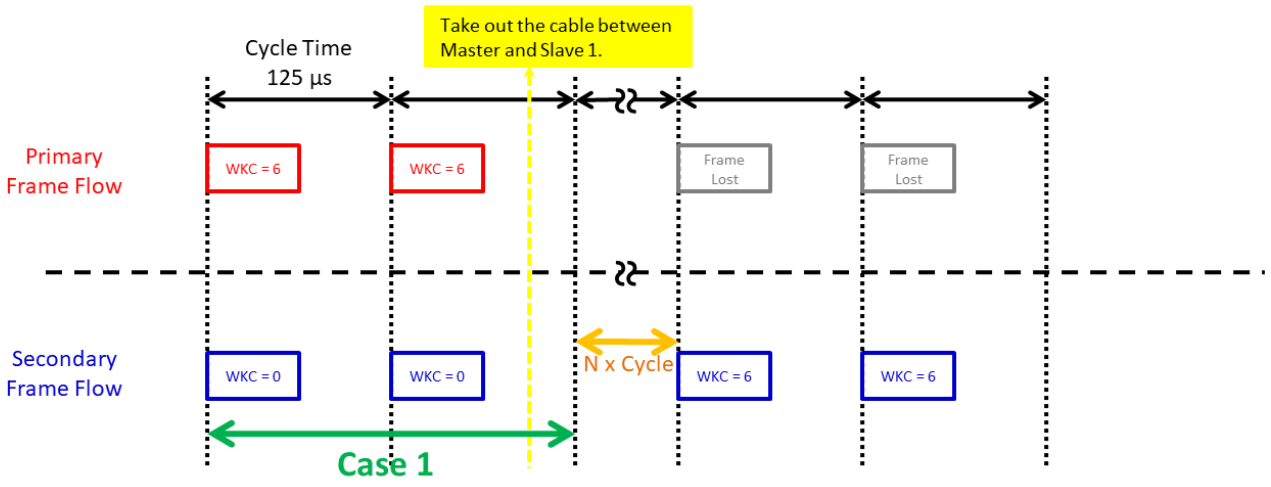
在主站與從站裝置 1 之間拔除網路線：

- 若您手動拔除網路線，可能會產生干擾，該干擾可能持續數個週期 (N = 0 ~ 多個週期)。
- 若干擾持續一段時間，部分子裝置可能會因 SyncManager Watchdog 偵測異常而進入 SAFEOP 安全操作模式。



主站與從站裝置 1 之間的網路線斷線：

- 從主通訊埠發出的主封包將會遺失。
- 從備援通訊埠發出的備援封包會由備援通訊埠接收，該封包的工作計數器為 6。
- 由於主封包已遺失，而備援封包的工作計數器等於期望工作計數器，這代表主站與從站裝置 1 之間發生了網路線中斷事件，因此主站將忽略主封包。



1.4 效能評估 (Benchmark)

EtherCAT 是一種以高同步性能著稱的現場總線技術。在需要高度同步的應用中，通常也會對即時效能與高控制頻率提出要求。

在這類應用情境中，使用者通常會關注以下幾項規格：

- 是否支援更短的控制週期時間
- 是否支援更大量的處理資料（Process Data）
- 是否支援更多的 EtherCAT 從站裝置

然而，要評估一個 EtherCAT 主站是否符合使用者的應用需求，通常需以**效能基準測試數據作為主要依據**。

1.4.1 系統變數

以下因素與變數會影響 EtherCAT 系統可達成的效能，以及週期時間的選擇：

- **網路線長度**
  影響網路封包的傳輸延遲。

- **從站裝置數量**
  會增加網路封包的傳輸延遲；每個子裝置約會造成 0.3 至 1 微秒 的延遲。

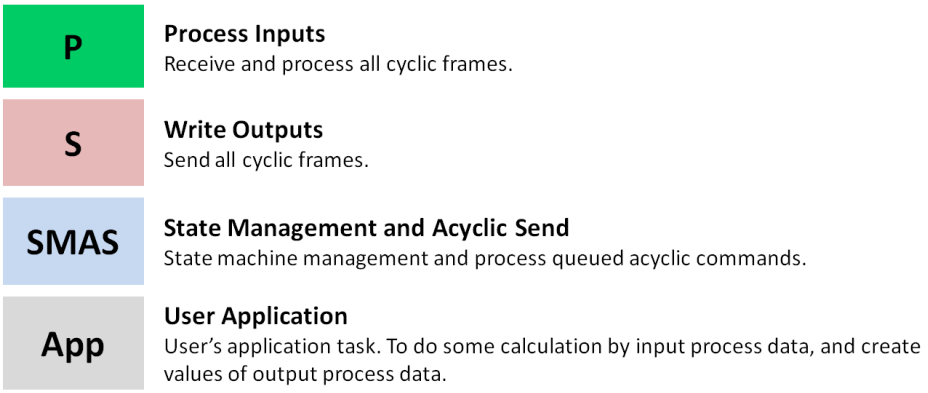
- **子裝置處理資料位元組數 (Process Data Bytes)**
  決定網路封包的長度。

- **從站裝置同步模式（Synchronous Mode）**
  受到 EtherCAT 從站裝置在應用中處理效能的限制。

- **主站的計算效率**
  EtherCAT 主站的效率不僅影響週期時間，亦影響同步精度。其效率取決於處理器運算速度、軟體架構與效率、記憶體傳輸速度等因素。

1.4.2 測量功能

對於具備即時性需求的 EtherCAT 應用而言，在週期時間內進行精確的任務排程是關鍵，可有效減少週期抖動（Cycle Time Jitter）。在 QEC 主站軟體中，週期時間被劃分為四個階段，每個階段負責不同的任務，如下所示：



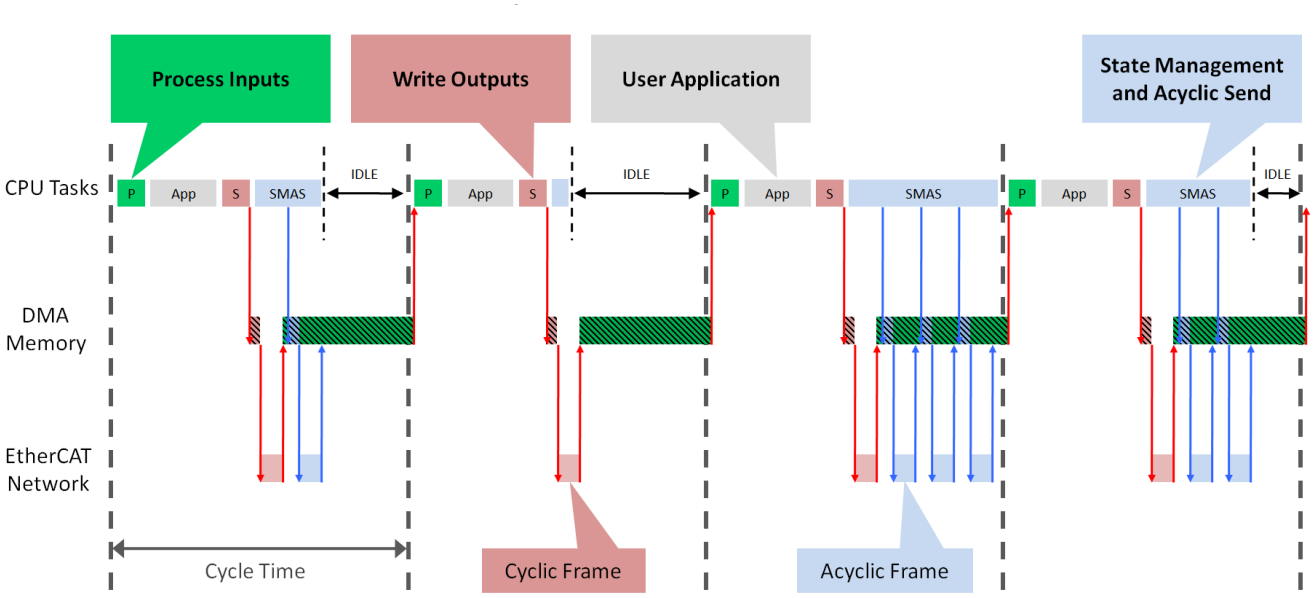
在預設的週期模式下，週期內各任務的時序圖如下：

- **處理輸入（Process Inputs）**
  這是週期開始的第一個任務，由 EtherCAT MDevice 韌體執行，負責接收並處理週期性處理資料，並將輸入處理資料（Input Process Data）傳送至雙系統的共用記憶體中。

- **使用者應用程式（User Application）**
  EtherCAT 主站韌體會向主系統發出週期性中斷（Cyclic Interrupt）。主系統會將輸入處理資料從共用記憶體搬移至主記憶體，讓使用者能讀取前一週期的處理資料。隨後，系統會呼叫使用者註冊的 Cyclic Callback，使用者可在正確的控制時間點對 EtherCAT 從站裝置進行操作。接著，使用者將本週期的輸出處理資料（Output Process Data）寫入主記憶體。主系統在回呼完成後，會將輸出資料搬回共用記憶體，並回應 EtherCAT 主站韌體：使用者應用程式執行完畢。

- **寫入輸出（Write Outputs）**
  EtherCAT 主站韌體在接收到主系統回應或逾時後，執行此任務。該任務會將共用記憶體中的輸出處理資料打包為 EtherCAT 的週期性處理資料封包，並發送至所有子裝置。

- **狀態管理與非週期傳輸（State Management and Acyclic Send）**
  這是 EtherCAT 主站韌體在每個週期時間內的最後一個任務，負責 EtherCAT 狀態機的封包傳送、接收與處理，同時也處理非週期資料傳輸（例如：Mailbox、CoE）。



1.4.3 測量結果

以下測試結果是在相同控制器上，設定不同週期時間並連接 5 個從站裝置所進行的效能測試。

這 5 個從站裝置分別為：

- QEC-R00DC4D：12 路數位輸入、4 路數位輸出
- QEC-R00D4CD：4 路數位輸入、12 路數位輸出
- QEC-R00D88D：8 路數位輸入、8 路數位輸出
- QEC-R00D0FS：16 路數位輸出
- QEC-R00DF0D：16 路數位輸入

平台條件如下：

- 處理器：Vortex86EX2 600/400 MHz
- 作業系統：FreeDOS
- 編譯器：GCC 8.3.0
- 封包負載（Payload）: 60 Bytes

A. 設定週期時間為 125 微秒 (µs)

啟用非週期傳輸 (Acyclic Transfer)。

MDevice Function	min.	avg.	max.
Measured cycle time.	124.90	124.92	125.09
Process Inputs.	6.36	7.26	10.81
Write Outputs.	7.22	7.35	11.89
State Management and Acyclic Send.	2.46	2.48	34.89

未啟用非週期傳輸。

每秒最大 SDO 命令處理量為 **769.48**。

MDevice Function	min.	avg.	max.
Measured cycle time.	124.85	124.86	125.12
Process Inputs.	4.79	4.88	9.80
Write Outputs.	7.91	8.08	14.06
State Management and Acyclic Send.	2.47	60.55	96.30

B. 設定週期時間為 250 微秒 (µs)

啟用非週期傳輸。

MDevice Function	min.	avg.	max.
Measured cycle time.	249.92	249.93	250.11
Process Inputs.	5.99	7.83	10.42
Write Outputs.	7.90	7.92	12.29
State Management and Acyclic Send.	2.33	2.33	26.66

未啟用非週期傳輸。

每秒最大 SDO 命令處理量為 **1409.11**。

MDevice Function	min.	avg.	max.
Measured cycle time.	249.86	249.87	250.19
Process Inputs.	5.42	5.50	13.03
Write Outputs.	8.39	8.66	13.27
State Management and Acyclic Send.	2.90	12.27	227.15

C. 設定週期時間為 500 微秒 (µs)

啟用非週期傳輸。

MDevice Function	min.	avg.	max.
Measured cycle time.	499.93	499.93	500.08
Process Inputs.	6.13	7.91	10.26
Write Outputs.	7.91	7.98	12.50
State Management and Acyclic Send.	2.28	2.34	35.17

未啟用非週期傳輸。

每秒最大 SDO 命令處理量為 **1657.79**。

MDevice Function	min.	avg.	max.
Measured cycle time.	499.81	499.83	500.19
Process Inputs.	5.71	6.01	14.03
Write Outputs.	8.84	9.04	14.14
State Management and Acyclic Send.	2.77	123.89	474.36

D. 設定週期時間為 1000 微秒 (µs)

啟用非週期傳輸。

MDevice Function	min.	avg.	max.
Measured cycle time.	999.95	999.96	1000.07
Process Inputs.	6.57	8.29	10.11
Write Outputs.	7.99	8.01	11.72
State Management and Acyclic Send.	2.32	2.33	34.71

未啟用非週期傳輸。

每秒最大 SDO 命令處理量為 **999.99**。

MDevice Function	min.	avg.	max.
Measured cycle time.	999.93	999.94	1000.08
Process Inputs.	6.42	6.60	9.18
Write Outputs.	10.32	11.57	14.03
State Management and Acyclic Send.	9.15	345.94	593.68

1.4.4 範例程式碼

以下為效能測試的範例程式碼，測試設定的週期時間為 **1000 微秒**。

```
#include <Ethercat.h>

EthercatMaster master;

void setup() {
  Serial.begin(115200);

  EthercatMasterSettings settings;
  EthercatBenchmark benchmark;

  master.readSettings(&settings);
  settings.EnableBenchmarkMeasurement = 1;
  master.saveSettings(&settings);

  if (master.begin() < 0) {
    Serial.println("ERROR: master.begin() failed.");
    return;
  }

  // Start EtherCAT with a cycle time of 1000000 ns (1ms)
  if (master.start(1000000) < 0) {
    Serial.println("ERROR: master.start() failed.");
    master.end();
    return;
  }

  delay(30000);  // Delay for 30 seconds to collect benchmark data
  master.getBenchmarkResult(&benchmark);

  Serial.println();
  Serial.println("|================================================|");
  Serial.println("| [C]    Cycle Time.      (min/avg/max) [usec]: ");
```

```
  Serial.print("    "); Serial.print(benchmark.CycleTime.min / 1000.0, 2);
  Serial.print(" / "); Serial.print(benchmark.CycleTime.avg / 1000.0, 2);
  Serial.print(" / "); Serial.println(benchmark.CycleTime.max / 1000.0, 2);

  Serial.println("| [P]    Receive PDO.     (min/avg/max) [usec]: ");
  Serial.print("    "); Serial.print(benchmark.ReceiveCyclicFrame.min / 1000.0, 2);
  Serial.print(" / "); Serial.print(benchmark.ReceiveCyclicFrame.avg / 1000.0, 2);
  Serial.print(" / "); Serial.println(benchmark.ReceiveCyclicFrame.max / 1000.0, 2);

  Serial.println("| [S]    Send PDO.        (min/avg/max) [usec]: ");
  Serial.print("    "); Serial.print(benchmark.SendCyclicFrame.min / 1000.0, 2);
  Serial.print(" / "); Serial.print(benchmark.SendCyclicFrame.avg / 1000.0, 2);
  Serial.print(" / "); Serial.println(benchmark.SendCyclicFrame.max / 1000.0, 2);

  Serial.println("| [AS]   Process Acyclic.  (min/avg/max) [usec]: ");
  Serial.print("    "); Serial.print(benchmark.ProcessAcyclicFrame.min / 1000.0, 2);
  Serial.print(" / "); Serial.print(benchmark.ProcessAcyclicFrame.avg / 1000.0, 2);
  Serial.print(" / "); Serial.println(benchmark.ProcessAcyclicFrame.max / 1000.0,
2);

  Serial.println("| [App.] User Application. (min/avg/max) [usec]: ");
  Serial.print("    "); Serial.print(benchmark.UserApp.min / 1000.0, 2);
  Serial.print(" / "); Serial.print(benchmark.UserApp.avg / 1000.0, 2);
  Serial.print(" / "); Serial.println(benchmark.UserApp.max / 1000.0, 2);

  Serial.println("|=====================================================|");
}

void loop() {
  // ...
}
```

1.4.5 EtherCAT 主站週期封包的抖動（Jitter）

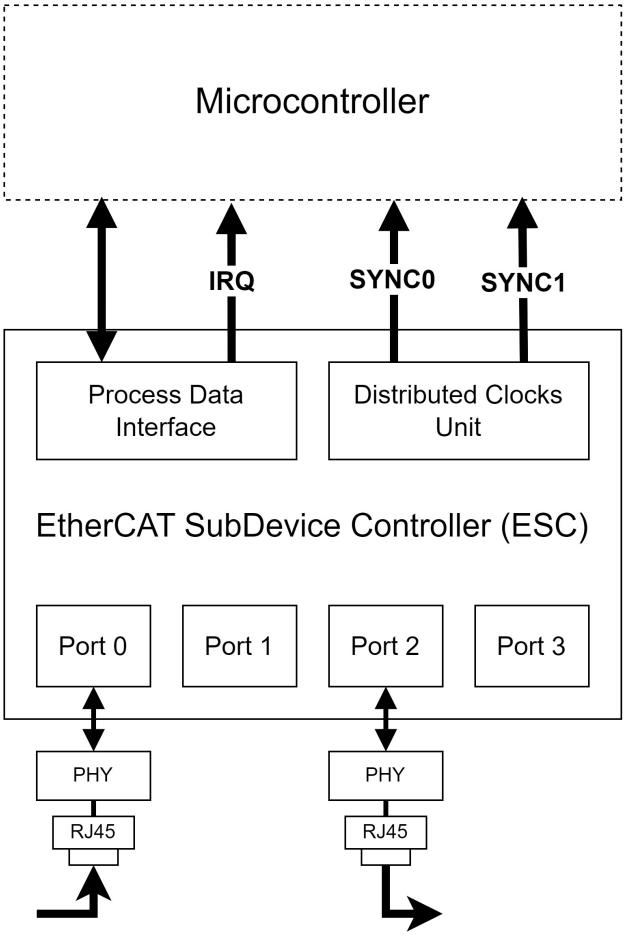
此為 EtherCAT 主站傳送週期性封包時所產生的抖動，參考基準為 **DC SYNC0**（分散式時鐘的同步訊號 0）。

- 影片：https://youtu.be/O888jD4XUsY?si=NaI9gsafyA1D2DlK

1.5 同步

在 EtherCAT 網路中，所有從站裝置之間的時間同步是透過 EtherCAT 從站裝置控制器 (ESC) 中的**分散式時鐘 (Distributed Clocks, DC)** 單元來實現，以確保整個系統的時間一致性。

通常，網路中**第一個具備 DC 功能的從站裝置**會作為**系統的參考時鐘**，用來同步其他具備 DC 功能的從站裝置。關於**分散式時鐘**的詳細說明，請參閱「分散式時鐘」章節。

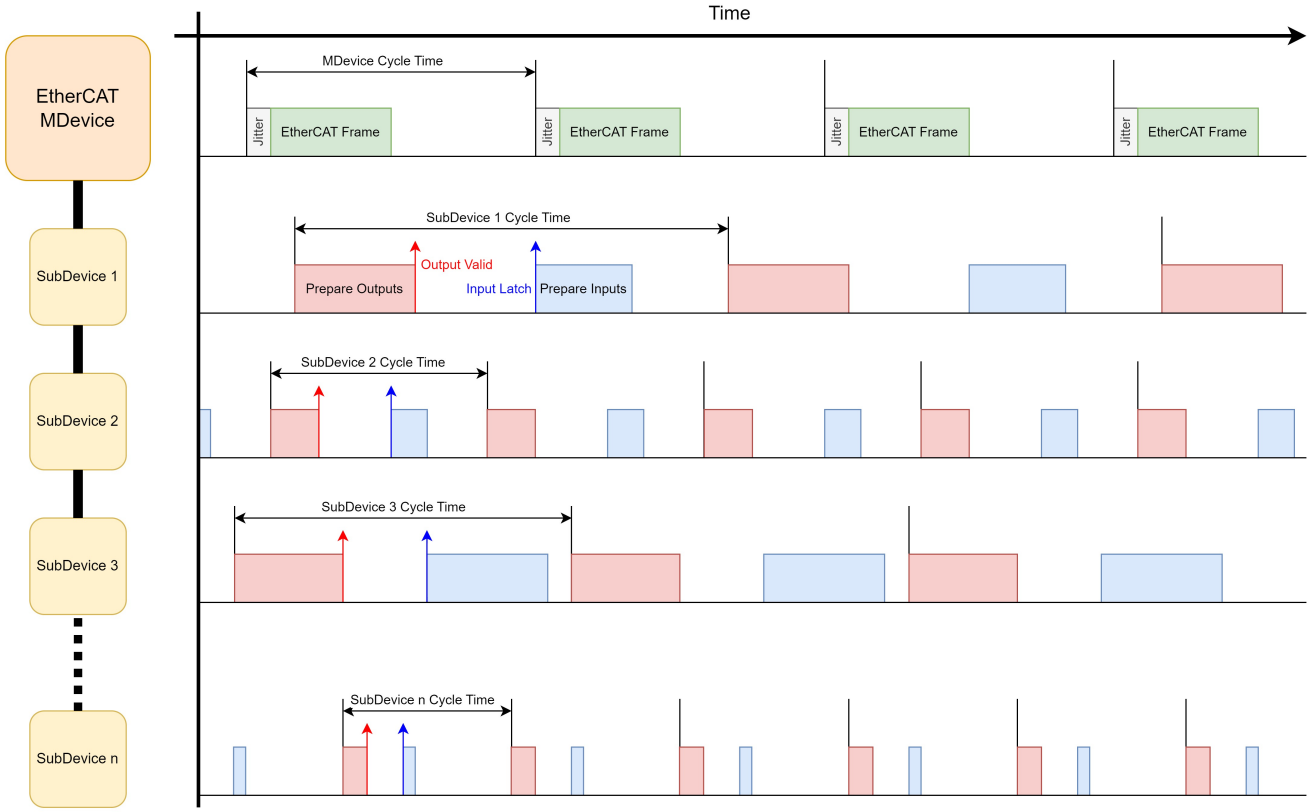


ESC 擁有三個同步輸出腳位：***IRQ***、***SYNC0*** 和 ***SYNC1***。當 ESC 收到 EtherCAT 週期性封包後，IRQ 腳位會向上層的微控制器（µC）發出訊號。SYNC0 和 SYNC1 腳位則根據 ESC 中與 DC 有關的暫存器設定，週期性地向微控制器輸出訊號。因此，若某個 EtherCAT 從站裝置未搭載微控制器（µC），則不支援同步功能。EtherCAT 支援三種同步模式：

- Free Run
- SM-Synchronous
- DC-Synchronous

1.5.1 Free Run 模式

EtherCAT 主站與所有 EtherCAT 從站裝置各自擁有獨立的本地計時器，而且它們的週期時間是獨立的，因此它們不同步。如下圖所示，EtherCAT 主站以及從站 1、從站 2、從站 3 至 從站 n 均有各自的週期時間，導致 **輸出有效時間 (Output Valid)** 與 **輸入鎖存時間 (Input Latch)** 不一致。此種情況不適用於對同步性要求較高的應用。

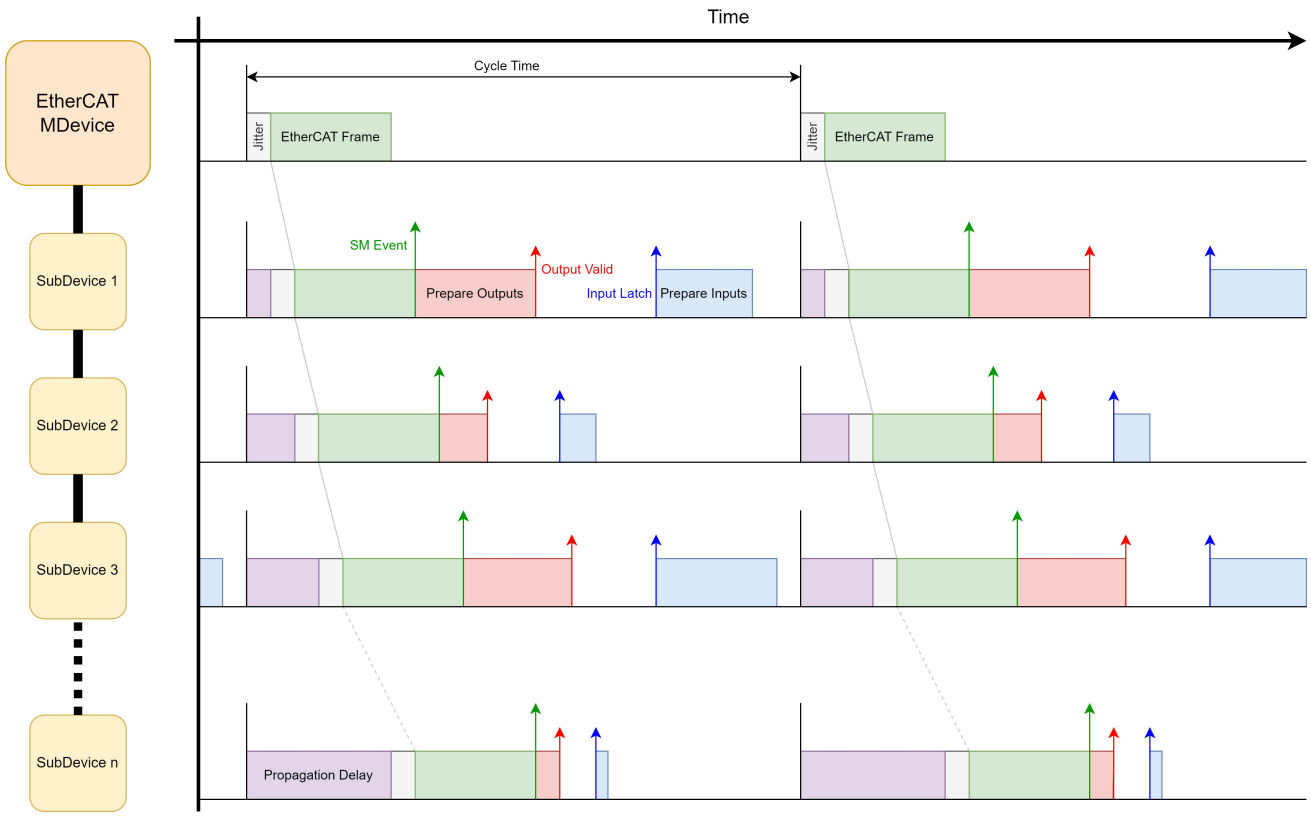


1.5.2 SM-Synchronous 模式

當 ESC 接收到週期性封包時，IRQ 腳位會產生一個 SM 事件 (SM Event)，此事件將觸發微控制器（μC）中本地應用程式的執行。如下圖所示，子裝置接收到的週期性封包會帶有與主站傳送時相同的抖動。即便假設抖動為零，由於**硬體傳播延遲 (Propagation Delay)** 的存在，最後面的從站裝置仍會比前面的從站裝置晚收到封包。

由於傳播延遲，從站裝置之間 SM 事件的時間會出現偏移，導致 SM-Synchronous 模式的同步精度僅能達到**微秒級**。

若每個從站裝置都支援 **SyncManager 參數物件** 中的 **Shift Time 設定 (0x1C32.3/0x1C33.3)**，則可嘗試將所有從站裝置的 **輸出有效時間 (Output Valid)** 與 **輸入鎖存時間 (Input Latch)** 對齊。然而，由於無法準確計算每個從站裝置的實際傳播延遲，這種調整在實務上具有相當的挑戰性。



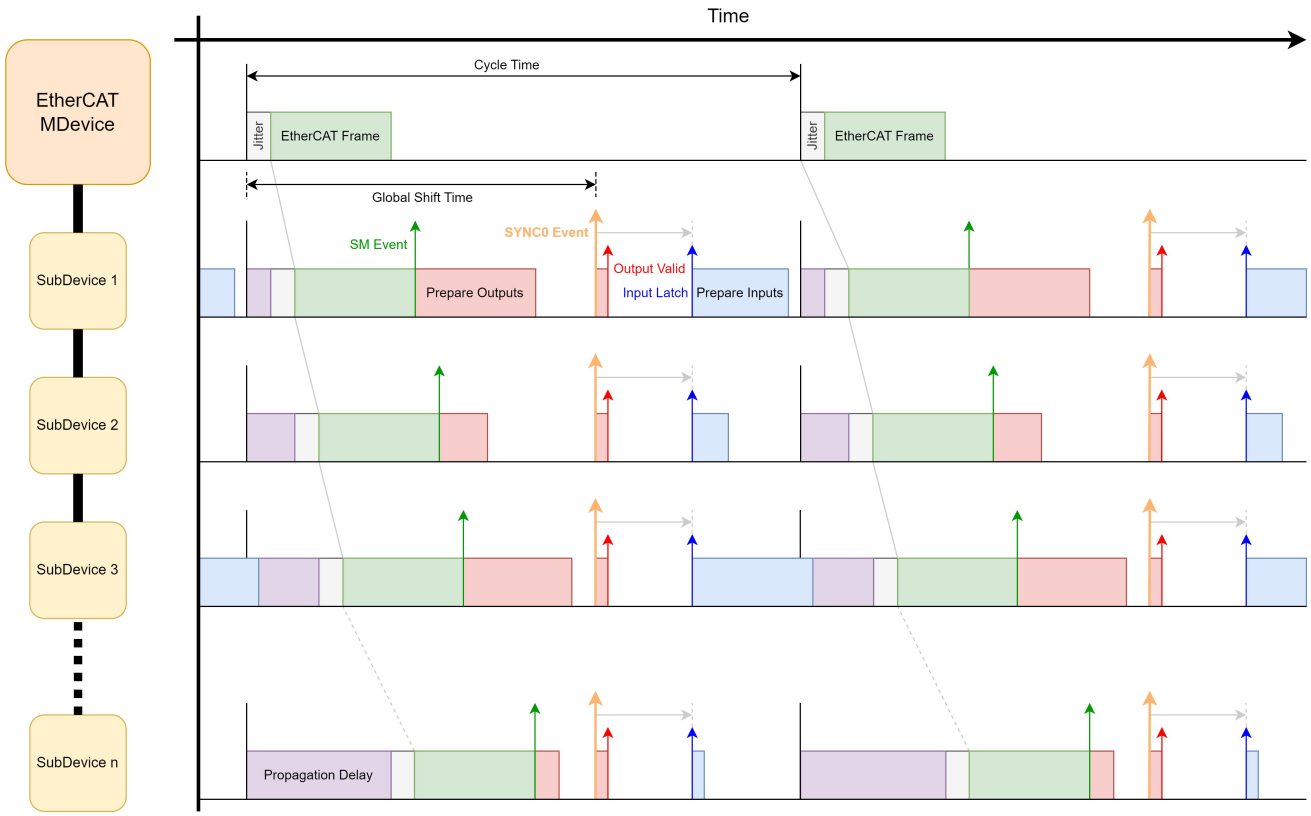
1.5.3 DC-Synchronous 模式

SYNC0 或 SYNC1 腳位會根據 ESC 中與 DC 相關暫存器的設定，週期性地產生 **SYNC 事件**，此事件會觸發微控制器中本地應用程式的執行。如下圖所示，雖然抖動與傳播延遲依然存在，且 SM 事件仍在接收到週期性封包後才被觸發，但在此 **DC-Synchronous** 模式下，SYNC0 事件是以週期方式產生，**不會受到抖動與傳播延遲的影響**。

由於 SYNC0 事件是由 **DC 單元**所觸發，且所有從站裝置之間的 SYNC0 事件幾乎**沒有偏移時間差**，再加上系統會週期性地傳送 **APRW/FPRW 命令**以同步所有從站裝置的系統時間，因此其**同步精度可達奈秒等級 (nanosecond)**。
若子裝置支援 SyncManager 參數物件 (0x1C32.3/0x1C33.3) 中的 **Shift Time** 設定，即可嘗試將各子裝置的 **輸出有效時間 (Output Valid)** 與 **輸入鎖存時間 (Input Latch)** 的時序調整為同一時間點。

然而，圖中的 **Global Shift Time (全域偏移時間)** 的選擇至關重要，並且必須符合以下條件：

- 在所有子裝置皆完成接收週期性封包之後
- 在下一週期的週期性封包傳送之前
- 根據不同的 DC-Synchronous 方法，可能需要在執行 **Prepare Outputs** 之後選擇：
    - 當 SM 事件發生時觸發 µC 執行 Prepare Outputs
    - 當 SYNC 事件發生時觸發 µC 執行 Output Valid

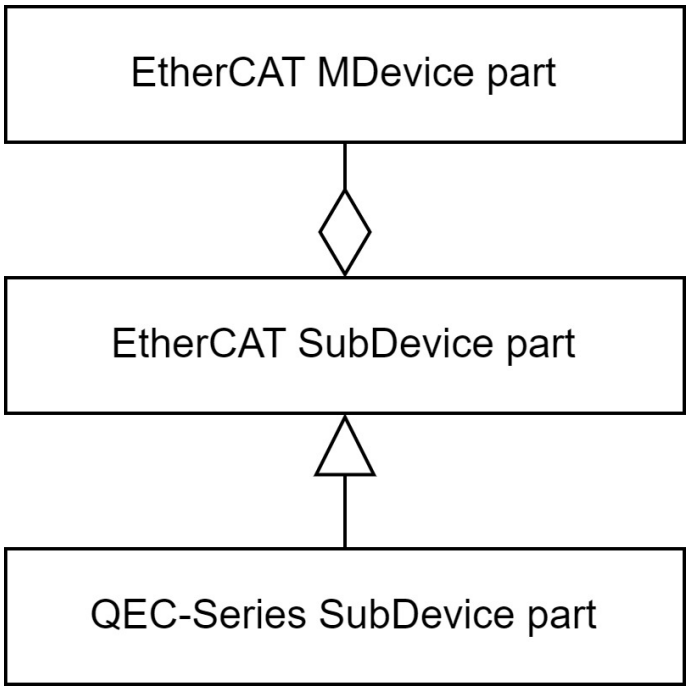


正確的 **Global Shift Time** 不是唯一的，而是可以在整個週期時間內的區間中進行選擇。。要了解有關各種 DC-Synchronous 方法的更多信息，請參閱 **ETG.1020 EtherCAT Protocol Enhancements**。

章節. 2

功能函式

EtherCAT 是一種廣泛應用於自動化控制系統的即時工業乙太網路通訊協定。 QEC 主站是一個以 C/C++ 實作的 EtherCAT 主站函式庫，其中包括主站類別、通用從站裝置類別、CiA 402 從站裝置類別以及 QEC 系列從站裝置的專用類別。這些類別不僅具有明確定義的職責，而且還考慮了未來的可擴展性。



這些類別可分為三個部分：

- EtherCAT MDevice
  EtherCAT MDevice 部分提供多樣且彈性的主站組態與操作功能，並為 EtherCAT SubDevice 部分提供各種 EtherCAT 從站裝置操作函式的呼叫介面。

- EtherCAT SubDevice
  EtherCAT SubDevice 部分提供通用的 EtherCAT 從站裝置類別，可操作 PDO、CoE、FoE 等功能，並包含 CiA 402 通用從站裝置類別。

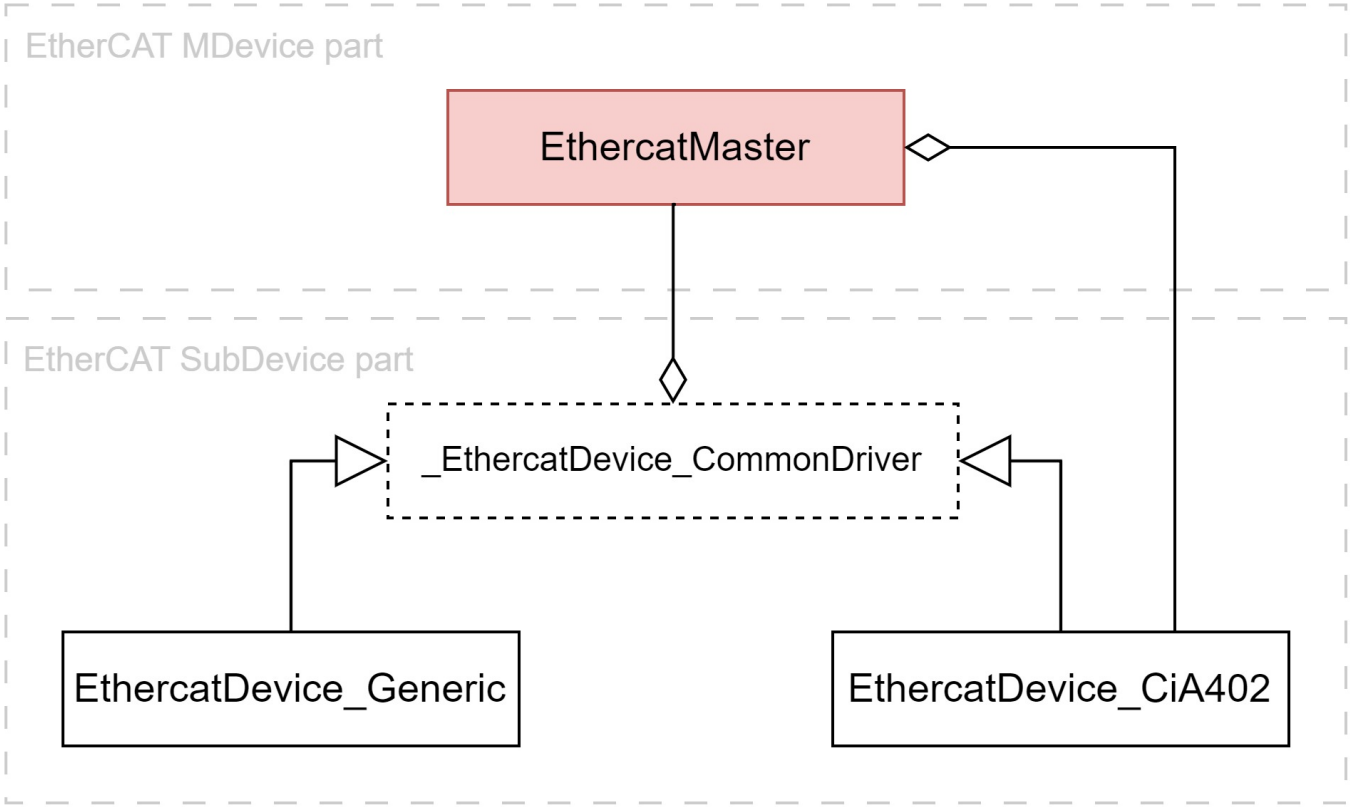
- QEC-Series SubDevice
  QEC-Series SubDevice 部分則提供專為 ICOP QEC 系列從站裝置設計的功能，讓使用者能以更直觀、簡潔的方式進行程式撰寫。

2.1 EtherCAT MDevice

EtherCAT MDevice 部分不僅提供了豐富且靈活的主站配置和操作函式，還提供了豐富的 EtherCAT 從站操作函式，供 EtherCAT SubDevice 部分呼叫。

EthercatMaster 是 EtherCAT MDevice 部分中唯一的類別，它是與 EtherCAT 韌體的關鍵通訊橋樑。在雙機通訊方面，其職責包括通訊介面初始化、循環進行過程資料交換、處理非循環傳輸介面、管理中斷事件等。在 API 方面，EthercatMaster 類別提供與主站初始化、主站控制、存取從站資訊相關的函數。

EtherCAT MDevice 與 EtherCAT SubDevice 之間的主要類別關係為關聯關係 (Association)，即 EtherCAT SubDevice 端是依賴 EtherCAT MDevice 端。EthercatMaster 的類別關係如下圖所示：



- EthercatMaster 與 _EthercatDevice_CommonDriver 之間為關聯關係，_EthercatDevice_CommonDriver 依賴 EthercatMaster。
- EthercatMaster 與 EthercatDevice_CiA402 之間為關聯關係，EthercatMaster 依賴 EthercatDevice_CiA402。

函式清單：

函式名稱	說明	Callback 中呼叫
初始化相關函式		
begin()	初始化 EtherCAT 主站	
end()	解除初始化 EtherCAT 主站	
isRedundancy()	檢查 EtherCAT 主站是否啟用了線路冗餘	O
libraryVersion()	取得 EtherCAT 主站函式庫版本	O
firmwareVersion()	取得 EtherCAT 韌體版本	O
readSettings()	讀取目前的 EtherCAT 主站設定	
saveSettings()	儲存 EtherCAT 主站設定	
控制相關函式		
start()	啟動 EtherCAT 主站	
stop()	停止 EtherCAT 主站	
update()	更新處理資料並處理非週期性命令	O
setShiftTime()	設定 DC 同步模式的 Global Shift Time	
getShiftTime()	取得 DC 同步模式的 Global Shift Time	O
getSystemTime()	取得當前週期的系統時間	O
getWorkingCounter()	取得當前週期的 Working Counter 值	O
getExpectedWorkingCounter()	取得預期的 Working Counter 值	O
Callback 相關函式		
attachCyclicCallback()	註冊週期性 Callback 函式	
detachCyclicCallback()	解除註冊週期性 Callback 函式	
attachErrorCallback()	註冊錯誤 Callback 函式	
detachErrorCallback()	解除錯誤 Callback 函式	
attachEventCallback()	註冊事件 Callback 函式	
detachEventCallback()	解除事件 Callback 函式	
errGetCableBrokenLocation1()	在錯誤 Callback 中取得斷線位置 1	O[1]
errGetCableBrokenLocation2()	在錯誤 Callback 中取得斷線位置 2	O[1]
evtGetMasterState()	在事件 Callback 中取得 EtherCAT 主站狀態	O[2]
從站裝置資訊相關函式		
getSlaveCount()	取得網路上從站裝置的數量	O
getVendorID()	取得指定裝置的廠商 ID	O
getProductCode()	取得指定裝置的產品代碼	O
getRevisionNumber()	取得指定裝置的版本編號	O
getSerialNumber()	取得指定裝置的序號	O
getAliasAddress()	取得指定裝置的別名位址	O
getSlaveNo()	查找符合條件的 EtherCAT 從站裝置序號	O

- **Note 1**：此函式僅可在錯誤回呼中呼叫。
- **Note 2**：此函式僅可在事件回呼中呼叫。

函式分類：

- 初始化
- 控制
- Callback
- 從站裝置資訊

EtherCAT 主站設定

本函式庫提供多項可供使用者選擇的設定參數，旨在滿足不同應用場景的需求。以下列出本函式庫所提供的設定參數。

```
typedef struct {

    EthercatDcSyncMode DcSyncMode;
    uint32_t StaticDriftCompensationFrames;

    uint32_t StateMachineTimeoutI2P;
    uint32_t StateMachineTimeoutP2S;
    uint32_t StateMachineTimeoutS2O;
    uint32_t ScanNetworkTimeout;
    uint32_t StartMasterTimeout;
    uint32_t StartDeviceTimeout;

    uint32_t ErrorDetectWkcMultipleFaultsThreshold;
    uint32_t ErrorDetectMultipleLostFramesThreshold;
    uint32_t EnableErrorBusReactionSyncUnitToSafeOp:1,
             EnableErrorBusReactionSyncUnitToSafeOpAutoRestart:1,
             IgnoreBiosOverride:1;

} EthercatMasterSettings;
```

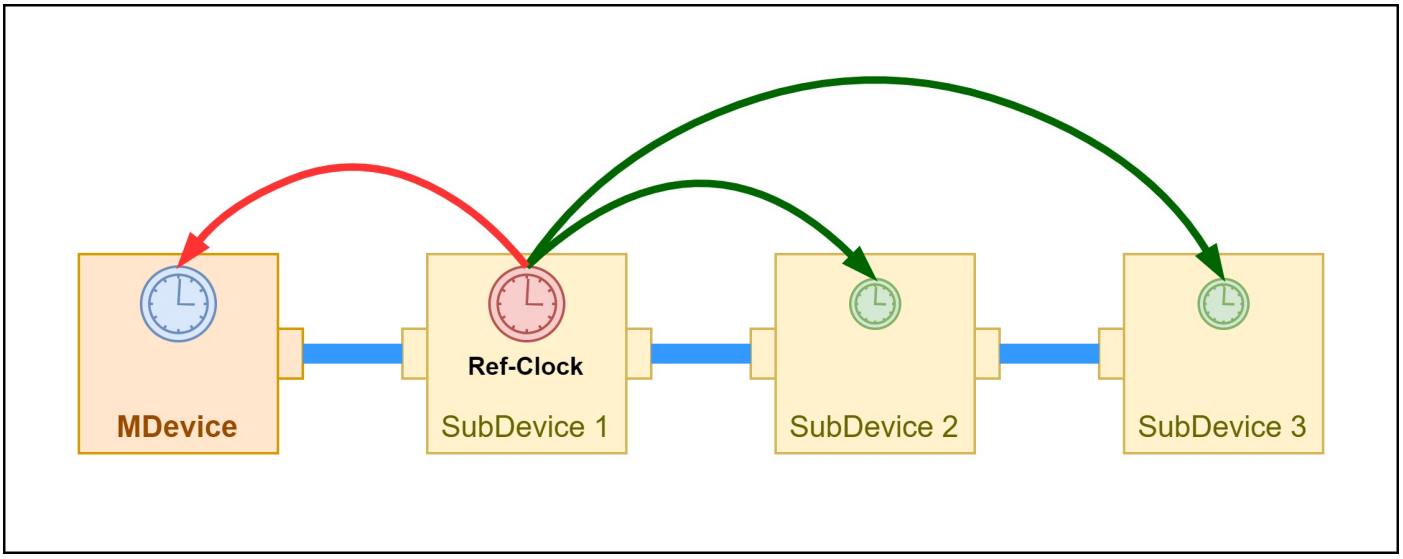
DcSyncMode

預設值：`ECAT_MASTER_SHIFT`

在 DC-Synchronous 模式下，第一個具備 DC 功能的從站裝置會作為系統參考時鐘，以同步網路上其他具備 DC 的從站裝置。然而，這僅限於同步從站裝置之間的系統時間，不包含 EtherCAT 主站。在啟用 DC-Synchronous 模式的應用時，主站通常需要精確且週期性地控制從站裝置，因此主站本身也必須與網路上的所有從站裝置進行系統時間同步。

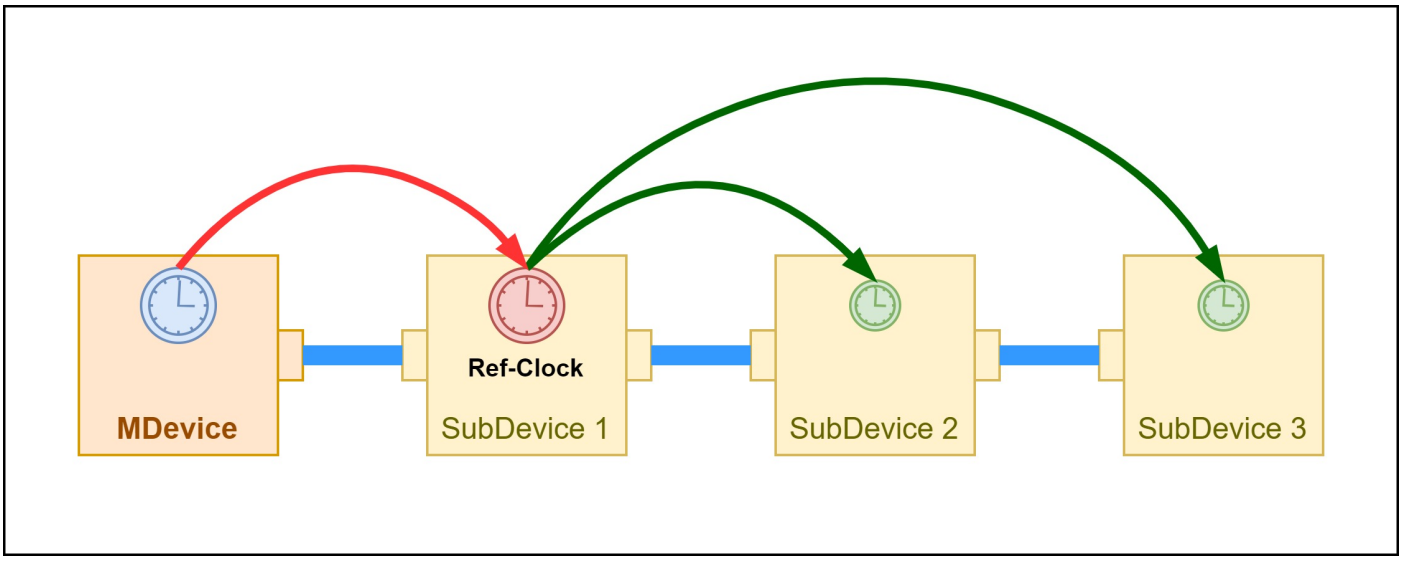
此同步有兩種方法：

- **主站偏移模式 (MDevice Shift Mode)** - `ECAT_MASTER_SHIFT`



  - 主站的系統時間同步到參考時鐘。
  - 所有具 DC 功能的從站裝置同步到參考時鐘。

- **總線偏移模式 (Bus Shift Mode)** - `ECAT_BUS_SHIFT`



  - 參考時鐘同步到主站的系統時間。
  - 所有具 DC 功能的從站裝置同步到參考時鐘

StaticDriftCompensationFrames

預設值：30000，單位：frames
EtherCAT 主站會傳送多個獨立的 ARMW 或 FRMW 漂移補償封包，以將參考時鐘的系統時間分發給所有具 DC 功能的從站裝置。

StateMachineTimeoutI2P

預設值：3000，單位：毫秒
狀態機從 Init 狀態轉換至 Pre-Operational 狀態的逾時時間。

StateMachineTimeoutP2S

預設值：10000，單位：毫秒
狀態機從 Pre-Operational 狀態轉換至 Safe-Operational 狀態的逾時時間。

StateMachineTimeoutS2O

預設值：10000，單位：毫秒
狀態機從 Safe-Operational 狀態轉換至 Operational 狀態的逾時時間。

ScanNetworkTimeout

預設值：5000，單位：毫秒
網路掃描逾時時間。此操作會在 `EthercatMaster::begin()` 中執行。

StartMasterTimeout

預設值：3000，單位：毫秒
啟動主站的基礎逾時時間，記作 T_{base}。
在 `EthercatMaster::start()` 中，韌體會被要求啟動 EtherCAT，此請求的逾時時間稱為啟動逾時時間 $T_{startup}$。 EtherCAT 的啟動逾時時間計算公式如下：
$T_{startup} = T_{base} + (T_{SubDevice} \times N_{SubDevices})$
其中， $N_{SubDevices}$ 是網路上的從站裝置數量。

StartDeviceTimeout

預設值：500，單位：毫秒
每個從站裝置在啟動主站時的個別逾時時間，記作 $T_{SubDevice}$.

ErrorDetectWkcMultipleFaultsThreshold

預設值：3
主站會檢查接收到的 EtherCAT 資料封包中的工作計數器。若該值與預期不符，則會視為錯誤。當連續錯誤次數超過此參數設定值時，將觸發 *ECAT_ERR_WKC_MULTIPLE_FAULTS* 錯誤中斷。

ErrorDetectMultipleLostFramesThreshold

預設值：3
主站可透過 EtherCAT 封包標頭中的索引來檢查是否所有傳送出的封包皆有接收回來。若封包遺失，則視為錯誤。當連續遺失封包的次數超過此參數設定值時，將觸發 *ECAT_ERR_MULTIPLE_LOST_FRAMES* 錯誤中斷。

EnableErrorBusReactionSyncUnitToSafeOp

預設值：0
若此參數設為 1，主站將會針對具備應用控制器的子裝置改變其 EtherCAT 狀態，並針對僅支援 EtherCAT 狀態機模擬的子裝置停用其 Sync Manager 通道。

EnableErrorBusReactionSyncUnitToSafeOpAutoRestart

預設值：1
此參數僅在 *EnableErrorBusReactionSyncUnitToSafeOp* 設為 1 時才生效。若此參數設為 1，主站將會依據 ETG.1020 EtherCAT Protocol Enhancements 中定義的 Sync Unit 自動重啟行為，自動嘗試重新啟動 Sync Unit，並將 EtherCAT 狀態機切換回 Operational 狀態。

IgnoreBiosOverride

預設值：0
QEC 主站在 BIOS 中儲存了一些 EtherCAT 組態參數。若此參數設為 1，表示忽略 BIOS 中的 EtherCAT 組態參數；若為 0，則不忽略，會使用 BIOS 設定。

2.1.1 初始化函式

在啟動 EtherCAT 主站之前，必須先進行初始化。本函式庫提供多項可選擇的組態參數，以滿足使用者多元的應用需求。

函式：

- begin()
- end()
- isRedundancy()
- libraryVersion()
- firmwareVersion()
- readSettings()
- saveSettings()

begin()

說明

初始化 EtherCAT 主站，掃描網路上所有 EtherCAT 從站裝置，並將 EtherCAT 狀態機切換至 Pre-Operational 狀態。

語法

```
int begin(EthernetPort eth = ECAT_ETH_0, const char *eni_filename = NULL);
```

參數

- *[in] EthernetPort eth*
  指定用於 EtherCAT 通訊的乙太網路介面。可選項如下：
  - **ECAT_ETH_0**：僅使用 eth0 作為 EtherCAT 通訊介面（預設）
  - **ECAT_ETH_1**：僅使用 eth1 作為 EtherCAT 通訊介面
  - **ECAT_ETH_REDUNDANCY**：啟用線路備援，eth0 為主通訊埠，eth1 為備援通訊埠
- *[in] const char *eni_filename*
  EtherCAT Network Information (ENI) 檔案名稱。關於本函式庫支援的 ENI 檔案內容，請參閱 EtherCAT Network Information 章節。預設值為 NULL。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式為阻塞式函式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"
EthercatMaster master;

void setup() {
    master.begin();
    master.start(1000000); // cycle time set as 1 millisecond.
}
void loop() {

}
```

end()

說明

解除初始化 EtherCAT 主站裝置。

語法

```
void end();
```

參數

無。

傳回值

無。

備註

此函式必須在呼叫 `EthercatMaster::begin()` 之後且在 `EthercatMaster::start()` 之前執行。此函式為阻塞式函式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"
EthercatMaster master;

void setup() {
    master.begin();
    master.end();
}
void loop() {

}
```

isRedundancy()

說明

檢查 EtherCAT 主站是否已啟用纜線冗餘功能。

語法

```
bool isRedundancy();
```

參數

無。

傳回值

回傳是否已啟用 EtherCAT 主站的纜線冗餘功能。若回傳 true，表示已啟用；回傳 false，則表示未啟用。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，且可在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"
EthercatMaster master;

void setup() {
  Serial.begin(115200);
  master.begin();
  Serial.println(master.isRedundancy());
}
void loop() {

}
```

libraryVersion()

說明

取得 EtherCAT 主站函式庫的版本資訊。

語法

```
char *libraryVersion();
```

參數

無。

傳回值

回傳指向 EtherCAT 主站函式庫版本字串的指標。

備註

此函式為非阻塞式，且可在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"
EthercatMaster master;

void setup() {
  Serial.begin(115200);
  Serial.println(master.libraryVersion());
}
void loop() {

}
```

firmwareVersion()

說明

取得 EtherCAT 韌體版本資訊。

語法

```
char *firmwareVersion();
```

參數

無。

傳回值

回傳指向 EtherCAT 韌體版本字串的指標。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，且可在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"
EthercatMaster master;

void setup() {
  Serial.begin(115200);
  master.begin();
  Serial.println(master.firmwareVersion());
}
void loop() {

}
```

readSettings()

說明

讀取目前 EtherCAT 主站的設定值。

語法

```
int readSettings(EthercatMasterSettings *settings);
```

參數

- *`[in] EthercatMasterSettings *settings`*
  指向 EthercatMasterSettings 資料結構的指標。關於 EthercatMasterSettings 參數的詳細說明，請參閱 EtherCAT 主站設定。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在 `EthercatMaster::begin()` 之前或 `EthercatMaster::end()` 之後呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup(void) {
  EthercatMasterSettings settings;

  Serial.begin(115200);
  while (!Serial);

  master.readSettings(&settings);
  settings.DcSyncMode = ECAT_BUS_SHIFT;
  settings.StateMachineTimeoutI2P = 3000;
  settings.StateMachineTimeoutP2S = 10000;
  settings.StateMachineTimeoutS2O = 1000;
  settings.ScanNetworkTimeout = 5000;
  settings.StartMasterTimeout = 3000;
  settings.StartDeviceTimeout = 2000;
  master.saveSettings(&settings);

  Serial.println(master.begin());
  slave.attach(0, master);

  Serial.println(master.start(1000000, ECAT_SYNC));
```

```
}

void loop() {

}
```

saveSettings()

說明

儲存 EtherCAT 主站的設定值。

語法

`int saveSettings(EthercatMasterSettings *settings);`

參數

- *`[in] EthercatMasterSettings *settings`*

  指向 EthercatMasterSettings 資料結構的指標。關於 EthercatMasterSettings 參數的詳細說明，請參閱 EtherCAT 主站設定。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在 `EthercatMaster::begin()` 之前或 `EthercatMaster::end()` 之後呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup(void) {
  EthercatMasterSettings settings;

  Serial.begin(115200);
  while (!Serial);

  master.readSettings(&settings);
  settings.DcSyncMode = ECAT_BUS_SHIFT;
  settings.StateMachineTimeoutI2P = 3000;
  settings.StateMachineTimeoutP2S = 10000;
  settings.StateMachineTimeoutS2O = 1000;
  settings.ScanNetworkTimeout = 5000;
  settings.StartMasterTimeout = 3000;
  settings.StartDeviceTimeout = 2000;
  master.saveSettings(&settings);

  Serial.println(master.begin());
  slave.attach(0, master);
```

```
  Serial.println(master.start(1000000, ECAT_SYNC));
}

void loop() {

}
```

2.1.2 控制函式

EtherCAT 主站函式庫所提供的控制函式，對於管理 EtherCAT 網路的狀態與運作至關重要。透過這些函式，使用者可以對網路進行精確控制，確保主站與從站裝置之間達成穩定且同步的通訊。

函式：

- start()
- stop()
- update()
- setShiftTime()
- getShiftTime()
- getSystemTime()
- getWorkingCounter()
- getExpectedWorkingCounter()

start()

說明

啟動 EtherCAT 主站，配置所有從站的 SM、FMMU 和 DC 暫存器，並將 EtherCAT 狀態機切換至 Operational 狀態。

語法

```
int start(uint64_t cycletime_ns = 0, EthercatMasterMode mode = ECAT_FREERUN);
```

參數

- *[in] uint64_t cycletime_ns*
  EtherCAT 週期時間。EtherCAT 韌體將根據此週期時間週期性地產生中斷，以更新製程資料與處理非週期性傳輸。若使用者註冊了 Cyclic Callback，且 EtherCAT 主站控制模式未設定為 ECAT_FREERUN_MANUAL，該 callback 將於這些中斷中被呼叫。
- *[in] EthercatMasterMode mode*
  選擇 EtherCAT 控制模式。有關每種模式的詳細說明，請參閱以下範例。
  a. ***ECAT_SYNC***
  b. ***ECAT_FREERUN***
  c. ***ECAT_FREERUN_MANUAL***

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

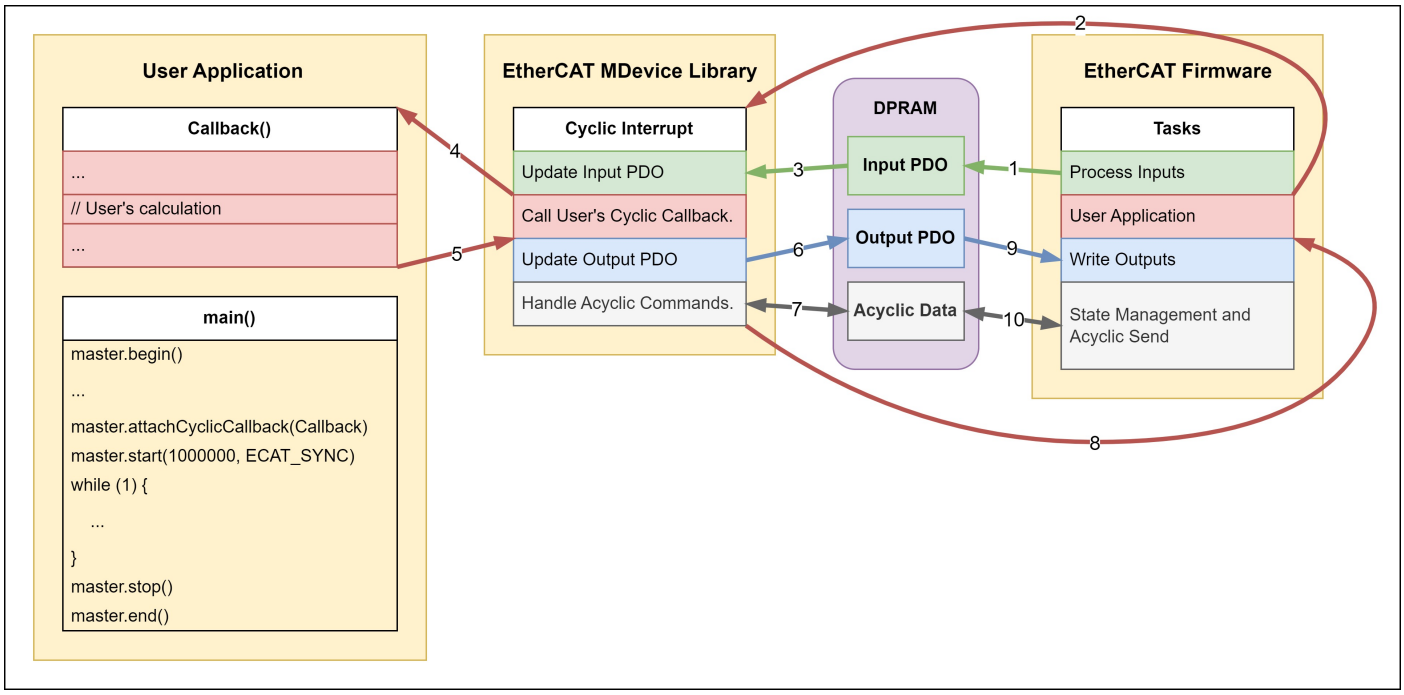
備註

此函式為阻塞式函式，不能在 callback 函式中呼叫。

範例

a. ECAT_SYNC
b. ECAT_FREERUN
c. ECAT_FREERUN_MANUAL

a. ECAT_SYNC

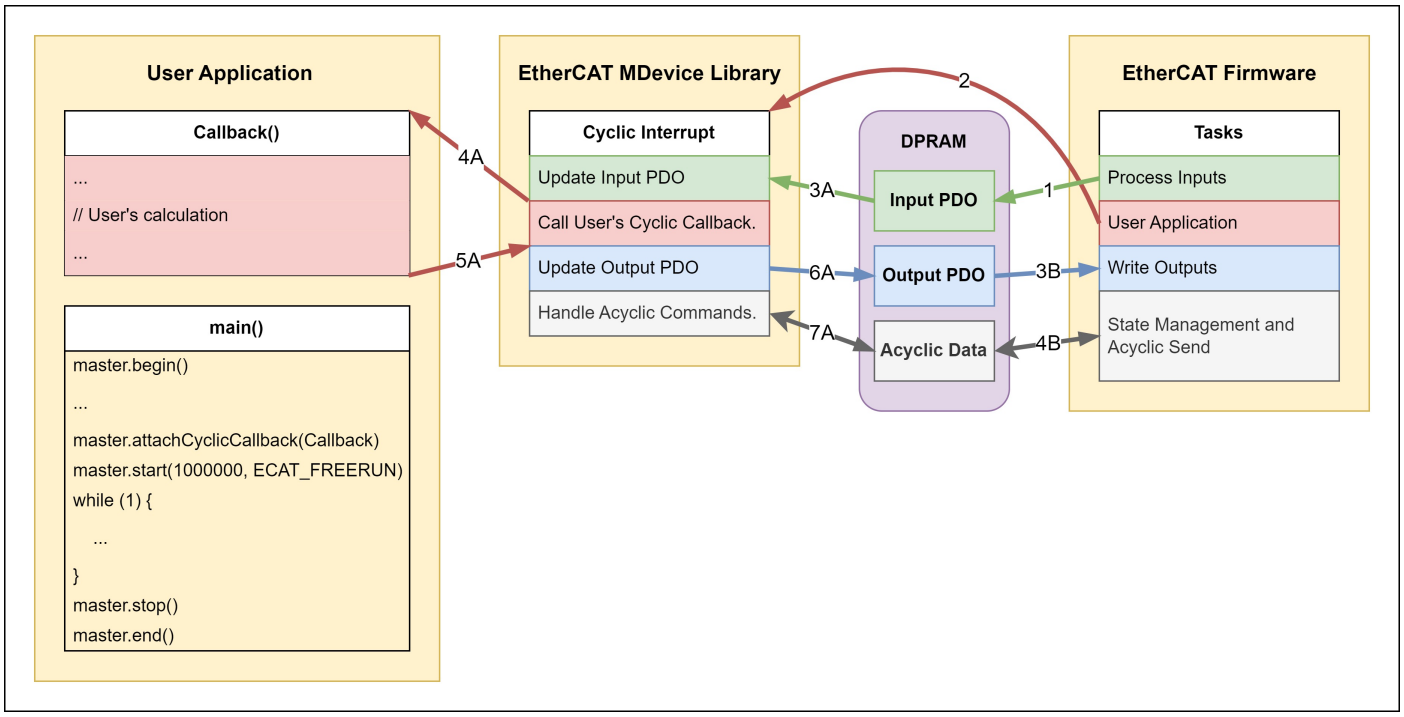


此模式提供最高等級的雙系統同步。如圖所示，編號箭頭代表操作順序，流程中無分支。當 EtherCAT 韌體對 EtherCAT 主站函式庫觸發週期性中斷 (步驟 2) 後，會等待 EtherCAT 主站函式庫回應 ACK (步驟 8)後，才進行下一個動作。

若使用者已註冊 cyclic callback，則在週期性中斷時會呼叫該 callback，如步驟 4 所示。只要使用者在 cyclic callback 中讀取目前的輸入製程資料，進行處理、計算輸出製程資料並寫回，當前週期就會將輸出製程資料送出至 EtherCAT 網路，從而滿足即時控制系統的需求。

```
#include "Ethercat.h"

EthercatMaster master;

void CyclicCallback() {
  // ...
}
void setup() {
  master.begin();
  master.attachCyclicCallback(CyclicCallback);
  master.start(1000000, ECAT_SYNC);
}
void loop() {
  // ...
}
```

b. ECAT_FREERUN

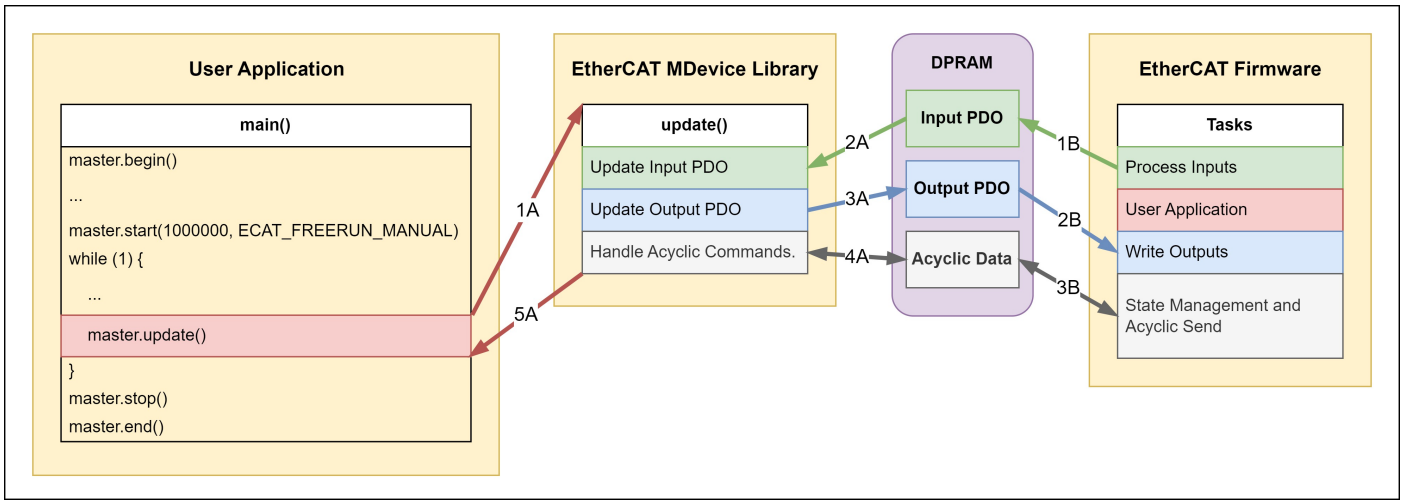


此為無雙系統同步的自由執行模式。如圖所示，編號箭頭表示操作順序。但在步驟 3 發生了分歧，原因是在 EtherCAT 韌體觸發對 EtherCAT 主站函式庫的週期性中斷 (步驟 2) 後，並不會等待主站函式庫回應，而是直接進行下一個動作。此模式下兩個系統彼此獨立運作，沒有同步機制。

若使用者已註冊 cyclic callback，則在週期性中斷時仍會呼叫該 callback，如步驟 4A 所示。

```
#include "Ethercat.h"

EthercatMaster master;

void CyclicCallback() {
  // ...
}

void setup() {
  master.begin();
  master.attachCyclicCallback(CyclicCallback);
  master.start(1000000, ECAT_FREERUN);
}

void loop() {
  // ...
}
```

c. ECAT_FREERUN_MANUAL



此模式同樣為無雙系統同步的自由執行模式。與 ECAT_FREERUN 模式的主要差異在於：本模式中不會有週期性中斷來更新 Process Data 或處理非週期命令，而是必須由使用者手動呼叫 `EthercatMaster::update()` 來完成上述工作。

此外，由於此模式下沒有週期性中斷，因此也不會呼叫 cyclic callback。正如圖中編號箭頭所示，兩個系統是獨立運作的，彼此之間無同步機制。

```
#include "Ethercat.h"

EthercatMaster master;

void setup() {
  master.begin();
  master.start(1000000, ECAT_FREERUN_MANUAL);
}
void loop() {
  // ...
  master.update();
}
```

stop()

說明

停止 EtherCAT 主站，並將 EtherCAT 狀態機切換至 Pre-Operational 狀態。

語法

```
int stop();
```

參數

無。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::start()` 之後呼叫。此函式為阻塞式函式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;

void setup() {
    master.begin();
    master.start();

    master.stop();
    master.end();
}
void loop() {

}
```

update()

說明

更新處理資料與非週期命令。此函式僅能在 ECAT_FREERUN_MANUAL 控制模式下運作。

語法

```
void update();
```

參數

無。

傳回值

無。

備註

此函式必須在成功執行 `EthercatMaster::start()` 且尚未呼叫 `EthercatMaster::stop()` 之前使用。此函式為非阻塞式，且可在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;

void setup() {
  master.begin();
  master.start(1000000, ECAT_FREERUN_MANUAL);
}

void loop() {
  // ...
  master.update();
}
```

setShiftTime()

說明

設定 DC 同步模式下的全域偏移時間 (Global Shift Time)。如果未設定或設定為 INT32_MAX，EtherCAT 韌體將自動計算適當的值。關於全域偏移時間的定義，請參考同步。

語法

```
int setShiftTime(int32_t nanoseconds);
```

參數

- *[in] int32_t nanoseconds*
  欲設定的全域偏移時間，單位為奈秒。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 且尚未執行 `EthercatMaster::start()`，或在成功執行 `EthercatMaster::stop()` 且尚未執行 `EthercatMaster::end()` 的情況下呼叫。此函式為非阻塞式，可以在 callback 函式中呼叫，但**不建議**這樣使用。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
    master.begin();
    slave.attach(0, master);
    slave.setDc(1000000);
    master.setShiftTime(250000); // 250000 ns
    master.start(1000000);
}
void loop() {
  // ...
}
```

getShiftTime()

說明

取得 DC 同步模式下的全域偏移時間 (Global Shift Time)。關於全域偏移時間的定義，請參考同步。

語法

```
int getShiftTime();
```

參數

無。

傳回值

回傳全域偏移時間，單位為奈秒。

備註

此函式必須在成功執行 `EthercatMaster::start()` 且尚未執行 `EthercatMaster::stop()` 的情況下呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  slave.setDc(1000000);
  master.start(1000000);

  Serial.print("Global Shift Time: "); Serial.println(master.getShiftTime());
}

void loop() {
  // ...
}
```

getSystemTime()

說明

取得目前週期的系統時間。建議在 cyclic callback 中使用，以確保所取得的系統時間屬於正確的控制週期。

語法

```
uint64_t getSystemTime();
```

參數

無。

傳回值

回傳目前週期的系統時間。

備註

此函式必須在成功執行 `EthercatMaster::start()` 且尚未執行 `EthercatMaster::stop()` 的情況下呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
uint64_t CurrentSystemTime;

void CyclicCallback() {
  CurrentSystemTime = master.getSystemTime();
}

void setup() {
  Serial.begin(115200);

  master.begin();
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  Serial.print("Current System Time: ");
  Serial.print((uint32_t)(CurrentSystemTime >> 32));
  Serial.print(" ");
```

```
  Serial.println((uint32_t)(CurrentSystemTime & 0xFFFFFFFF));

  delay(1000);
}
```

getWorkingCounter()

說明

取得目前週期的 Working Counter 值。建議在 cyclic callback 中使用，以確保所取得的值屬於正確的控制週期。

語法

```
int getWorkingCounter();
```

參數

無。

傳回值

回傳目前週期的 Working Counter 數值。

備註

此函式必須在成功執行 `EthercatMaster::start()` 且尚未執行 `EthercatMaster::stop()` 的情況下呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"
EthercatMaster master;
int CurrentWorkingCounter;

void CyclicCallback() {
  CurrentWorkingCounter = master.getWorkingCounter();
}

void setup() {
  Serial.begin(115200);
  master.begin();
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  Serial.print("Current Working Counter: ");
  Serial.println(CurrentWorkingCounter);
  delay(1000);
}
```

getExpectedWorkingCounter()

說明

取得預期的 Working Counter 數值。

語法

```
int getExpectedWorkingCounter();
```

參數

無。

傳回值

回傳預期的 Working Counter 數值。

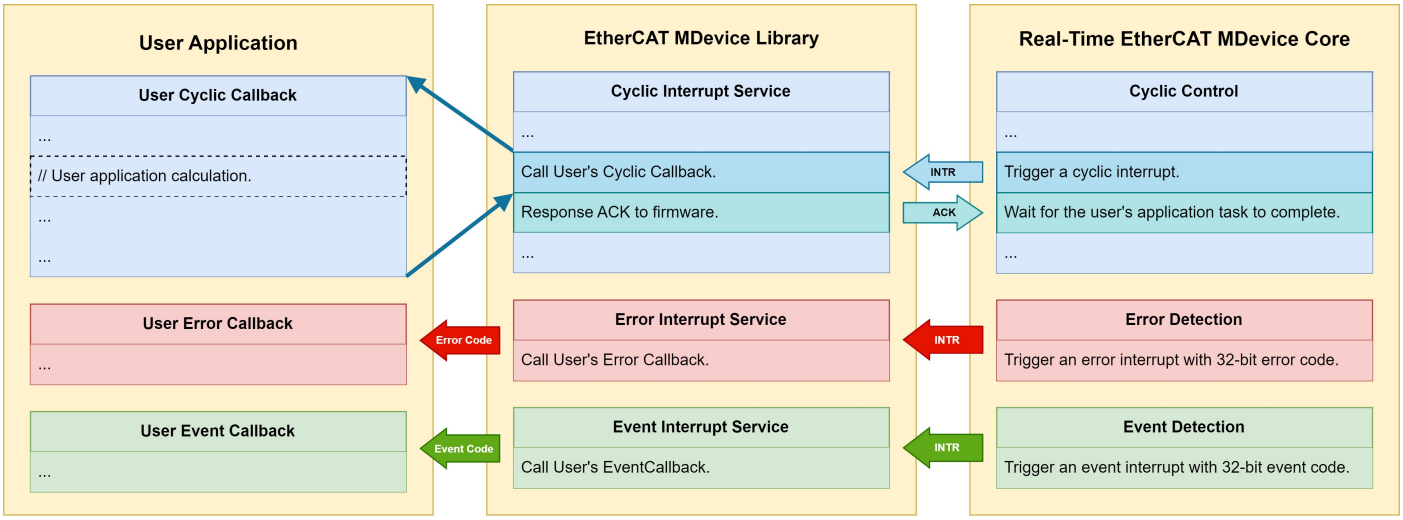
備註

此函式必須在成功執行 `EthercatMaster::start()` 且尚未執行 `EthercatMaster::stop()` 的情況下呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;

void setup() {
  Serial.begin(115200);
  master.begin();
  master.start();

  Serial.print("Expected Working Counter: ");
  Serial.println(master.getExpectedWorkingCounter());
  // ...
}

void loop() {
  // ...
}
```

2.1.3 Callback 函式



本函式庫提供三種類型的回呼函式如下：

- **週期性 Callback (Cyclic Callback)**
  週期性 Callback 的目的是讓使用者實作週期性控制系統，例如：運動控制、CNC 控制與機器人控制。即時的 EtherCAT 主站核心會在指定的週期時間觸發一次週期性中斷至主站函式庫，並等待 ACK (確認訊號)，以確保處理資料的同步。若使用者已註冊週期性 Callback 函式，系統將呼叫該 Callback，以實現週期性控制。

- **錯誤 Callback (Error Callback)**
  當即時 EtherCAT 主站核心偵測到錯誤時，會觸發錯誤中斷，並傳送一個 32 位元錯誤代碼給主站函式庫。若使用者已註冊錯誤 Callback 函式，系統將呼叫該 Callback，通知使用者錯誤內容。

  支援的錯誤代碼如下：

定義	代碼	說明
ECAT_ERR_WKC_SINGLE_FAULT	2000001	偵測到單次 Working Counter 錯誤。
ECAT_ERR_WKC_MULTIPLE_FAULTS	2000002	多次 Working Counter 錯誤。
ECAT_ERR_SINGLE_LOST_FRAME	2000003	封包遺失。
ECAT_ERR_MULTIPLE_LOST_FRAMES	2000004	多次封包遺失。
ECAT_ERR_CABLE_BROKEN	2000007	網路線斷線。
ECAT_ERR_WAIT_ACK_TIMEOUT	2001000	等待週期性中斷 ACK 超時。

- **事件 Callback (Event Callback)**
  當即時 EtherCAT 主站核心偵測到事件時，會觸發事件中斷，並傳送一個 32 位元事件代碼給主站函式庫。若使用者已註冊事件 Callback 函式，系統將呼叫該 Callback，通知使用者相關事件。

  支援的事件代碼如下：

定義	代碼	說明
ECAT_EVT_STATE_CHANGED	1000001	主站的 EtherCAT 狀態已改變。
ECAT_EVT_CABLE_RECONNECTED	1000002	網路線已重新接上。

函式：

- attachCyclicCallback()
- detachCyclicCallback()
- attachErrorCallback()
- detachErrorCallback()
- attachEventCallback()
- detachEventCallback()
- errGetCableBrokenLocation1()
- errGetCableBrokenLocation2()
- evtGetMasterState()

限制事項

這些 callback 函式是由中斷處理程序 (Interrupt Handler) 所呼叫，因此執行時處於中斷上下文 (Interrupt Context)。

因此，必須遵守中斷上下文的相關限制：

- 避免呼叫**阻塞式**函式（Blocking Functions）
- 避免呼叫**不可重入**函式（Non-Reentrant Functions）
- 避免呼叫**耗時**的函式
- 避免呼叫**非中斷**安全函式（Non-Interrupt-Safe Functions），例如：
  - malloc() / free()
  - printf() / scanf()
  - fopen() / fclose() / fprintf() / fscanf() / fwrite() / fread()
  - ...
- 避免呼叫**第三方函式庫中的函式**，除非確定其不屬於上述類別

浮點運算 (Floating-Point Arithmetic)

FPU (浮點運算單元) 是一種專門設計用來執行浮點數運算的硬體元件，通常整合在電腦的中央處理器 (CPU) 中。FPU 暫存器是 FPU 中用來暫存資料與運算中間結果的儲存單元，使得 FPU 能夠快速進行複雜的浮點數運算。

當發生中斷時，CPU 會從使用者模式切換到核心模式。此時的 FPU 狀態可能與中斷處理常式所需的狀態不同。如果不儲存該狀態，中斷處理常式可能會不經意地修改使用者程式的 FPU 狀態，導致執行結果不可預期。因此，在中斷處理常式執行完畢後，必須還原原本的 FPU 狀態，以確保使用者程式可以正確繼續執行，而不會因 FPU 狀態改變而產生錯誤。

如果在中斷上下文 (Interrupt Context) 中執行的 callback 函式需要使用浮點數運算，則必須在進入該函式前儲存 FPU 狀態，並在離開後還原狀態，以避免影響處理序上下文 (Process Context) 中的浮點運算。

包含 FPU 狀態儲存與還原機制的 callback 函式稱為 FPU-enabled callback function，反之則稱為 FPU-disabled callback function。關於如何註冊 FPU-enabled 或 FPU-disabled callback 函式的詳細說明，請參考以下函式的說明：

- `attachCyclicCallback()`
- `attachErrorCallback()`
- `attachEventCallback()`

需要注意的是，儲存與還原 FPU 狀態會帶來一定的效能負擔，因此在性能與可靠性之間需做出適當的權衡。

attachCyclicCallback()

說明

註冊 Cyclic Callback 函式。

語法

```
int attachCyclicCallback(void (*callback)(void), bool use_fpu = false);
```

參數

- *[in] void (*callback)(void)*
  要註冊的 cyclic callback 函式，無參數、無回傳值。
- *[in] use_fpu*
  宣告此 callback 函式是否為 FPU 啟用函式。預設為 false。
  - true: FPU-enabled.
  - false: FPU-disabled

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 且尚未執行 `EthercatMaster::start()`，或在成功執行 `EthercatMaster::stop()` 且尚未執行 `EthercatMaster::end()` 的情況下呼叫。

有關使用上的限制，請參考 Callback 函式 章節。

範例

```
#include "Ethercat.h"
EthercatMaster master;

void CyclicCallback() {
    // put your cyclic Callback function here.
}
void setup() {
    master.begin();
    master.attachCyclicCallback(CyclicCallback);
    // ...
}
void loop() {
    // ...
}
```

detachCyclicCallback()

說明

取消註冊 Cyclic Callback 函式。

語法

```
int detachCyclicCallback();
```

參數

無。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 且尚未執行 `EthercatMaster::start()`，或在成功執行 `EthercatMaster::stop()` 且尚未執行 `EthercatMaster::end()` 的情況下呼叫。

範例

```
#include "Ethercat.h"
EthercatMaster master;

void CyclicCallback() {
    // ...
}
void setup() {
    master.begin();
    master.attachCyclicCallback(CyclicCallback);
    master.start(1000000, ECAT_SYNC);  // 1000000 ns = 1 ms

    delay(5000);
    master.stop();
    master.detachCyclicCallback();
    master.end();
}
void loop() {
    // ...
}
```

attachErrorCallback()

說明

註冊 Error Callback 函式。

語法

```
void attachErrorCallback(void (*callback)(uint32_t), bool use_fpu = false);
```

參數

- *[in] void (*callback)(uint32_t)*
  要註冊的 Error Callback 函式，僅有一個參數，為 32 位元的錯誤碼。
- *[in] use_fpu*
  宣告此 callback 函式是否為 FPU 啟用函式。預設為 false。
  - true: FPU-enabled.
  - false: FPU-disabled

傳回值

無。

備註

此函式必須在 `EthercatMaster::begin()` 之前或 `EthercatMaster::end()` 之後呼叫。

有關使用上的限制，請參考 Callback 函式 章節。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

uint32_t ErrorCode = 0;

void ErrorCallback(uint32_t errorcode) {
    ErrorCode = errorcode;
}

void setup() {
    Serial.begin(115200);
    master.attachErrorCallback(ErrorCallback); // ErrorCallback Function.

    master.begin();
    slave.attach(0, master);
    master.start(1000000, ECAT_SYNC);  // 1000000 ns = 1 ms
```

```
}

void loop() {
    /* Error Code */
    uint32_t error = ErrorCode;
    ErrorCode = 0;
    if (error) {
        Serial.print("ErrorCode:");
        Serial.println(error);
    }
}
```

detachErrorCallback()

說明

取消註冊 Error Callback 函式。

語法

```
void detachErrorCallback();
```

參數

無。

傳回值

無。

備註

此函式必須在 `EthercatMaster::begin()` 之前或 `EthercatMaster::end()` 之後呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;

void ErrorCallback(uint32_t errorcode) {
  // ...
}

void setup() {
  master.attachErrorCallback(ErrorCallback);
  master.begin();
  // ...
  master.end();
  master.detachEventCallback();
}

void loop() {
  // ...
}
```

attachEventCallback()

說明

註冊 Event Callback 函式.

語法

```
void attachEventCallback(void (*callback)(uint32_t), bool use_fpu = false);
```

參數

- *[in] void (*callback)(uint32_t)*
  要註冊的 event callback 函式，具有一個參數，為 32 位元的事件代碼。
- *[in] use_fpu*
  宣告此 callback 函式是否為 FPU 啟用函式。預設為 false。
  - true: FPU-enabled.
  - false: FPU-disabled

傳回值

無。

備註

此函式必須在 `EthercatMaster::begin()` 之前或 `EthercatMaster::end()` 之後呼叫。

有關使用上的限制，請參考 Callback 函式 章節。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

uint32_t EventCode = 0;

void EventCallback(uint32_t eventcode) {
  EventCode = eventcode;
}

void setup() {
  Serial.begin(115200);
  master.attachEventCallback(EventCallback); // EventCallback Function.

  master.begin();
  slave.attach(0, master);
  master.start(1000000, ECAT_SYNC);  // 1000000 ns = 1 ms
}
```

```
void loop() {
  /* Event Code */
  uint32_t event = EventCode;
  EventCode = 0;
  if (event) {
    Serial.print("EventCode:");
    Serial.println(event);
  }
}
```

detachEventCallback()

說明

取消註冊 Event Callback 函式.

語法

`void detachEventCallback();`

參數

無。

傳回值

無。

備註

此函式必須在 `EthercatMaster::begin()` 之前或 `EthercatMaster::end()` 之後呼叫。

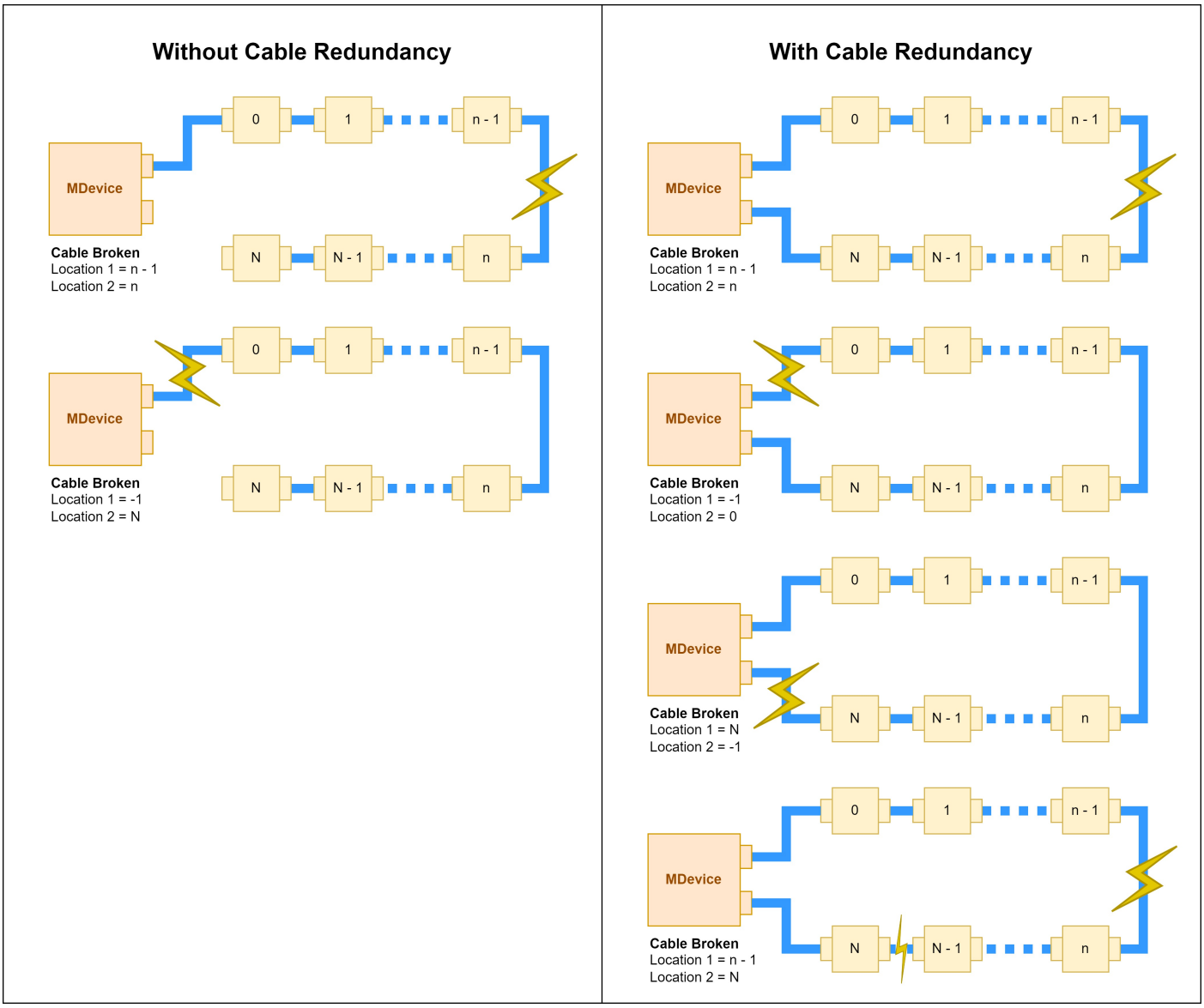
範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void EventCallback(uint32_t eventcode){

}

void setup() {
  master.attachEventCallback(EventCallback);
  master.begin();
  slave.attach(0, master);
  master.start();

  delay(5000);
  master.stop();
  master.detachEventCallback();
  master.end();
}

void loop() {

}
```

errGetCableBrokenLocation1()

說明

取得 ECAT_ERR_CABLE_BROKEN 錯誤內容，該錯誤代表「斷線位置 1」(Cable Broken Location 1)。關於「斷線位置 1」的定義，請參考下方圖示說明。



語法

```
int errGetCableBrokenLocation1();
```

參數

無。

傳回值

回傳斷線位置 1。

備註

此函式僅能在 error callback 函式中呼叫。

範例

```
#include <Ethercat.h>

EthercatMaster master;

bool CableBrokenLatched = false;
int CableBrokenLocation1;
int CableBrokenLocation2;

void ErrorCallback(uint32_t errorcode)
{
  if (errorcode == ECAT_ERR_CABLE_BROKEN) {
    if (CableBrokenLatched == false) {
      CableBrokenLatched = true;
      CableBrokenLocation1 = master.errGetCableBrokenLocation1();
      CableBrokenLocation2 = master.errGetCableBrokenLocation2();
    }
  }
}

void setup() {
  Serial.begin(115200);
  master.attachErrorCallback(ErrorCallback);
  master.begin();
  master.start();
}

void loop() {
  if (CableBrokenLatched == true) {
    Serial.print("Cable broken between ");
    if (CableBrokenLocation1 < 0) Serial.print("Primary Port");
    else Serial.print("Slave "); Serial.print(CableBrokenLocation1);
    Serial.print(" and ");
    if (CableBrokenLocation2 < 0) Serial.println("Secondary Port");
    else Serial.println("Slave "); Serial.println(CableBrokenLocation2);
    CableBrokenLatched = false;
  }
}
```

errGetCableBrokenLocation2()

說明

取得 ECAT_ERR_CABLE_BROKEN 錯誤內容，該錯誤代表「斷線位置 2」(Cable Broken Location 2)。關於「斷線位置 2」的定義，請參考 errGetCableBrokenLocation1() 說明。

語法

```
int errGetCableBrokenLocation2();
```

參數

無。

傳回值

回傳斷線位置 2。

備註

此函式僅能在 error callback 函式中呼叫。

範例

```
#include <Ethercat.h>

EthercatMaster master;

bool CableBrokenLatched = false;
int CableBrokenLocation1;
int CableBrokenLocation2;

void ErrorCallback(uint32_t errorcode)
{
  if (errorcode == ECAT_ERR_CABLE_BROKEN) {
    if (CableBrokenLatched == false) {
      CableBrokenLatched = true;
      CableBrokenLocation1 = master.errGetCableBrokenLocation1();
      CableBrokenLocation2 = master.errGetCableBrokenLocation2();
    }
  }
}

void setup() {
  Serial.begin(115200);
  master.attachErrorCallback(ErrorCallback);
  master.begin();
  master.start();
}
```

```
void loop() {
  if (CableBrokenLatched == true) {
    Serial.print("Cable broken between ");
    if (CableBrokenLocation1 < 0) Serial.print("Primary Port");
    else Serial.print("Slave "); Serial.print(CableBrokenLocation1);
    Serial.print(" and ");
    if (CableBrokenLocation2 < 0) Serial.println("Secondary Port");
    else Serial.println("Slave "); Serial.println(CableBrokenLocation2);
    CableBrokenLatched = false;
  }
}
```

evtGetMasterState()

說明

取得 ECAT_EVT_STATE_CHANGED 事件的內容，此事件代表 EtherCAT 主站的狀態變化。

語法

```
int evtGetMasterState();
```

參數

無。

傳回值

回傳 EtherCAT 主站的狀態。

備註

此函式僅能在 event callback 函式中呼叫。

範例

```
#include <Ethercat.h>

EthercatMaster master;
int CurrentMasterState;

void EventCallback(uint32_t eventcode)
{
  if (eventcode == ECAT_EVT_STATE_CHANGED)
    CurrentMasterState = master.evtGetMasterState();
}

void setup() {
  Serial.begin(115200);
  master.attachEventCallback(EventCallback);
  master.begin();
  master.start();
}

void loop() {
  Serial.print("CurrentMasterState: ");
  Serial.println(CurrentMasterState);
  delay(1000);
}
```

2.1.4 從站裝置資訊函式

本函式庫提供用於獲取網路上 EtherCAT 從站裝置資訊的函式。它包含查詢網路中從站裝置的數量，透過序號取得從站裝置的 Vendor ID、Product Code、Alias Address，以及使用上述資訊反查從站裝置編號。

此功能可用於辨識從站裝置的類型，並選擇對應的 EtherCAT 從站裝置類別進行掛載與操作。

函式：

- getSlaveCount()
- getVendorID()
- getProductCode()
- getRevisionNumber()
- getSerialNumber()
- getAliasAddress()
- getSlaveNo()

getSlaveCount()

說明

取得目前 EtherCAT 網路上的從站裝置數量。

語法

`uint16_t getSlaveCount();`

參數

無。

傳回值

回傳網路上從站裝置的數量。如果發生錯誤，則回傳 0。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，且可在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;

void setup() {
  uint16_t slavecount, i;
  Serial.begin(115200);
  master.begin();
  slavecount = master.getSlaveCount();
  for (i = 0; i < slavecount; i++) {
    Serial.print("Slave");
    Serial.print(i);

  }
}

void loop() {
  // ...
}
```

getVendorID()

說明

取得指定 EtherCAT 從站裝置的 Vendor ID。

語法

```
uint32_t getVendorID(uint16_t slave_no);
```

參數

- *[in] unit16_t slave_no*
  指定 EtherCAT 從站裝置在網路中的序號，0 表示第一個從站，1 表示第二個，依此類推。

傳回值

回傳指定裝置的 Vendor ID。若發生錯誤，則回傳 UINT32_MAX ($2^{32} - 1$)。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，且可在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;

void setup() {
  uint32_t slavevendorID;
  uint16_t slavecount = 1;

  Serial.begin(115200);

  master.begin();
  slavevendorID = master.getVendorID(slavecount);
  Serial.println(slavevendorID);
}

void loop() {
  // ...
}
```

getProductCode()

說明

取得指定 EtherCAT 從站裝置的 Product Code。

語法

```
uint32_t getProductCode(uint16_t slave_no);
```

參數

- *[in] unit16_t slave_no*
  指定 EtherCAT 從站裝置在網路中的序號，0 表示第一個從站，1 表示第二個，依此類推。

傳回值

回傳指定裝置的 Product Code。若發生錯誤，則回傳 UINT32_MAX ($2^{32} - 1$)。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，且可在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;

void setup() {
    uint32_t slaveproductcode;
    uint16_t slavecount=1;

    Serial.begin(115200);

    master.begin();
    slaveproductcode = master.getProductCode(slavecount);
    Serial.println(slaveproductcode);
}
void loop() {
    // ...
}
```

getRevisionNumber()

說明

取得指定 EtherCAT 從站裝置的 Revision Number。

語法

```
uint32_t getRevisionNumber(uint16_t slave_no);
```

參數

- *[in] unit16_t slave_no*
  指定 EtherCAT 從站裝置在網路中的序號，0 表示第一個從站，1 表示第二個，依此類推。

傳回值

回傳指定裝置的 Revision Number。若發生錯誤，則回傳 UINT32_MAX ($2^{32} - 1$)。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，且可在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;

void setup() {
    uint32_t slaverevisionnumber;
    uint16_t slavecount=1;

    Serial.begin(115200);

    master.begin();
    slaverevisionnumber = master.getRevisionNumber(slavecount);
    Serial.println(slaverevisionnumber);
}
void loop() {
    // ...
}
```

getSerialNumber()

說明

取得指定 EtherCAT 從站裝置的 Serial Number。

語法

```
uint32_t getSerialNumber(uint16_t slave_no);
```

參數

- *[in] unit16_t slave_no*
  指定 EtherCAT 從站裝置在網路中的序號，0 表示第一個從站，1 表示第二個，依此類推。

傳回值

回傳指定裝置的 Serial Number。若發生錯誤，則回傳 UINT32_MAX ($2^{32} - 1$)。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，且可在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;

void setup() {
    uint32_t slaveserialnum;
    uint16_t slavecount=1;

    Serial.begin(115200);

    master.begin();
    slaveserialnum = master.getSerialNumber(slavecount);
    Serial.println(slaveserialnum);
}
void loop() {
    // ...
}
```

getAliasAddress()

說明

取得指定 EtherCAT 從站裝置的 Alias Address。

語法

```
int getAliasAddress(uint16_t slave_no);
```

參數

- *[in] unit16_t slave_no*
  指定 EtherCAT 從站裝置在網路中的序號，0 表示第一個從站，1 表示第二個，依此類推。

傳回值

回傳指定從站的 Alias Address。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，且可在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;

void setup() {
    uint32_t slavealiasaddress;
    uint16_t slavecount=1;

    Serial.begin(115200);

    master.begin();
    slavealiasaddress = master.getAliasAddress(slavecount);
    Serial.println(slavealiasaddress);
}
void loop() {
    // ...
}
```

getSlaveNo()

說明

根據輸入的 Alias Address、Vendor ID、Product Code、Revision Number 或 Serial Number，尋找符合條件的 EtherCAT 從站裝置在網路中的序號。

語法

```
int getSlaveNo(uint16_t alias_addr);
int getSlaveNo(uint32_t vendor, uint32_t product);
int getSlaveNo(uint32_t vendor, uint32_t product, uint32_t revision, uint32_t serial_num);
int getSlaveNo(uint16_t alias_addr, uint32_t vendor, uint32_t product);
int getSlaveNo(uint16_t alias_addr, uint32_t vendor, uint32_t product, uint32_t revision, uint32_t serial_num);
```

參數

- *[in] uint16_t alias_addr*
  想要查找的從站裝置的 Alias Address。如果輸入為 0，表示查找時不使用此參數。
- *[in] uint32_t vendor*
  想要查找的從站裝置的 Vendor ID。如果輸入為 0，表示查找時不使用此參數。
- *[in] uint32_t product*
  想要查找的從站裝置的 Product Code。如果輸入為 0，表示查找時不使用此參數。
- *[in] uint32_t revision*
  想要查找的從站裝置的 Revision Number。如果輸入為 0，表示查找時不使用此參數。
- *[in] uint32_t serial_num*
  想要查找的從站裝置的 Serial Number。如果輸入為 0，表示查找時不使用此參數。

傳回值

回傳符合條件的從站裝置在網路中的序號。若回傳值為 -1，表示查無符合條件的從站裝置。若回傳值小於 0 的其他值，表示出現錯誤並回傳錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，且可在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;

void setup() {
  Serial.begin(115200);
```

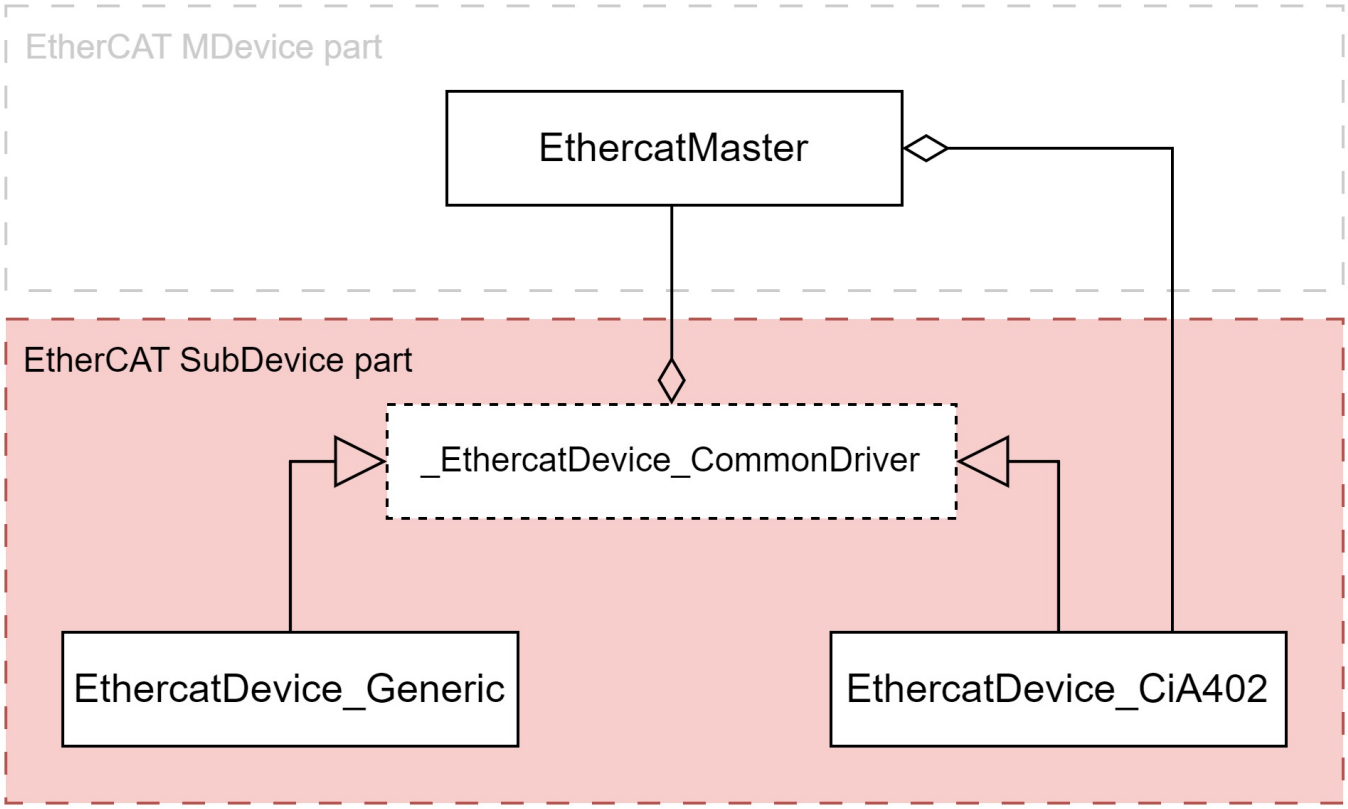
```
  master.begin();

  Serial.print("Search Test 1 => ");
  Serial.println(master.getSlaveNo(1200));

  Serial.print("Search Test 2 => ");
  Serial.println(master.getSlaveNo(0x00000BC3, 0x0086D324));

  Serial.print("Search Test 3 => ");
  Serial.println(master.getSlaveNo(0x00000BC3, 0x0086D304, 0x20220316,
0x00000000));

  Serial.print("Search Test 4 => ");
  Serial.println(master.getSlaveNo(251, 0x00000BC3, 0x0086D302));

  Serial.print("Search Test 5 => ");
  Serial.println(master.getSlaveNo(252, 0x00000BC3, 0x0086D301, 0x20220316,
0x00000000));
}

void loop() {
  // ...
}
```

2.2 EtherCAT SubDevice

EtherCAT SubDevice 部分提供了通用的從站裝置類別，可操作 PDO、CoE、FoE 等功能，並包含 CiA 402 通用從站裝置類別。

EtherCAT SubDevice 與 EtherCAT MDevice 部分之間的主要類別關係為關聯關係 (Association)，EtherCAT SubDevice 部分依賴於 EtherCAT MDevice 部分。

如下圖所示，_EthercatDevice_CommonDriver 與 EthercatMaster 之間存在關聯關係。



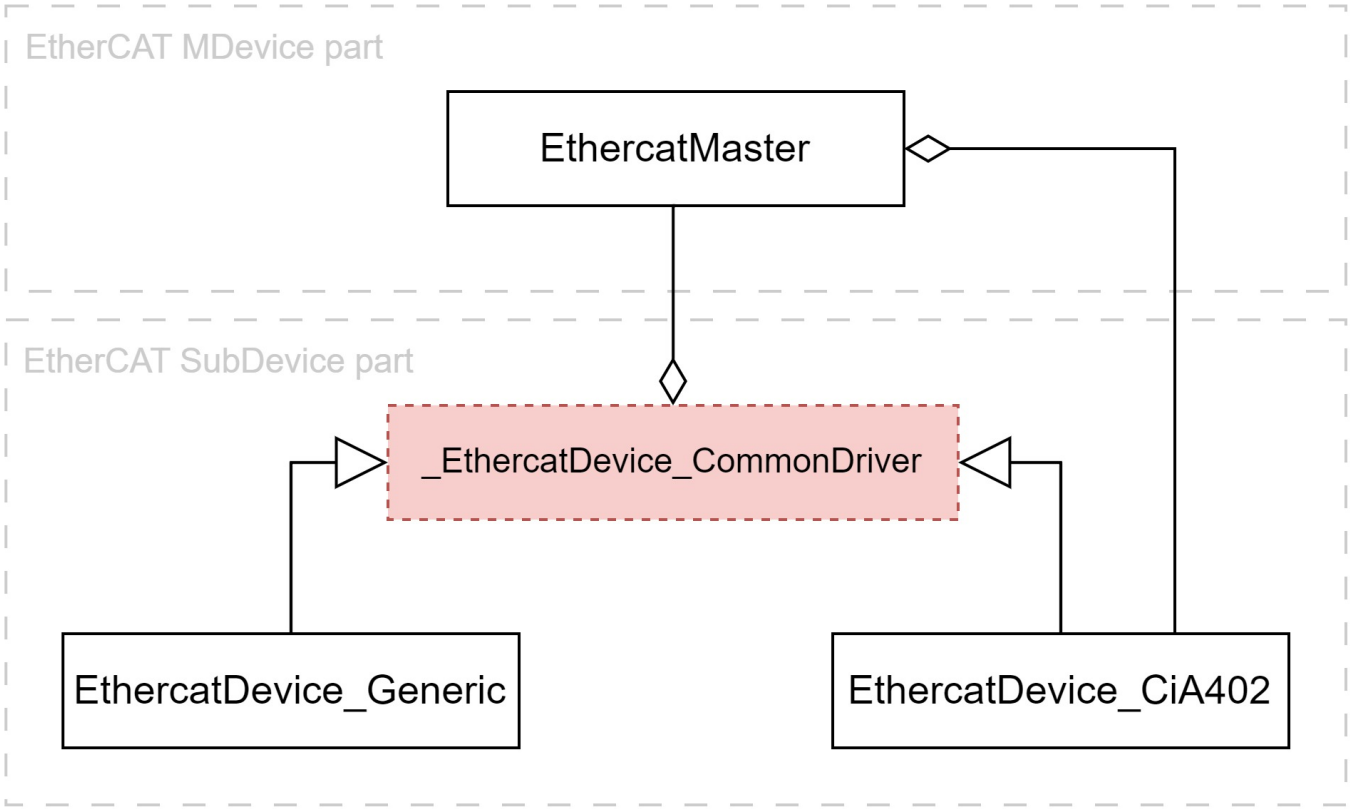
類別：

- _EthercatDevice_CommonDriver
- EthercatDevice_Generic
- EthercatDevice_CiA402

2.2.1 _EthercatDevice_CommonDriver

_EthercatDevice_CommonDriver 是一個抽象類別，不僅具備存取從站裝置資訊的功能，還提供多種 EtherCAT 功能的存取方法，包括 PDO、SII、CoE、FoE、DC 等。所有 EtherCAT 從站裝置類別皆繼承自此類別。

_EthercatDevice_CommonDriver 的類別關係如下圖所示：



- EthercatMaster 與 _EthercatDevice_CommonDriver 之間存在關聯關係，且 _EthercatDevice_CommonDriver 依賴於 EthercatMaster。
- 所有其他 EtherCAT 從站裝置類別皆繼承自 _EthercatDevice_CommonDriver。

警告：禁止使用此類別建立物件。

函式分類：

- 裝置資訊
- PDO
- CoE
- FoE
- DC
- SII EEPROM

函式：

函式名稱	說明	Callback 中呼叫
從站裝置資訊相關函式		
getVendorID()	取得廠商 ID	O
getProductCode()	取得產品代碼	O
getRevisionNumber()	取得版本號	O
getSerialNumber()	取得序號	O
getAliasAddress()	取得別名位址	O
getSlaveNo()	取得從站裝置在 EtherCAT 網路上的序號	O
getDeviceName()	取得裝置名稱	O
getMailboxProtocol()	取得支援的 Mailbox 協定類型	O
getCoEDetails()	取得支援的 CoE 詳細資訊	O
getFoEDetails()	取得支援的 FoE 詳細資訊	O
getEoEDetails()	取得支援的 EoE 詳細資訊	O
getSoEChannels()	取得支援的 SoE 通道數	O
isSupportDC()	檢查從站裝置是否支援 DC	O
PDO 存取函式		
pdoBitWrite()	寫入 1-bit 輸出處理資料	O
pdoBitRead()	讀取 1-bit 輸入處理資料	O
pdoGetOutputBuffer()	取得輸出處理資料的記憶體指標	O
pdoGetInputBuffer()	取得輸入處理資料的記憶體指標	O
pdoWrite()	寫入多位元組輸出處理資料	O
pdoWrite8()	寫入 8-bit 輸出處理資料	O
pdoWrite16()	寫入 16-bit 輸出處理資料	O
pdoWrite32()	寫入 32-bit 輸出處理資料	O
pdoWrite64()	寫入 64-bit 輸出處理資料	O
pdoRead()	讀取多位元組輸入處理資料	O
pdoRead8()	讀取 8-bit 輸入處理資料	O
pdoRead16()	讀取 16-bit 輸入處理資料	O
pdoRead32()	讀取 32-bit 輸入處理資料	O
pdoRead64()	讀取 64-bit 輸入處理資料	O
CoE 通訊函式		
sdoDownload()	將多位元組資料寫入物件	
sdoDownload8()	將 8-bit 資料寫入物件	
sdoDownload16()	將 16-bit 資料寫入物件	
sdoDownload32()	將 32-bit 資料寫入物件	
sdoDownload64()	將 64-bit 資料寫入物件	

sdoUpload()	讀取物件中的多位元組資料	
sdoUpload8()	讀取 8-bit 資料	
sdoUpload16()	讀取 16-bit 資	
sdoUpload32()	讀取 32-bit 資料	
sdoUpload64()	讀取 64-bit 資料	
getODlist()	取得物件字典中存在的物件清單	
getObjectDescription()	取得物件的說明資訊	
getEntryDescription()	取得物件子項目的說明資訊	
FoE 通訊函式		
readFoE()	從從站裝置讀取檔案	
writeFoE()	向從站裝置寫入檔案	
DC 同步設定函式		
setDc()	設定 DC 參數	
SII EEPROM 存取函式		
writeSII()	寫入多位元組資料至 SII EEPROM	
writeSII8()	寫入 8-bit 資料至 SII EEPROM	
writeSII16()	寫入 16-bit 資料至 SII EEPROM	
writeSII32()	寫入 32-bit 資料至 SII EEPROM	
readSII()	讀取多位元組資料從 SII EEPROM	
readSII8()	讀取 8-bit 資料從 SII EEPROM	
readSII16()	讀取 16-bit 資料從 SII EEPROM	
readSII32()	讀取 32-bit 資料從 SII EEPROM	

裝置資訊函式

本函式庫提供多種函式，可用於獲取網路上 EtherCAT 從站裝置的相關資訊。這些資訊包括：廠商 ID（Vendor ID）、產品代碼（Product Code）、別名位址（Alias Address）與裝置名稱（Device Name），可用於從站裝置的識別。

此外，亦可查詢從站裝置是否支援特定功能，如 CoE、FoE、DC 等。這些從站資訊可幫助使用者了解裝置在網路中的特性與能力，進而執行對應的組態與控制操作。

函式：

- getVendorID()
- getProductCode()
- getRevisionNumber()
- getSerialNumber()
- getAliasAddress()
- getSlaveNo()
- getDeviceName()
- getMailboxProtocol()
- getCoEDetails()
- getFoEDetails()
- getEoEDetails()
- getSoEChannels()
- isSupportDC()

getVendorID()

說明

取得目前 EtherCAT 從站裝置的 Vendor ID。

語法

```
uint32_t getVendorID();
```

參數

無。

傳回值

回傳該 EtherCAT 從站裝置的 Vendor ID。若發生錯誤，將回傳 UINT32_MAX (2^{32} - 1)。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，且可在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);

  Serial.println(slave.getVendorID());
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

getProductCode()

說明

取得目前 EtherCAT 從站裝置的 Product Code。

語法

```
uint32_t getProductCode();
```

參數

無。

傳回值

回傳該 EtherCAT 從站裝置的 Product Code。若發生錯誤，將回傳 UINT32_MAX (2^{32} - 1)。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，且可在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);

  Serial.println(slave.getProductCode());
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

getRevisionNumber()

說明

取得目前 EtherCAT 從站裝置的 Revision Number。

語法

```
uint32_t getRevisionNumber();
```

參數

無。

傳回值

回傳該 EtherCAT 從站裝置的 Revision Number。若發生錯誤，將回傳 UINT32_MAX (2^{32} - 1)。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，且可在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);

  Serial.println(slave.getRevisionNumber());
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

getSerialNumber()

說明

取得目前 EtherCAT 從站裝置的 Serial Number。

語法

```
uint32_t getSerialNumber();
```

參數

無。

傳回值

回傳該 EtherCAT 從站裝置的 Serial Number。若發生錯誤，將回傳 UINT32_MAX (2^{32} - 1)。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，且可在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);

  Serial.println(slave.getSerialNumber());
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

getAliasAddress()

說明

取得目前 EtherCAT 從站裝置的 Alias Address。

語法

```
int getAliasAddress();
```

參數

無。

傳回值

回傳該 EtherCAT 從站裝置的 Alias Address。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，且可在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);

  Serial.println(slave.getAliasAddress());
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

getSlaveNo()

說明

取得目前 EtherCAT 從站裝置在 EtherCAT 網路上的順序編號。

語法

```
int getSlaveNo();
```

參數

無。

傳回值

回傳該 EtherCAT 從站裝置在網路中的順序編號。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，且可在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);

  Serial.println(slave.getSlaveNo());
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

getDeviceName()

說明

取得 EtherCAT 從站裝置的裝置名稱。

語法

```
char *getDeviceName(char *name, size_t len);
```

參數

- *[out] char *name*
  用於儲存裝置名稱的緩衝區 (buffer)。
- *[in] size_t len*
  緩衝區 name 的大小

傳回值

回傳指向儲存裝置名稱的緩衝區指標。若回傳值為 NULL，表示發生錯誤。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，且可在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  char name[64];
  Serial.println(slave.getDeviceName(name, 64));
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

getMailboxProtocol()

說明

取得 EtherCAT 從站裝置所支援的 Mailbox 協定類型。

語法

```
int getMailboxProtocol();
```

參數

無。

傳回值

回傳所支援的 Mailbox 協定類型：

- Bit 0: ADS over EtherCAT.
- Bit 1: Ethernet over EtherCAT.
- Bit 2: CAN application protocol over EtherCAT.
- Bit 3: File Access over EtherCAT.
- Bit 4: Servo Drive Profile over EtherCAT.
- Bit 5: Vendor specific protocol over EtherCAT.

若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，且可在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  Serial.begin(115200);
  master.begin();
  slave.attach(0, master);

  Serial.print("Mailbox Protocols: ");
  Serial.println(slave.getMailboxProtocol());
}

void loop() {
  // ...
}
```

getCoEDetails()

說明

取得 EtherCAT 從站裝置所支援的 CoE (CAN application protocol over EtherCAT) 細節功能。

語法

```
int getCoEDetails();
```

參數

無。

傳回值

返回 EtherCAT 從站設備支援的 CoE 的詳細資訊。

- Bit 0: Enable SDO.
- Bit 1: Enable SDO Info.
- Bit 2: Enable PDO Assign.
- Bit 3: Enable PDO Configuration.
- Bit 4: Enable PDO Upload at startup.
- Bit 5: Enable SDO complete access.

若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，且可在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  Serial.begin(115200);
  master.begin();
  slave.attach(0, master);
  Serial.print("CoE Details: ");   Serial.println(slave.getCoEDetails());
}

void loop() {
  // ...
}
```

getFoEDetails()

說明

取得 EtherCAT 從站裝置所支援的 FoE (File Access over EtherCAT) 細節功能。

語法

```
int getFoEDetails();
```

參數

無。

傳回值

返回 EtherCAT 從站設備支援的 FoE 的詳細資訊。

- Bit 0: Enable FoE.

若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，且可在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);

  Serial.println(slave.getFoEDetails());
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

getEoEDetails()

說明

取得 EtherCAT 從站裝置所支援的 EoE (Ethernet over EtherCAT) 細節功能。

語法

```
int getEoEDetails();
```

參數

無。

傳回值

返回 EtherCAT 子設備支援的 EoE 的詳細資訊。

- Bit 0: Enable EoE.

若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，且可在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);

  Serial.println(slave.getEoEDetails());
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

getSoEChannels()

說明

取得 EtherCAT 從站裝置所支援的 SoE 通道數量。

語法

```
int getSoEChannels();
```

參數

無。

傳回值

返回 EtherCAT 子設備支援的 SoE 通道數。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，且可在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);

  Serial.println(slave.getSoEChannels());
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

isSupportDC()

說明

檢查 EtherCAT 從站裝置是否支援 DC（Distributed Clocks）。

語法

```
int isSupportDC();
```

參數

無。

傳回值

返回 EtherCAT 從站設備是否支援分散式時鐘 (DC)。正值表示支持，0 表示不支持，負值表示錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，且可在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);

  Serial.println(slave.isSupportDC());
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Process Data Objects (PDO) 函式

處理資料 (Process Data) 是指 EtherCAT 網路中主站與從站裝置之間交換的即時通訊資料。這些資料通常用於控制、監控與通訊等目的。EtherCAT 主站會以週期性方式傳送處理資料，以控制並監控所有從站裝置，確保高同步性與低延遲。

在 EtherCAT 從站裝置控制器（ESC）中，現場匯流排記憶體管理單元（FMMU） 可將雙埠記憶體映射至邏輯位址。所有從站節點會檢查來自主站的 EtherCAT 封包，並將封包中處理資料的邏輯位址與 FMMU 中設定的位址進行比對。若位址吻合，則輸出處理資料將被寫入雙埠記憶體，而輸入處理資料則會被插入至 EtherCAT 封包中。

整體而言，處理資料是 EtherCAT 技術的核心組成，廣泛應用於機器人控制、CNC 控制、自動化控制等即時性應用領域。

函式：

- pdoBitWrite()
- pdoBitRead()
- pdoGetOutputBuffer()
- pdoGetInputBuffer()
- pdoWrite()
- pdoWrite8()
- pdoWrite16()
- pdoWrite32()
- pdoWrite64()
- pdoRead()
- pdoRead8()
- pdoRead16()
- pdoRead32()
- pdoRead64()

pdoBitWrite()

說明

將指定的位元值寫入 EtherCAT 從站設備的的輸出處理資料 (Output Process Data)。

語法

```
int pdoBitWrite(uint32_t bit_offset, uint8_t value);
```

參數

- *[in] uint32_t bit_offset*
  EtherCAT 從站輸出處理資料的位元偏移值。
- *[in] unit_8_t value*
  要寫入 EtherCAT 從站設備輸出處理資料的位元值。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::start()` 且尚未執行 `EthercatMaster::stop()` 的情況下呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  master.start(1000000);
}

void loop() {
  slave.pdoBitWrite(0, 1);
  delay(500);
  slave.pdoBitWrite(0, 0);
  delay(1000);
}
```

pdoBitRead()

說明

從 EtherCAT 從站設備的輸入處理資料 (Input Process Data) 中讀取指定的位元值。

語法

```
int pdoBitRead(uint32_t bit_offset);
```

參數

- *[in] uint32_t bit_offset*
  EtherCAT 從站要讀取的輸入處理資料位元偏移值。

傳回值

從該子設備的輸入過程資料中讀取指定的位元值。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::start()` 且尚未執行 `EthercatMaster::stop()` 的情況下呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  Serial.begin(115200);
  master.begin();
  slave.attach(0, master);
  master.start(1000000);
}

void loop() {
  Serial.print("Bit-0 Value: ");
  Serial.println(slave.pdoBitRead(0));
  Serial.print("Bit-7 Value: ");
  Serial.println(slave.pdoBitRead(7));
  delay(1000);
}
```

pdoGetOutputBuffer()

說明

取得此 EtherCAT 從站之輸出處理資料 (Output Process Data) 的記憶體指標。

語法

```
uint8_t *pdoGetOutputBuffer();
```

參數

無。

傳回值

回傳該從站輸出處理資料的記憶體指標 (memory pointer)。若回傳值為 NULL，表示發生錯誤或該從站沒有輸出處理資料。

備註

此函式必須在成功執行 `EthercatMaster::start()` 且尚未執行 `EthercatMaster::stop()` 的情況下呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

volatile uint8_t *pointer;

void setup() {
  master.begin();
  slave.attach(0, master);
  master.start(1000000);
  pointer = slave.pdoGetOutputBuffer();
}

void loop() {
  if (pointer != NULL) {
    pointer[0] = 0x55;
    delay(500);
    pointer[0] = 0xaa;
    delay(1000);
  }
}
```

pdoGetInputBuffer()

說明

取得此 EtherCAT 從站之輸入處理資料 (Input Process Data) 的記憶體指標。

語法

```
uint8_t *pdoGetInputBuffer();
```

參數

無。

傳回值

回傳該從站輸入處理資料的記憶體指標。若回傳值為 NULL，表示發生錯誤或該從站沒有輸入處理資料。

備註

此函式必須在成功執行 `EthercatMaster::start()` 且尚未執行 `EthercatMaster::stop()` 的情況下呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

volatile uint8_t *pointer;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  master.start(1000000);

  pointer = slave.pdoGetInputBuffer();
}

void loop() {
  if (pointer != NULL) {
    Serial.print("Byte-0 Value:");
    Serial.println(pointer[0]);
    delay(1000);
  }
}
```

pdoWrite()

說明

將資料寫入 EtherCAT 從站輸出處理資料 (Output Process Data) 的指定位元組偏移位置，並指定寫入資料的大小。

語法

```
int pdoWrite(uint32_t offset, void *data, uint32_t size);
```

參數

- *[in] uint32_t offset*
  EtherCAT 從站輸出處理資料的位元組偏移值。
- *[in] void *data*
  指向輸出資料緩衝區的指標。
- *[in] uint32_t size*
  欲寫入的資料大小 (位元組)。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::start()` 且尚未執行 `EthercatMaster::stop()` 的情況下呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

uint8_t buffer[4] = {0x00, 0x55, 0xAA, 0xFF};

void setup() {
  master.begin();
  slave.attach(0, master);
  master.start(1000000);
}

void loop() {
  buffer[0] = ~buffer[0];
  buffer[1] = ~buffer[1];
  buffer[2] = ~buffer[2];
  buffer[3] = ~buffer[3];
  slave.pdoWrite(0, buffer, 4);
```

```
  delay(1000);
}
```

pdoWrite8()

說明

將 8-bit 資料寫入 EtherCAT 從站輸出處理資料 (Output Process Data) 的指定位元組偏移位置。

語法

```
int pdoWrite8(uint32_t offset, uint8_t value);
```

參數

- *[in] uint32_t offset*
  EtherCAT 從站輸出處理資料的位元組偏移值。
- *[in] uint8_t value*
  欲寫入至輸出處理資料的 8 位元數值。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::start()` 且尚未執行 `EthercatMaster::stop()` 的情況下呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  master.start(1000000);
}

void loop() {
  slave.pdoWrite8(0, 0x55);
  delay(500);
  slave.pdoWrite8(0, 0xAA);
  delay(1000);
}
```

pdoWrite16()

說明

將 16-bit 資料寫入 EtherCAT 從站輸出處理資料 (Output Process Data) 的指定位元組偏移位置。

語法

```
int pdoWrite16(uint32_t offset, uint16_t value);
```

參數

- *[in] uint32_t offset*
  EtherCAT 從站輸出處理資料的位元組偏移值。
- *[in] uint16_t value*
  欲寫入至輸出處理資料的 16 位元數值。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::start()` 且尚未執行 `EthercatMaster::stop()` 的情況下呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  master.start(1000000);
}

void loop() {
  slave.pdoWrite16(0, 0x5555);
  delay(500);
  slave.pdoWrite16(0, 0xAAAA);
  delay(1000);
}
```

pdoWrite32()

說明

將 32-bit 資料寫入 EtherCAT 從站輸出處理資料 (Output Process Data) 的指定位元組偏移位置。

語法

```
int pdoWrite32(uint32_t offset, uint32_t value);
```

參數

- *[in] uint32_t offset*
  EtherCAT 從站輸出處理資料的位元組偏移值。
- *[in] uint32_t value*
  欲寫入至輸出處理資料的 32 位元數值。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::start()` 且尚未執行 `EthercatMaster::stop()` 的情況下呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  master.start(1000000);
}

void loop() {
  slave.pdoWrite32(0, 0x55555555);
  delay(500);
  slave.pdoWrite32(0, 0xAAAAAAAA);
  delay(1000);
}
```

pdoWrite64()

說明

將 64-bit 資料寫入 EtherCAT 從站輸出處理資料 (Output Process Data) 的指定位元組偏移位置。

語法

```
int pdoWrite64(uint32_t offset, uint64_t value);
```

參數

- *[in] uint32_t offset*
  從站輸出處理資料的位元組偏移值。
- *[in] uint64_t value*
  欲寫入至輸出處理資料的 32 位元數值。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::start()` 且尚未執行 `EthercatMaster::stop()` 的情況下呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  master.start(1000000);
}

void loop() {
  slave.pdoWrite64(0, 0x5555555555555555ULL);
  delay(500);
  slave.pdoWrite64(0, 0xAAAAAAAAAAAAAAAAULL);
  delay(1000);
}
```

pdoRead()

說明

從 EtherCAT 從站的輸入處理資料 (Input Process Data) 中，從指定的位元組偏移位置開始，讀取指定大小的資料。

語法

```
int pdoRead(uint32_t offset, void *data, uint32_t size);
```

參數

- *[in] uint32_t offset*
  EtherCAT 從站輸入處理資料的位元組偏移值。
- *[out] void *data*
  用於讀取輸入過程資料的資料緩衝區。
- *[in] uint32_t size*
  用於讀取輸入過程資料的資料緩衝區的大小。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::start()` 且尚未執行 `EthercatMaster::stop()` 的情況下呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

uint8_t buffer[4];

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  master.start(1000000);
}

void loop() {
  slave.pdoRead(0, buffer, 4);
  Serial.print("Buffer: ");
  Serial.print(buffer[0]);
```

```
  Serial.print(buffer[1]);
  Serial.print(buffer[2]);
  Serial.println(buffer[3]);
  delay(1000);
}
```

pdoRead8()

說明

從 EtherCAT 從站的輸入處理資料中，從指定的位元組偏移位置開始讀取一個 8 位元的資料。

語法

```
uint8_t pdoRead8(uint32_t offset);
```

參數

- *[in] uint32_t offset*
  EtherCAT 從站輸入處理資料的位元組偏移值。

傳回值

返回一個 8 位元的輸入過程資料。

備註

此函式必須在成功執行 `EthercatMaster::start()` 且尚未執行 `EthercatMaster::stop()` 的情況下呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup(){
    Serial.begin(115200);
    master.begin();
    slave.attach(0, master);
    master.start();
}

void loop() {
    slave.pdoWrite8(0, 0xFF);
    Serial.println(slave.pdoRead8(0), HEX);
    delay(1000);
}
```

pdoRead16()

說明

從 EtherCAT 從站的輸入處理資料中，從指定的位元組偏移位置開始讀取一個 16 位元的資料。

語法

```
uint16_t pdoRead16(uint32_t offset);
```

參數

- *[in] uint32_t offset*
  EtherCAT 從站輸入處理資料的位元組偏移值。

傳回值

返回一個 16 位元的輸入過程資料。

備註

此函式必須在成功執行 `EthercatMaster::start()` 且尚未執行 `EthercatMaster::stop()` 的情況下呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  master.start(1000000);
}

void loop() {
  Serial.print("Value: ");
  Serial.println(slave.pdoRead16(0));
  delay(1000);
}
```

pdoRead32()

說明

從 EtherCAT 從站的輸入處理資料中，從指定的位元組偏移位置開始讀取一個 32 位元的資料。

語法

```
uint32_t pdoRead32(uint32_t offset);
```

參數

- *[in] uint32_t offset*
  EtherCAT 從站輸入處理資料的位元組偏移值。

傳回值

返回一個 32 位元的輸入過程資料。

備註

此函式必須在成功執行 `EthercatMaster::start()` 且尚未執行 `EthercatMaster::stop()` 的情況下呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  master.start(1000000);
}

void loop() {
  Serial.print("Value: ");
  Serial.println(slave.pdoRead32(0));
  delay(1000);
}
```

pdoRead64()

說明

從 EtherCAT 從站的輸入處理資料中，從指定的位元組偏移位置開始讀取一個 64 位元的資料。

語法

```
uint64_t pdoRead64(uint32_t offset);
```

參數

- *[in] uint32_t offset*
  EtherCAT 從站輸入處理資料的位元組偏移值。

傳回值

返回一個 64 位元的輸入過程資料。

備註

此函式必須在成功執行 `EthercatMaster::start()` 且尚未執行 `EthercatMaster::stop()` 的情況下呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  master.start(1000000);
}

void loop() {
  Serial.print("Value: ");
  uint64_t value = slave.pdoRead64(0);
  char buffer[21];
  sprintf(buffer, "%llu", value);

  Serial.println(buffer);
  delay(1000);
}
```

CANopen over EtherCAT (CoE) 函式

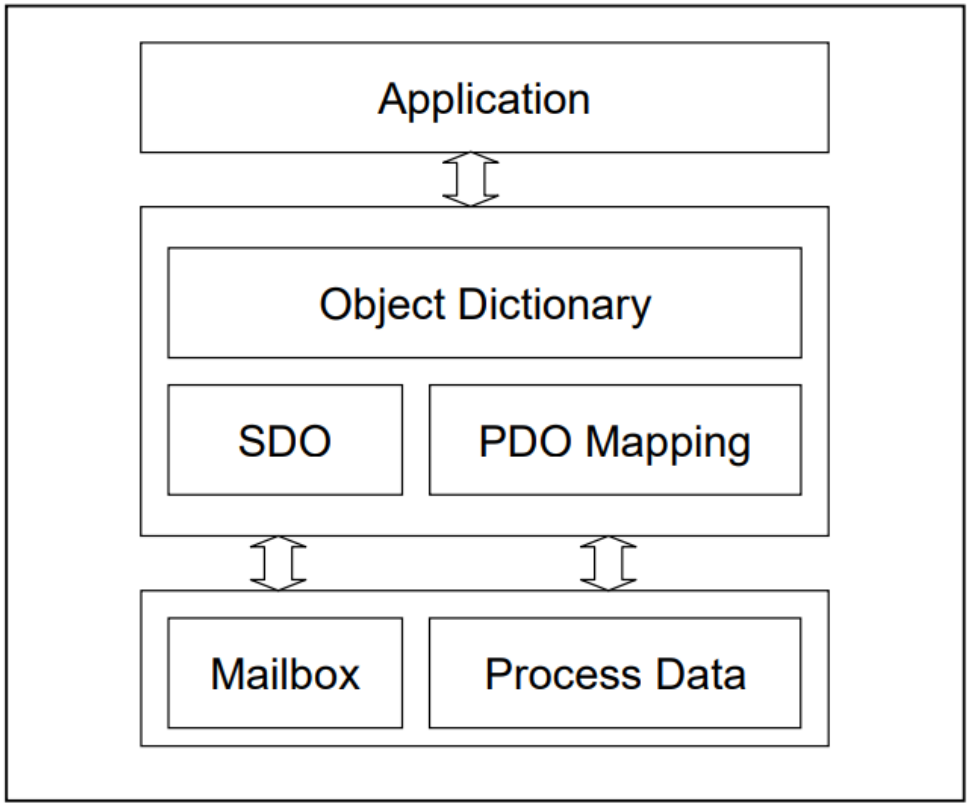
CANopen 是一種基於 CAN（Controller Area Network）匯流排的高階通訊協定，常用於工業應用中控制系統與裝置之間的通訊。該協定定義了一組通訊物件、資料型別與網路管理功能，以促進裝置之間的資料交換、設定與控制。

CANopen 協定包含以下幾個核心部分：

- **物件字典 (Object Dictionary)**
  定義了所有裝置間可交換的資料物件與參數，涵蓋變數、參數、事件與功能等類型。

- **PDO (Process Data Object)**
  用於即時資料傳輸。PDO 可固定週期或觸發方式傳輸資料，使裝置間能進行即時控制與資料交換。

- **SDO (Service Data Object)**
  用於裝置參數的設定與管理。SDO 提供讀寫與參數配置功能，允許裝置之間動態地交換組態資訊。



CoE (CAN application over EtherCAT) 是一種建立在 EtherCAT 網路上的 CANopen 協定。它支援透過 EtherCAT 網路使用 CANopen 協定進行通訊。物件字典包含參數、應用資料以及處理資料介面與應用資料（PDO 映射）之間的對應資訊，這些項目可以透過 SDO（Service Data Object） 存取。

SDO Service 主要由兩種類型的命令組成。 SDO 命令用於存取儲存在物件字典中的對象，而 SDO 資訊命令用於檢索有關這些對象的詳細資訊。

函式：

- **SDO 指令**
  - sdoDownload()
  - sdoDownload8()
  - sdoDownload16()
  - sdoDownload32()
  - sdoDownload64()
  - sdoUpload()
  - sdoUpload8()
  - sdoUpload16()
  - sdoUpload32()
  - sdoUpload64()
- **SDO 訊息指令**
  - getODlist()
  - getObjectDescription()
  - getEntryDescription()

sdoDownload()

說明

將多個位元組的資料寫入 EtherCAT 從站的指定物件中。

語法

```
int sdoDownload(
    uint16_t    od_index,
    uint8_t     od_subindex,
    void        *  data,
    uint32_t    size,
    uint32_t    *  abortcode = NULL,
    bool        complete_access = false,
    uint32_t    timeout_us = 2000000
);
```

參數

- *[in] uint16_t od_index*
  物件的索引。
- *[in] uint8_t od_subindex*
  物件的子索引。如果使用 Complete Access，則為 0 或 1。
- *[in] void data*
  要寫入的資料緩衝區。
- *[in] uint32_t size*
  要寫入的資料緩衝區大小。
- *[out] uint32_t abortcode*
  用於儲存 SDO Abort Code 的變數指標。預設為 NULL。
- *[in] bool complete_access*
  是否使用 Complete Access。
- *[in] uint32_t timeout_us*
  逾時時間 (微秒)。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
```

```
EthercatDevice_Generic slave;

uint16_t value = 0x000F;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  master.start(1000000);
}

void loop() {
  slave.sdoDownload(0x6040, 0x00, &value, sizeof(value));
  delay(1000);
}
```

sdoDownload8()

說明

將 8-bit 數值寫入 EtherCAT 從站的指定物件中。

語法

```
int sdoDownload8(uint16_t od_index, uint8_t od_subindex, uint8_t value,
uint32_t timeout_us = 2000000);
```

參數

- *[in] uint16_t od_index*
  物件的索引。
- *[in] uint8_t od_subindex*
  物件的子索引。
- *[in] uint8_t value*
  要寫入物件的 8 位元數值。
- *[in] uint32_t timeout_us*
  逾時時間（微秒）。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  Serial.begin(115200);
  master.begin();
  slave.attach(0, master);
  master.start(1000000);
}
void loop() {
  slave.sdoDownload8(0x6040, 0x00, 0x0F);
  delay(1000);
}
```

sdoDownload16()

說明

將 16-bit 數值寫入 EtherCAT 從站的指定物件中。

語法

```
int sdoDownload16(uint16_t od_index, uint8_t od_subindex, uint16_t value, uint32_t timeout_us = 2000000);
```

參數

- *[in] uint16_t od_index*
  物件的索引。
- *[in] uint8_t od_subindex*
  物件的子索引。
- *[in] uint16_t value*
  要寫入物件的 16 位元數值。
- *[in] uint32_t timeout_us*
  逾時時間（微秒）。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  master.start(1000000);
}

void loop() {
  slave.sdoDownload16(0x6040, 0x00, 0x000F);
  delay(1000);
}
```

sdoDownload32()

說明

將 32-bit 數值寫入 EtherCAT 從站的指定物件中。

語法

```
int sdoDownload32(uint16_t od_index, uint8_t od_subindex, uint32_t value, uint32_t timeout_us = 2000000);
```

參數

- *[in] uint16_t od_index*
  物件的索引。
- *[in] uint8_t od_subindex*
  物件的子索引。
- *[in] uint32_t value*
  要寫入物件的 32 位元數值。
- *[in] uint32_t timeout_us*
  逾時時間（微秒）。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  master.start(1000000);
}
void loop() {
  slave.sdoDownload32(0x607A, 0x00, 1000);
  delay(1000);
  slave.sdoDownload32(0x607A, 0x00, 0);
  delay(1000);
}
```

sdoDownload64()

說明

將 64-bit 數值寫入 EtherCAT 從站的指定物件中。

語法

```
int sdoDownload64(uint16_t od_index, uint8_t od_subindex, uint64_t value, uint32_t timeout_us = 2000000);
```

參數

- *[in] uint16_t od_index*
  物件的索引。
- *[in] uint8_t od_subindex*
  物件的子索引。
- *[in] uint64_t value*
  要寫入物件的 64 位元數值。
- *[in] uint32_t timeout_us*
  逾時時間（微秒）。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  master.start(1000000);
}
void loop() {
  slave.sdoDownload64(0x5000, 0x00, 100000);
  delay(1000);
  slave.sdoDownload64(0x5000, 0x00, 0);
  delay(1000);
}
```

sdoUpload()

說明

從 EtherCAT 從站的指定物件中讀取多個位元組的資料。

語法

```
int sdoUpload(
    uint16_t    od_index,
    uint8_t     od_subindex,
    void     *  data,
    uint32_t    size,
    uint32_t *  abortcode = NULL,
    bool        complete_access = false,
    uint32_t    timeout_us = 2000000
);
```

參數

- *[in] uint16_t od_index*
  物件的索引。
- *[in] uint8_t od_subindex*
  物件的子索引。若使用 Complete Access，請填 0 或 1。
- *[out] void *data*
  用來接收讀取資料的緩衝區。
- *[in] uint32_t size*
  讀取資料緩衝區的大小。
- *[out] uint32_t *abortcode*
  指向用來儲存 SDO Abort Code 的變數指標。
- *[in] bool complete_access*
  是否使用 Complete Access 模式。預設為 false。
- *[in] uint32_t timeout_us*
  逾時時間（單位為微秒）。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
```

```
EthercatDevice_Generic slave;

uint16_t value;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  master.start(1000000);
}

void loop() {
  slave.sdoUpload(0x6040, 0x00, &value, sizeof(value));
  Serial.print("Value: ");
  Serial.println(value);
  delay(1000);
}
```

sdoUpload8()

說明

從 EtherCAT 從站的指定物件中讀取 8-bit 的數值。

語法

```
uint8_t sdoUpload8(uint16_t od_index, uint8_t od_subindex, uint32_t timeout_us = 2000000);
```

參數

- *[in] uint16_t od_index*
  物件的索引。
- *[in] uint8_t od_subindex*
  物件的子索引。
- *[in] uint32_t timeout_us*
  逾時時間（單位為微秒）。預設為 2000000。

傳回值

傳回 8-bit 的數值。

備註

此函式必須在成功執行 EthercatMaster::begin() 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  master.start(1000000);
}

void loop() {
  Serial.print("Value: ");
  Serial.println(slave.sdoUpload8(0x6040, 0x00));
  delay(1000);
}
```

sdoUpload16()

說明

從 EtherCAT 從站的指定物件中讀取 16-bit 的數值。

語法

```
uint16_t sdoUpload16(uint16_t od_index, uint8_t od_subindex, uint32_t timeout_us = 2000000);
```

參數

- *[in] uint16_t od_index*
  物件的索引。
- *[in] uint8_t od_subindex*
  物件的子索引。
- *[in] uint32_t timeout_us*
  逾時時間（單位為微秒）。。預設為 2000000。

傳回值

傳回 16-bit 的數值。

備註

此函式必須在成功執行 EthercatMaster::begin() 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  master.start(1000000);
}

void loop() {
  Serial.print("Value: ");
  Serial.println(slave.sdoUpload16(0x6040, 0x00));
  delay(1000);
}
```

sdoUpload32()

說明

從 EtherCAT 從站的指定物件中讀取 32-bit 的數值。

語法

```
uint32_t sdoUpload32(uint16_t od_index, uint8_t od_subindex, uint32_t timeout_us = 2000000);
```

參數

- *[in] uint16_t od_index*
  物件的索引。
- *[in] uint8_t od_subindex*
  物件的子索引。
- *[in] uint32_t timeout_us*
  逾時時間（單位為微秒）。。預設為 2000000。

傳回值

傳回 32-bit 的數值。

備註

此函式必須在成功執行 EthercatMaster::begin() 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  master.start(1000000);
}

void loop() {
  Serial.print("Value: ");
  Serial.println(slave.sdoUpload32(0x607A, 0x00));
  delay(1000);
}
```

sdoUpload64()

說明

從 EtherCAT 從站的指定物件中讀取 64-bit 的數值。

語法

```
uint64_t sdoUpload64(uint16_t od_index, uint8_t od_subindex, uint32_t timeout_us = 2000000);
```

參數

- *[in] uint16_t od_index*
  物件的索引。
- *[in] uint8_t od_subindex*
  物件的子索引。
- *[in] uint32_t timeout_us*
  逾時時間（單位為微秒）。。預設為 2000000。

傳回值

傳回 64-bit 的數值。

備註

此函式必須在成功執行 EthercatMaster::begin() 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  master.start(1000000);
}

void loop() {
  uint64_t value = slave.sdoUpload64(0x5000, 0x00);

  uint32_t highPart = (uint32_t)(value >> 32);
  uint32_t lowPart = (uint32_t)(value & 0xFFFFFFFF);
```

```
  Serial.print("Value: ");
  Serial.print(highPart, HEX);
  Serial.print(lowPart, HEX);
  Serial.println();

  delay(1000);
}
```

getODlist()

說明

取得該 EtherCAT 從站在物件字典 (Object Dictionary) 中現有的物件列表。

語法

```
int getODlist(
    uint16_t *  list,
    uint32_t    list_size,
    uint32_t *  abortcode = NULL,
    uint32_t    timeout_us = 2000000
);
```

參數

- *[out] uint16_t *list*
  用來儲存物件列表的資料緩衝區。
- *[in] uint32_t list_size*
  資料緩衝區可儲存的物件數量。
- *[out] uint32_t *abortcode*
  用來儲存 SDO Abort Code 的變數指標。
- *[in] uint32_t timeout_us*
  逾時時間（單位為微秒）。

傳回值

傳回物件列表中的物件數量。若傳回值小於 0，表示發生錯誤，並回傳對應錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

uint16_t ODlist[1024];
int rc;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
```

```
  master.start(1000000);
}

void loop() {
  rc = sizeof(ODlist) / sizeof(ODlist[0]);
  rc = slave.getODlist(ODlist, rc);
  for (int i = 0; i < rc; i++) {
    Serial.print("Index ");
    Serial.println(ODlist[i]);
  }
}
```

getObjectDescription()

說明

取得 EtherCAT 從站中指定索引物件的物件描述資訊。

語法

```
int getObjectDescription(
    uint16_t    od_index,
    uint16_t *  datatype,
    uint8_t  *  max_od_subindex,
    uint8_t  *  objcode,
    char     *  objname,
    size_t      objname_size,
    uint32_t *  abortcode = NULL,
    uint32_t    timeout_us = 2000000
);
```

```
int getObjectDescription(
    uint16_t    od_index,
    uint16_t &  datatype,
    uint8_t  &  max_od_subindex,
    uint8_t  &  objcode,
    char     *  objname,
    size_t      objname_size,
    uint32_t *  abortcode = NULL,
    uint32_t    timeout_us = 2000000
);
```

參數

- *[in] uint16_t od_index*
  要查詢的物件索引。
- *[out] uint16_t *datatype*
  用來儲存資料型別的變數。請參考 資料型別。
- *[out] uint8_t *max_od_subindex*
  物件的最大子索引數量。
- *[out] uint8_t *objcode*
  物件類型代碼：
  7: Variable

8: Array

9: Record

- *[out] char objname*
  用來儲存物件名稱的緩衝區。
- *[in] size_t objname_size*
  物件名稱緩衝區的大小。
- *[out] uint32_t *abortcode*
  用來儲存 SDO Abort Code 的變數指標。
- *[in] uint32_t timeout_us*
  逾時時間（單位為微秒），預設為 2000000 µs（2 秒）。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

uint16_t DataType;
uint8_t MaxSubindex, ObjectCode;
char ObjectName[64];

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  slave.getObjectDescription(0x1C12, DataType, MaxSubindex, ObjectCode,
ObjectName, sizeof(ObjectName));
  Serial.print("Data Type: ");
  Serial.println(DataType);
  Serial.print("Object Code: ");
  Serial.println(ObjectCode);
  Serial.print("Max Subindex: ");
  Serial.println(MaxSubindex);
  Serial.print("Object Name: ");
  Serial.println(ObjectName);
```

```
}

void loop() {
  // ...
}
```

getEntryDescription()

說明

取得指定索引與子索引之物件項目的描述資訊，適用於 EtherCAT 從站。

語法

```
int getEntryDescription(
    uint16_t    od_index,
    uint8_t     od_subindex,
    uint8_t  *  valueinfo,
    uint16_t *  datatype,
    uint16_t *  bitlength,
    uint16_t *  objaccess,
    char     *  entryname,
    size_t      entryname_size,
    uint32_t *  abortcode = NULL,
    uint32_t    timeout_us = 2000000
);
```

```
int getEntryDescription(
    uint16_t    od_index,
    uint8_t     od_subindex,
    uint8_t  &  valueinfo,
    uint16_t &  datatype,
    uint16_t &  bitlength,
    uint16_t &  objaccess,
    char     *  entryname,
    size_t      entryname_size,
    uint32_t *  abortcode = NULL,
    uint32_t    timeout_us = 2000000
);
```

參數

- *[in] uint16_t od_index*
  物件的索引值。
- *[in] uint8_t od_subindex*
  物件的子索引值。
- *[in] uint8_t valueinfo*

用來儲存值資訊的變數，回應中包含下列位元意義：

Bit 0: reserved

Bit 1: reserved

Bit 2: reserved

Bit 3: unit type

Bit 4: default value

Bit 5: minimum value

Bit 6: maximum value

***NOTE**：此參數僅用於取得回應中位數資訊的上下文。檢索單位類型、預設值、最小值和最大值元素的值的功能尚不支援。

- `[out] uint16_t datatype`
  用來儲存資料型別的變數。請參考 資料型別。
- `[out] uint16_t bitlength`
  物件的位元長度。
- `[out] uint16_t objaccess`
  存取屬性，對應位元意義如下：
  Bit 0: read access in Pre-Operational state
  Bit 1: read access in Safe-Operational state
  Bit 2: read access in Operational state
  Bit 3: write access in Pre-Operational state
  Bit 4: write access in Safe-Operational state
  Bit 5: write access in Operational state
  Bit 6: object is mappable in a RxPDO
  Bit 7: object is mappable in a TxPDO
  Bit 8: object can be used for backup
  Bit 9: object can be used for settings
  Bit 10-15: reserved
- `[out] char entryname`
  儲存物件項目名稱的緩衝區。
- `[in] size_t entryname_size`
  名稱緩衝區的大小。
- `[out] uint32_t abortcode`
  用來儲存 SDO Abort Code 的變數指標。
- `[in] uint32_t timeout_us`
  逾時時間 (單位：微秒)，預設為 2,000,000 微秒 (2 秒)。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

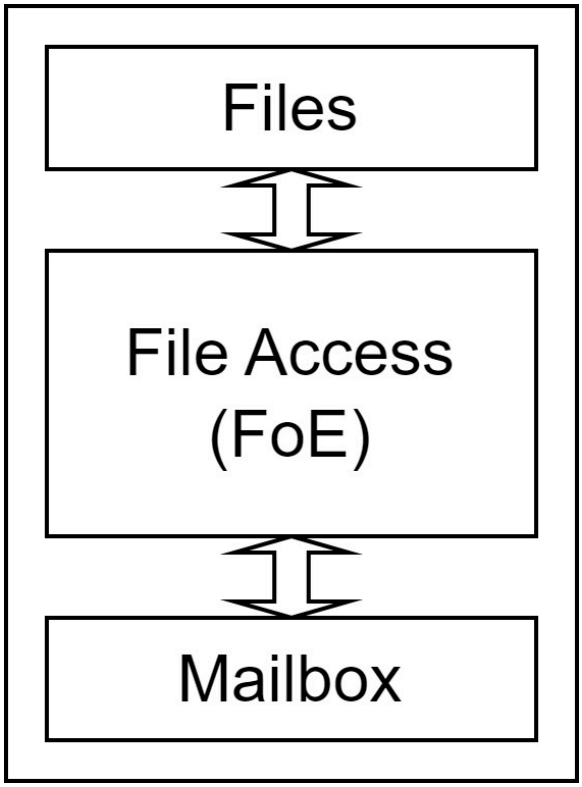
此函式必須在成功執行 EthercatMaster::begin() 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  uint16_t DataType, BitLength, ObjectAccess;
  uint8_t ValueInfo = 0;
  char EntryName[64];

  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);

  slave.getEntryDescription(0x1C12, 0x01, ValueInfo, DataType, BitLength,
ObjectAccess, EntryName, sizeof(EntryName));
  Serial.print("Data Type      : ");
  Serial.print(DataType, HEX);
  Serial.println("h");
  Serial.print("Bit Length     : ");
  Serial.print(BitLength, HEX);
  Serial.println("h");
  Serial.print("Object Access   : ");
  Serial.print(ObjectAccess, HEX);
  Serial.println("h");
  Serial.print("Entry Name     : ");
  Serial.println(EntryName);
}

void loop() {
  // Do nothing here
}
```

File over EtherCAT (FoE) 函式

File access over EtherCAT (FoE) 是 EtherCAT 協定的擴展功能之一，提供透過 EtherCAT 網路執行檔案傳輸的能力。它定義了一種標準方式，可將韌體或其他檔案下載至從站裝置，或從從站裝置上傳韌體或其他檔案。



函式：

- readFoE()
- writeFoE()

readFoE()

說明

透過 FoE 從 EtherCAT 從裝置中讀取檔案。

語法

```
int readFoE(char *filename, uint32_t password, void *data, uint32_t size,
uint32_t timeout_ms = 30000);
```

參數

- *[in] char *filename*
  欲讀取檔案的檔名。
- *[in] uint32_t password*
  32 位元密碼值。若為 0，表示不使用密碼。
- *[in] void *data*
  用來儲存檔案資料的緩衝區。
- *[in] uint32_t size*
  檔案資料緩衝區的大小。
- *[in] uint32_t timeout_ms*
  逾時時間 (單位：毫秒)，預設為 30,000 毫秒 (30 秒)。

傳回值

傳回實際讀取的檔案大小。若傳回值小於 0，表示發生錯誤，並傳回對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

#define MAX_FILE_DATA_SIZE  (2 * 1024 * 1024)

EthercatMaster master;
EthercatDevice_Generic slave;

char destination[] = {"destination.bin"};
char filename[] = {"firmware.bin"};
uint32_t password = 0;
uint8_t *filedata;
int filesize;
FILE *file;

void setup() {
```

```
  master.begin();
  slave.attach(0, master);

  filedata = (uint8_t *)malloc(MAX_FILE_DATA_SIZE);
  if (filedata != NULL) {
    filesize = slave.readFoE(filename, password, filedata,
MAX_FILE_DATA_SIZE);
    file = fopen(destination, "wb");
    if (file != NULL) {
      fwrite(filedata, sizeof(uint8_t), filesize, file);
      fclose(file);
    }
    free(filedata);
  }

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

writeFoE()

說明

透過 FoE 將檔案寫入 EtherCAT 從裝置。

語法

```
int writeFoE(char *filename, uint32_t password, void *data, uint32_t size,
uint32_t timeout_ms = 30000);
```

參數

- *[in] char *filename*
  欲寫入的檔案名稱。
- *[in] uint32_t password*
  32 位元密碼值。若為 0，表示不使用密碼。
- *[in] void *data*
  欲寫入檔案的資料緩衝區。
- *[in] uint32_t size*
  欲寫入資料的緩衝區大小。
- *[in] uint32_t timeout_ms*
  逾時時間 (單位：毫秒)，預設為 30,000 毫秒 (30 秒)。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

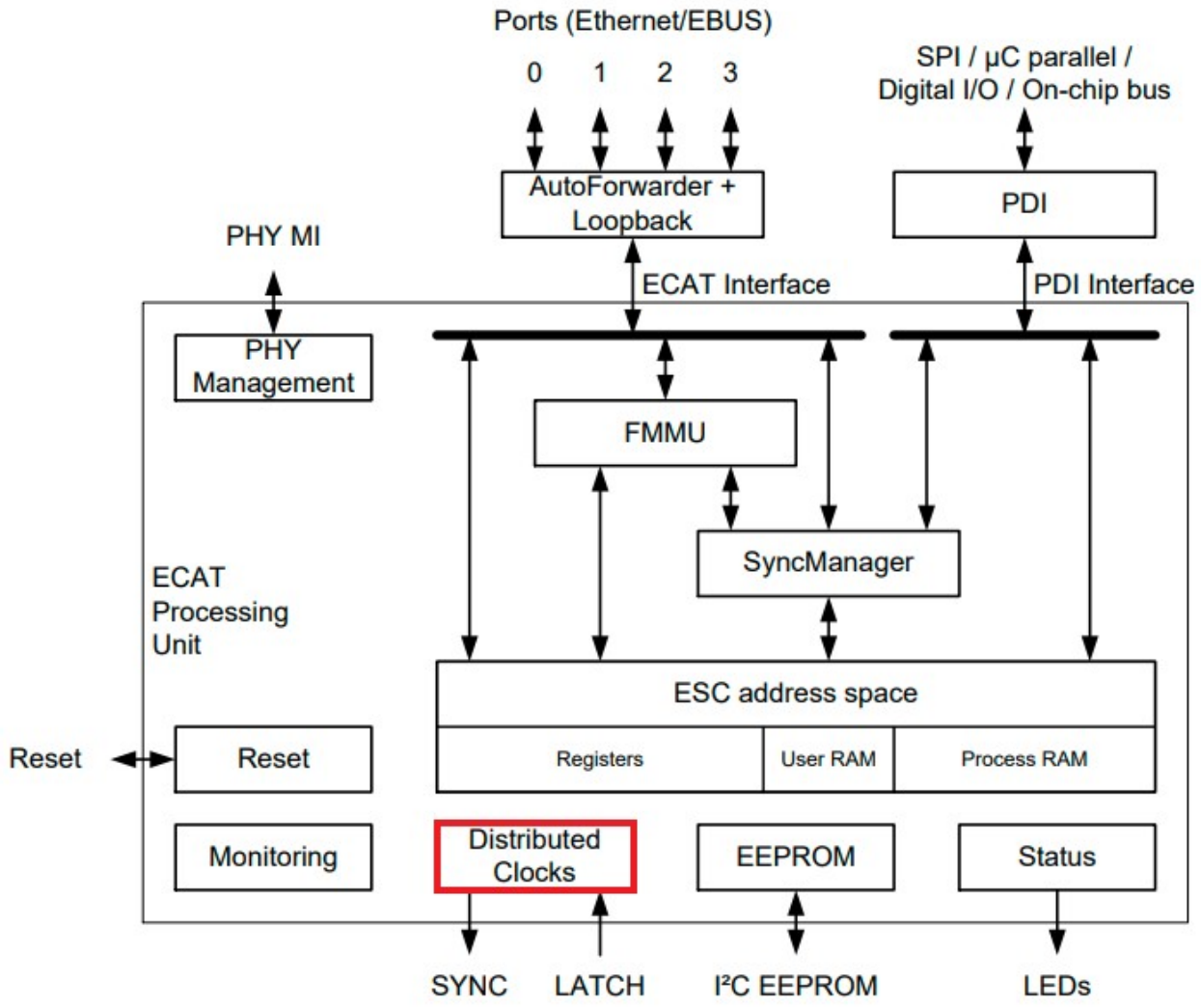
```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

char source[] = {"source.bin"};
char filename[] = {"firmware.bin"};
uint32_t password = 0;
uint8_t *filedata;
int filesize;
FILE *file;

void setup() {
  master.begin();
  slave.attach(0, master);
```

```
  file = fopen(source, "rb");
  if (file != NULL) {
    fseek(file, 0, SEEK_END);
    filesize = ftell(file);
    fseek(file, 0, SEEK_SET);
    filedata = (uint8_t *)malloc(filesize);
    if (filedata != NULL) {
      fread(filedata, sizeof(uint8_t), filesize, file);
      slave.writeFoE(filename, password, filedata, filesize);
      free(filedata);
    }
    fclose(file);
  }
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

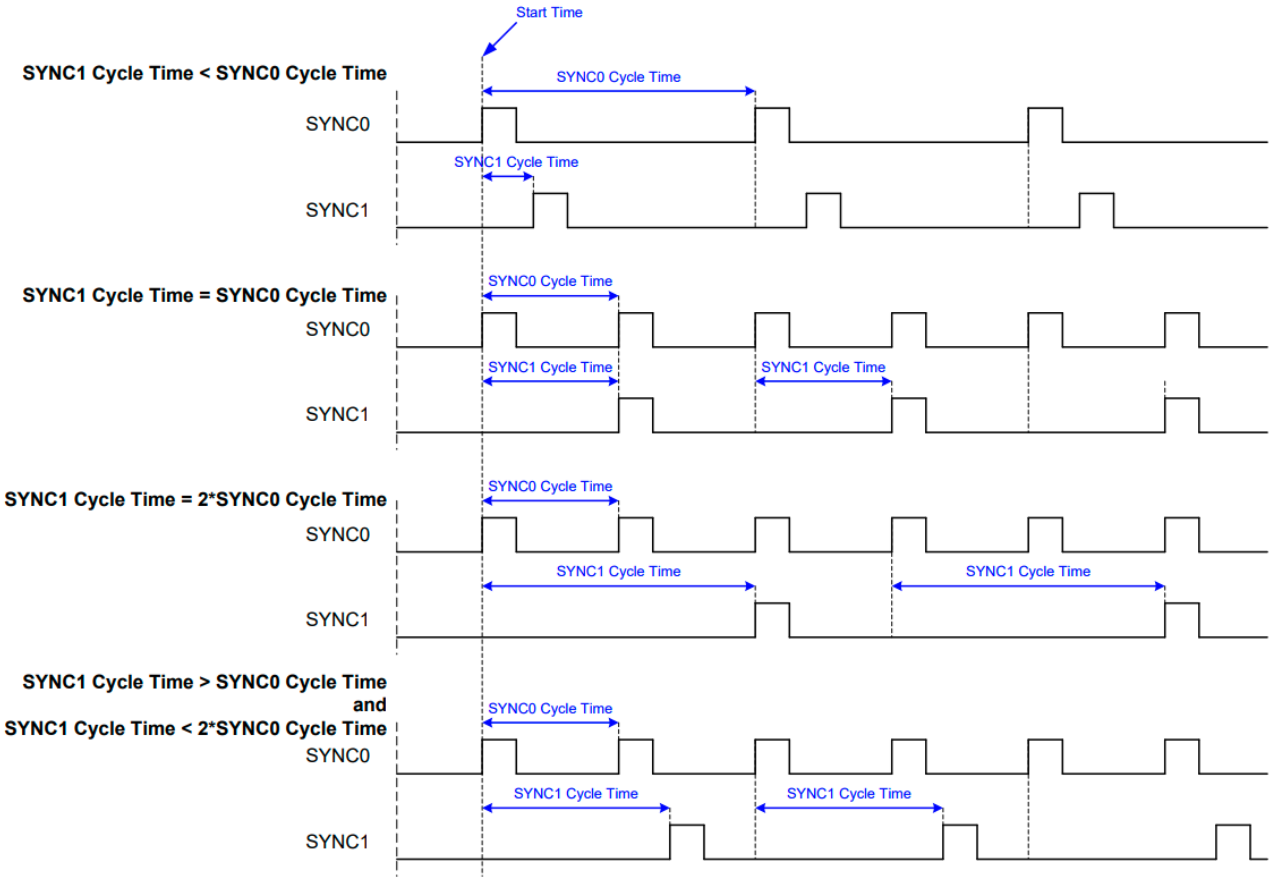
Distributed Clock (DC) 函式

Distributed Clocks (DC) 是 EtherCAT 從站裝置控制器 (ESC) 中的一項重要功能單元，負責在 EtherCAT 網路中實作時間同步機制。它確保所有從站裝置能根據統一的時間參考進行時鐘同步，進而確保整個系統的時間一致性。

為了實現系統同步，所有從站裝置都會與一個「參考時鐘 (Reference Clock)」同步。通常，在主站之後的第一個具備 DC 功能的 ESC 會成為此段 EtherCAT 網路的參考時鐘，並持有系統時間 (System Time)。其他從站與主站的 DC 時鐘都將與此參考時鐘同步，並會考考慮傳播延遲、本地時鐘來源漂移和本地時鐘偏移等因素進行校正。
ESC 可產生 **同步訊號 (SyncSignal)** 以讓本地應用程式與 EtherCAT 系統時間同步。這些同步訊號可直接用於中斷觸發、數位輸出更新或數位輸入採樣。另外，鎖存訊號 (LatchSignal) 也可基於 EtherCAT 系統時間進行時間戳記。



DC 單元支援產生基本同步訊號 **SYNC0** 和從屬同步訊號 **SYNC1**。第二個同步訊號 (SYNC1) 取決於 SYNC0，它可以在 SYNC0 脈衝之後以預先定義的延遲產生。如果 SYNC1 週期時間大於 SYNC0 週期時間，則會如下產生：當達到起始時間週期操作時，會產生 SYNC0 脈衝。 SYNC1 脈衝在 SYNC0 脈衝之後產生，延遲時間為 SYNC1 週期時間。當下一個 SYNC0 脈衝產生時，加上 SYNC1 週期時間，就會產生下一個 SYNC1 脈衝。下圖顯示了一些範例配置：



如需詳細資訊，請參閱 ESC Hardware Data Sheet Section I。

函式：

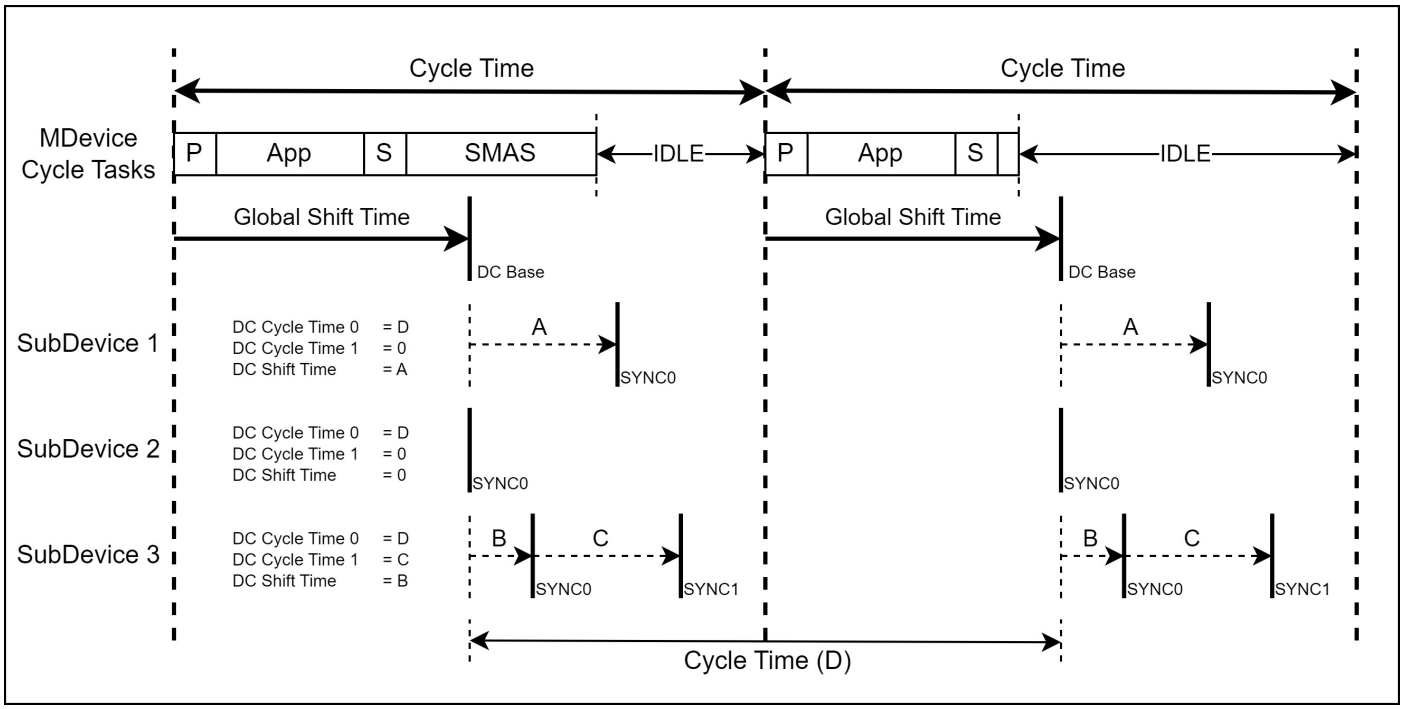
- `setDc()`

setDc()

說明

設定 EtherCAT 從裝置的 DC (Distributed Clocks) 參數。本函式可設定三個 DC 參數：

- ***DC Cycle Time 0*** 用來設定 SYNC0 訊號的週期時間，通常與 EtherCAT 通訊週期對齊。
- ***DC Cycle Time 1*** 用來設定 SYNC1 訊號的週期時間，表示在 SYNC0 脈衝之後的延遲時間（選填）。
- ***DC Shift Time*** 用來設定 SYNC0 訊號相對於 DC 基準時間的偏移量。



語法

```
int setDc(uint32_t cycletime0_ns, int32_t shifttime_ns = 0, uint32_t cycletime1_ns = 0);
```

參數

- *[in] uint32_t cycletime0_ns*
  DC SYNC0 的週期時間 (單位：奈秒)。
- *[in] int32_t shifttime_ns*
  DC SYNC0 的偏移時間 (單位：奈秒)。
- *[in] uint32_t cycletime1_ns*
  DC SYNC1 的週期時間 (單位：奈秒)。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後、呼叫 `EthercatMaster::start()` 之前執行。本函式為阻塞式，不可在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.setDc(1000000);
  master.start(1000000);
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

SII EEPROM 函式

EtherCAT 從站裝置控制器使用一個強制性的非揮發性記憶體 (NVRAM)，通常是一個具備 I²C 介面的序列 EEPROM，用於儲存 EtherCAT 從站裝置資訊 (ESI)。這些資訊包含廠商 ID、產品代碼、Mailbox 設定、FMMU、PDO 等。根據 ESC 的規格，支援的 EEPROM 容量從 1 Kbit 到 4 Mbit 不等。
ESC 組態區 (EEPROM 字位址 0 到 7) 會在開機或重置後由 ESC 自動讀取，該區包含 PDI 設定、DC 設定以及已設定的從站別名。ESC 組態資料的一致性由檢查碼 (Checksum) 確保。
如需詳細資料，請參閱 ESC Hardware Data Sheet Section I。

函式：

- readSII()
- readSII8()
- readSII16()
- readSII32()
- writeSII()
- writeSII8()
- writeSII16()
- writeSII32()

readSII()

說明

從 EtherCAT 從站裝置的 SII EEPROM 指定位移位置讀取多個位元組的資料。

語法

```
int readSII(uint32_t offset, void *data, size_t len, uint32_t timeout_ms = 500);
```

參數

- *[in] uint32_t offset*
  EtherCAT 從站裝置之 SII EEPROM 的位移位置。
- *[in] void *data*
  用於讀取 SII EEPROM 的資料緩衝區。
- *[in] size_t len*
  讀取資料緩衝區的大小 (位元組)。
- *[in] uint32_t timeout_ms*
  逾時時間 (單位：毫秒)。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

uint8_t buffer[4];

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
}

void loop() {
  slave.readSII(0, buffer, 4);  // Read SII data
  Serial.print("Buffer: ");
```

```
  Serial.print(buffer[0], HEX);
  Serial.print(", ");
  Serial.print(buffer[1], HEX);
  Serial.print(", ");
  Serial.print(buffer[2], HEX);
  Serial.print(", ");
  Serial.println(buffer[3], HEX);
  delay(1000);
}
```

readSII8()

說明

從 EtherCAT 從站裝置的 SII EEPROM 指定位移位置讀取 8-bit 的資料。

語法

```
uint8_t readSII8(uint32_t offset, uint32_t timeout_ms = 500);
```

參數

- *[in] uint32_t offset*
  EtherCAT 從站裝置之 SII EEPROM 的位移位置。
- *[in] uint32_t timeout_ms*
  逾時時間 (單位：毫秒)。

傳回值

回傳 SII EEPROM 中的 8 位元資料。

備註

此函式必須在成功執行 EthercatMaster::begin() 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
    int readSII8;
    Serial.begin(115200);
    while (!Serial);
    master.begin();
    slave.attach(0, master);
    master.start();
    readSII8 = slave.readSII8(0);
    Serial.println(readSII8);
}
void loop() {
    // ...
}
```

readSII16()

說明

從 EtherCAT 從站裝置的 SII EEPROM 指定位移位置讀取 16-bit 的資料。

語法

```
uint16_t readSII16(uint32_t offset, uint32_t timeout_ms = 500);
```

參數

- *[in] uint32_t offset*
  EtherCAT 從站裝置之 SII EEPROM 的位移位置。
- *[in] uint32_t timeout_ms*
  逾時時間 (單位：毫秒)。

傳回值

回傳 SII EEPROM 中的 16 位元資料。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
}

void loop() {
  Serial.print("Value: ");
  Serial.println(slave.readSII16(0), HEX);
  delay(1000);
}
```

readSII32()

說明

從 EtherCAT 從站裝置的 SII EEPROM 指定位移位置讀取 32-bit 的資料。

語法

```
uint32_t readSII32(uint32_t offset, uint32_t timeout_ms = 500);
```

參數

- *[in] uint32_t offset*
  EtherCAT 從站裝置之 SII EEPROM 的位移位置。
- *[in] uint32_t timeout_ms*
  逾時時間 (單位：毫秒)。

傳回值

回傳 SII EEPROM 中的 32 位元資料。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
}

void loop() {
  Serial.print("Value: ");
  Serial.println(slave.readSII32(0), HEX);
  delay(1000);
}
```

writeSII()

說明

將多個位元組的資料寫入 EtherCAT 從站裝置之 SII EEPROM 的指定位移位置。

語法

```
int writeSII(uint32_t offset, void *data, size_t len, uint32_t timeout_ms = 500);
```

參數

- *[in] uint32_t offset*
  EtherCAT 從站裝置之 SII EEPROM 的位移位置。
- *[in] void *data*
  用於寫入 SII EEPROM 的資料緩衝區。
- *[in] size_t len*
  寫入資料緩衝區的大小。
- *[in] uint32_t timeout_ms*
  逾時時間 (毫秒為單位)。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

uint8_t buffer[4] = {0x00, 0x55, 0xAA, 0xFF};

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.writeSII(64, buffer, 4);
}

void loop() {

}
```

writeSII8()

說明

將 8-bit 整數值寫入 EtherCAT 從站裝置的 SII EEPROM 中指定的位移位置。

語法

```
int writeSII8(uint32_t offset, uint8_t value, uint32_t timeout_ms = 500);
```

參數

- *[in] uint32_t offset*
  EtherCAT 從站裝置之 SII EEPROM 的位移位置。
- *[in] uint8_t value*
  欲寫入至 SII EEPROM 的 8-bit 整數值。
- *[in] uint32_t timeout_ms*
  逾時時間 (毫秒為單位)。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.writeSII8(64, 0x55);
}

void loop() {
  // ...
}
```

writeSII16()

說明

將 16-bit 整數值寫入 EtherCAT 從站裝置的 SII EEPROM 中指定的位移位置。

語法

```
int writeSII16(uint32_t offset, uint16_t value, uint32_t timeout_ms = 500);
```

參數

- *[in] uint32_t offset*
  EtherCAT 從站裝置之 SII EEPROM 的位移位置。
- *[in] uint16_t value*
  欲寫入至 SII EEPROM 的 16-bit 整數值。
- *[in] uint32_t timeout_ms*
  逾時時間 (毫秒為單位)。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.writeSII16(64, 0x5555);
}

void loop() {
  // ...
}
```

writeSII32()

說明

將 32-bit 整數值寫入 EtherCAT 從站裝置的 SII EEPROM 中指定的位移位置。

語法

```
int writeSII32(uint32_t offset, uint32_t value, uint32_t timeout_ms = 500);
```

參數

- *[in] uint32_t offset*
  EtherCAT 從站裝置之 SII EEPROM 的位移位置。
- *[in] uint32_t value*
  欲寫入至 SII EEPROM 的 32-bit 整數值。
- *[in] uint32_t timeout_ms*
  逾時時間 (毫秒為單位)。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

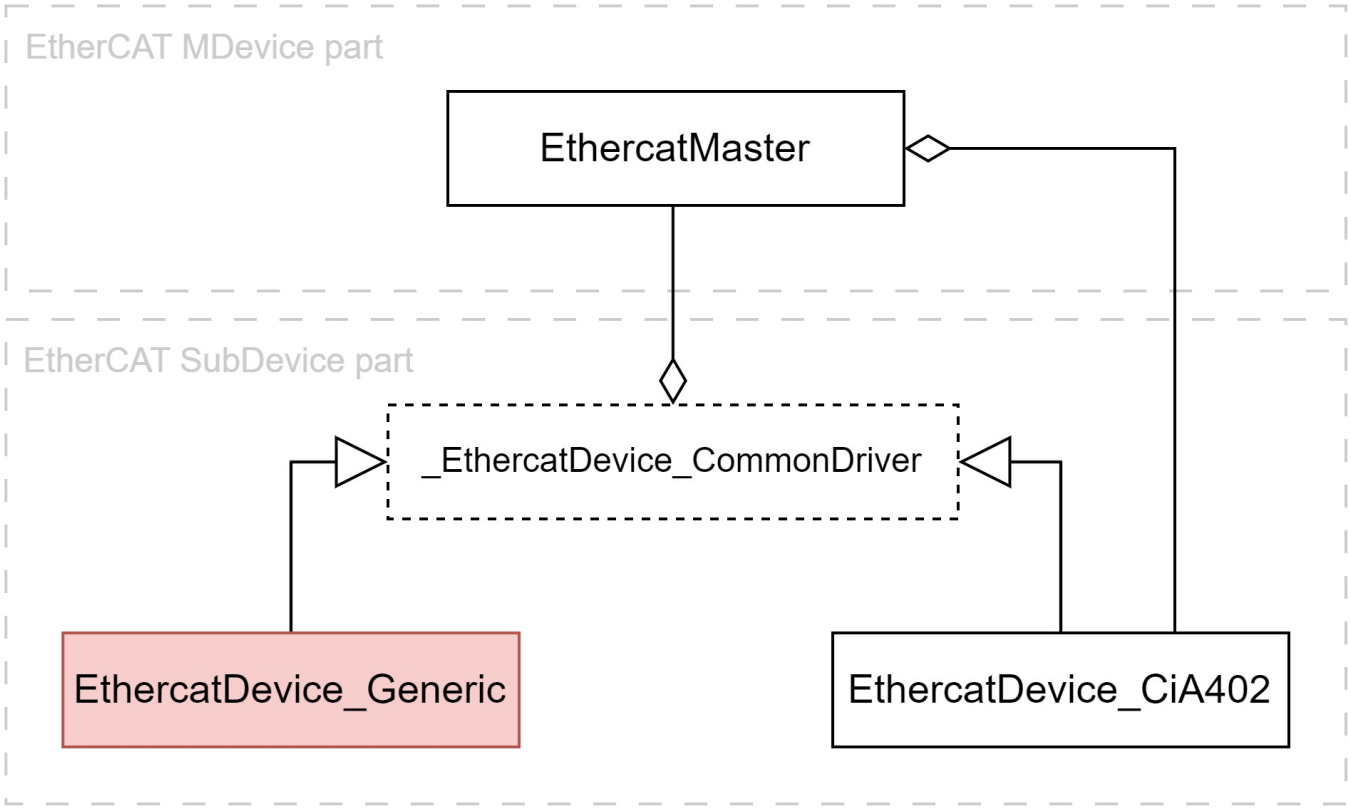
範例

```
#include "Ethercat.h"
EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.writeSII32(64, 0x55555555);
}

void loop() {
  // ...
}
```

2.2.2 EthercatDevice_Generic

EthercatDevice_Generic 是一個通用的 EtherCAT 從站裝置類別，可用於控制所有 EtherCAT 從站裝置，並提供存取從站裝置資訊、PDO、CoE、FoE、DC 等功能。

EthercatDevice_Generic 的類別關係如下圖所示：



- *EthercatDevice_Generic* 繼承自 *_EthercatDevice_CommonDriver*

基礎類別：

- _EthercatDevice_CommonDriver

函式分類：

- 初始化

函式：

函式名稱	說明	Callback 中呼叫
初始化相關函式		
attach()	初始化此 EtherCAT 從站裝置類別的物件	
detach()	解除初始化此 EtherCAT 從站裝置類別的物件	

初始化函式

EthercatDevice_Generic 類別的初始化相關函式。

函式：

- attach()
- detach()

attach()

說明

初始化此 EtherCAT 從站類別的物件，並根據網路上從站的 ID，將其附加至 EthercatMaster 類別的物件中。

語法

```
int attach(uint16_t slave_id, EthercatMaster *master, EthercatAttachMode mode = ECAT_SLAVE_NO);
int attach(uint16_t slave_id, EthercatMaster &master, EthercatAttachMode mode = ECAT_SLAVE_NO);
```

參數

- *[in] uint16_t slave_id*
  EtherCAT 網路上的從站的 ID。此 ID 的定義依據 mode 參數而定。
- *[in] EthercatMaster *master*
  要附加的 EthercatMaster 類別物件。
- *[in] EthercatAttachMode mode*
  slave_id 的定義方式：
  - ***ECAT_SLAVE_NO***
    EtherCAT 從站在網路上的順序編號，例如 0 表示第一個從站，1 表示第二個，依此類推。
  - ***ECAT_ALIAS_ADDRESS***
    從站的 alias address，該地址定義於其 SII EEPROM 的位元組偏移 8 處。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。

警告：禁止在 callback 函式中呼叫此函式。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
    master.begin();
    slave.attach(0, master);
    master.start();
}
```

```
void loop() {
  //...

}
```

detach()

說明

解除初始化此 EtherCAT 從站類別的物件，並將其自 EthercatMaster 類別的物件中解除連結。

語法

```
int detach();
```

參數

無。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatDevice Generic::attach()` 之後才能呼叫。

警告：禁止在 callback 函式中呼叫此函式。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  master.begin();
  slave.attach(0, master);

  delay(3000);
  slave.detach();
  master.end();
}

void loop() {
  // do something here.

}
```

2.2.3 EthercatDevice_CiA402

EthercatDevice_CiA402 是一個通用的 CiA 402 EtherCAT 從站裝置類別，旨在控制任何支援 CiA 402 標準的 EtherCAT 伺服驅動器。

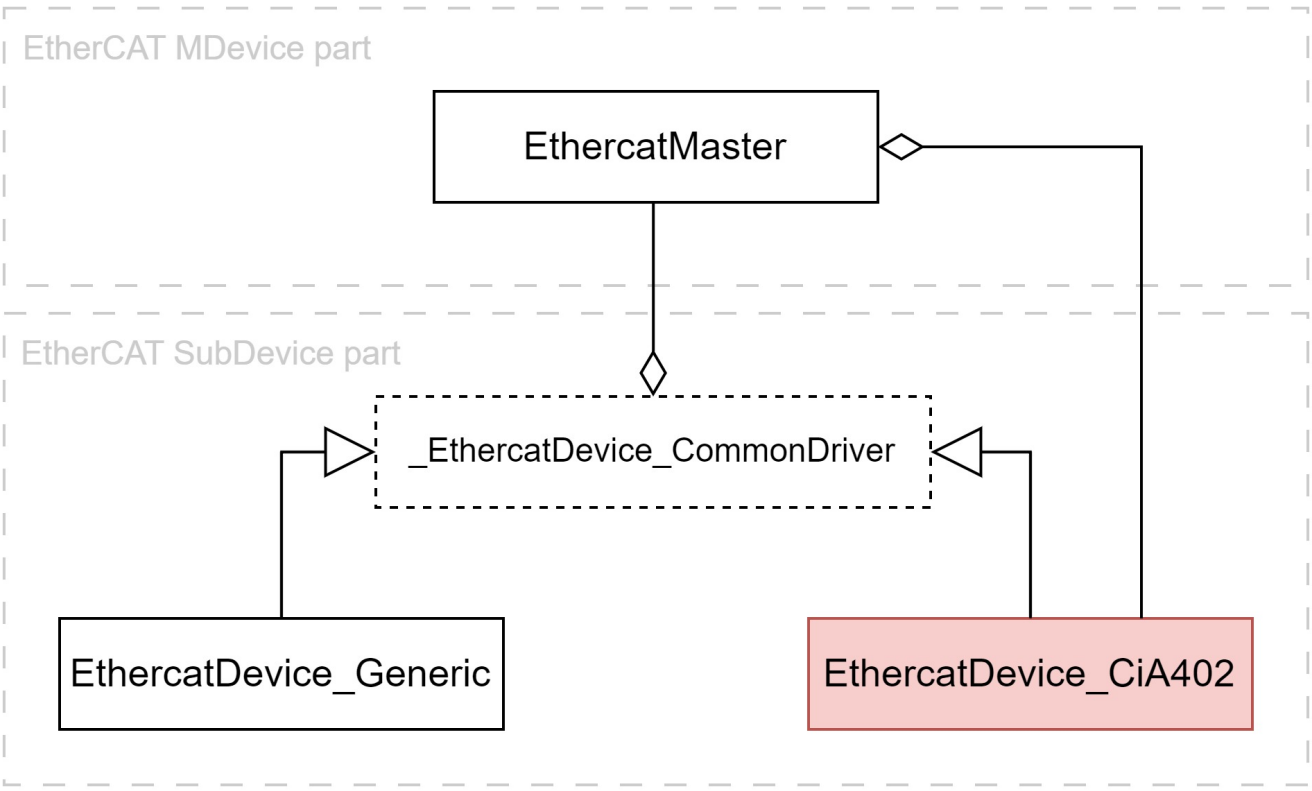
它提供了常用的 CiA 402 物件的存取函式以及多種 CiA 402 操作模式和功能組的操作函式，包括：

- *操作模式 (Operation Modes)*
  - Profile Position (pp)
  - Profile Velocity (pv)
  - Profile Torque (tq)
  - Homing (hm)
  - Cyclic Synchronous Position (csp)
  - Cyclic Synchronous Velocity (csv)
  - Cyclic Synchronous Torque (cst)
- *功能群組 (Function Groups)*
  - Touch Probe

如需更詳細的 CiA 402 說明，請參考以下文件：

- *CiA Draft Standard 402: CANopen device profile drives and motion control*
- *ETG.6010 Implementation Directive for CiA402 Drive Profile*
- *User manual for the currently used CiA 402 drive device*

EthercatDevice_CiA402 的類別關係如下圖所示：



- EthercatDevice_CiA402 繼承自 _EthercatDevice_CommonDriver.

基礎類別：

- _EthercatDevice_CommonDriver

函式：

函式名稱	說明	Callback 中呼叫
Initialization-related functions		
`attach()`	Initialize the object of this EtherCAT SubDevice class.	
`detach()`	Deinitialize the object of this EtherCAT SubDevice class.	
`isStepper()`	Check if the EtherCAT SubDevice is a stepper motor drive.	O
Control-related functions		
`getCiA402Mode()`	Get the current mode of operation. (6061_h)	O[1]
`setCiA402Mode()`	Switch the mode of operation. (6060_h, 6061_h, 6502_h)	O[1, 2]
`getCiA402State()`	Get the current CiA 402 state. (6041_h)	O
`setCiA402State()`	Switch the CiA 402 state. (6040h, 6041_h)	O[2]
`enable()`	Enable the drive function and power on the motor. (6040_h, 6041_h)	O[2]
`disable()`	Disable the drive function and power off the motor. (6040_h, 6041_h)	O[2]

Operation-related functions		
`setTargetPosition()`	Set the target position. ($607A_h$)	O[1]
`setTargetVelocity()`	Set the target velocity. ($60FF_h$)	O[1]
`setTargetTorque()`	Set the target torque. (6071_h)	O[1]
`setProfileAcceleration()`	Set the profile acceleration. (6083_h)	O[1]
`setProfileDeceleration()`	Set the profile deceleration. (6084_h)	O[1]
`setMaxAcceleration()`	Set the max acceleration. ($60C5_h$)	O[1]
`setMaxDeceleration()`	Set the max deceleration. ($60C6_h$)	O[1]
`setMaxProfileVelocity()`	Set the max profile velocity. ($607F_h$)	O[1]
`setMotionProfileType()`	Set the motion profile type. (6086_h)	O[1]
`setPositionWindow()`	Set the position window. (6067_h)	O[1]
`setPositionWindowTime()`	Set the position window time. (6068_h)	O[1]
`setPositionOffset()`	Set the position offset. ($60B0_h$)	O[1]
`setSoftwarePositionLimit()`	Set the software position limit. ($607D_h$)	O[1]
`setFollowingErrorWindow()`	Set the following error window. (6065_h)	O[1]
`setPositionPolarity()`	Set the position polarity. ($607E_h$)	O[1]
`setVelocityWindow()`	Set the velocity window. ($606D_h$)	O[1]
`setVelocityWindowTime()`	Set the velocity window time. ($606E_h$)	O[1]
`setVelocityThreshold()`	Set the velocity threshold. ($606F_h$)	O[1]
`setVelocityOffset()`	Set the velocity offset. ($60B1_h$)	O[1]
`setMaxMotorSpeed()`	Set the max motor speed. (6080_h)	O[1]
`setVelocityPolarity()`	Set the velocity polarity. ($607E_h$)	O[1]
`setTorqueOffset()`	Set the torque offset. ($60B2_h$)	O[1]
`setMaxTorque()`	Set the max torque. (6072_h)	O[1]
`setPositiveTorqueLimit()`	Set the positive torque limit. ($60E0_h$)	O[1]
`setNegativeTorqueLimit()`	Set the negative torque limit. ($60E1_h$)	O[1]
`setQuickStopDeceleration()`	Set the quick stop deceleration. (6085_h)	O[1]
`setQuickStopOptionCode()`	Set the quick stop option code. ($605A_h$)	
`setShutdownOptionCode()`	Set the shutdown option code. ($605B_h$)	
`setDisableOperationOptionCode()`	Set the disable operation option code. ($605C_h$)	
`setHaltOptionCode()`	Set the halt option code. ($605D_h$)	
`setFaultReactionOptionCode()`	Set the fault reaction option code. ($605E_h$)	
`getErrorCode()`	Get the error code. ($603F_h$)	O[1]
`getSupportedDriveModes()`	Get the supported drive modes. (6502_h)	O[1]
`getMotorResolution()`	Get the motor resolution. ($60EF_h$)	
`getPositionActualValue()`	Get the position actual value. (6064_h)	O[1]
`getVelocityActualValue()`	Get the velocity actual value. ($606C_h$)	O[1]
`getTorqueActualValue()`	Get the torque actual value. (6077_h)	O[1]
`getCurrentActualValue()`	Get the current actual value. (6078_h)	O[1]
`getPositionDemandValue()`	Get the position demand value. (6062_h)	O[1]
`getPositionDemandInternalValue()`	Get the position demand internal value. ($60FC_h$)	O[1]

`getPositionActualInternalValue()`	Get the position actual internal value. (6063_h)	O[1]
`getAdditionalPositionActualValue()`	Get the additional position actual value. ($60E4_h$)	O[1]
`getFollowingErrorActualValue()`	Get the following error actual value. ($60F4_h$)	O[1]
`getVelocityDemandValue()`	Get the velocity demand value. ($606B_h$)	O[1]
`getTorqueDemandValue()`	Get the torque demand value. (6074_h)	O[1]
`getDigitalInputs()`	Get the digital inputs. ($60FD_h$)	O[1]
Profile Position mode (pp) related functions		
`pp_SetVelocity()`	Set the profile velocity. (6081_h)	
`pp_SetAcceleration()`	Set the profile acceleration. (6083_h)	
`pp_SetDeceleration()`	Set the profile deceleration. (6084_h)	
`pp_SetMotionProfileType()`	Set the motion profile type. (6086_h)	
`pp_Run()`	Move to the target position. (6040_h, 6041_h, $607A_h$)	
`pp_IsTargetReached()`	Check if the target position has been reached. (6041_h)	O
`pp_CheckFollowingError()`	Check if the following error occurs. (6041_h)	O
`pp_Halt()`	Pause the current operation. (6040h, 6041_h)	
`pp_Resume()`	Resume the paused operation. (6040_h, 6041_h)	
Profile Velocity mode (pv) related functions		
`pv_SetAcceleration()`	Set the profile acceleration. (6083_h)	
`pv_SetDeceleration()`	Set the profile deceleration. (6084_h)	
`pv_SetMotionProfileType()`	Set the motion profile type. (6086_h)	
`pv_Run()`	Move at a target velocity continuously. (6041_h, $60FF_h$)	
`pv_IsTargetReached()`	Check if the target velocity has been reached. (6041_h)	O
`pv_CheckZeroSpeed()`	Check if the speed is zero. (6041_h)	O
`pv_CheckMaxSlippageError()`	Check if the maximum slippage error occurs. (6041_h)	O
`pv_Halt()`	Pause the current operation. (6040_h, 6041_h)	
`pv_Resume()`	Resume the paused operation. (6040_h, 6041_h)	
Profile Torque mode (tq) related functions		
`tq_SetTorqueSlope()`	Set the torque slope. (6087_h)	
`tq_SetTorqueProfileType()`	Set the torque profile type. (6088_h)	
`tq_SetMotorRatedCurrent()`	Set the motor rated current. (6075_h)	
`tq_SetMotorRatedTorque()`	Set the motor rated torque. (6076_h)	
`tq_Run()`	Drive continuously at the target torque. (6041_h, 6071_h)	
`tq_IsTargetReached()`	Check if the target torque has been reached. (6041_h)	O
`tq_Halt()`	Pause the current operation. (6040_h, 6041_h)	
`tq_Resume()`	Resume the paused operation. (6040_h, 6041_h)	
Homing mode (hm) related functions		
`hm_SetHomeOffset()`	Set the home offset. ($607C_h$)	
`hm_SetHomingMethod()`	Set the homing method. (6098_h)	
`hm_SetHomingSpeeds()`	Set the homing speeds. (6099_h)	
`hm_SetHomingAcceleration()`	Set the homing acceleration. ($609A_h$)	
`hm_Run()`	Initiate a homing operation. (6040_h, 6041_h)	

`hm_IsAttained()`	Check the status of the homing operation. (6041_h)	O
`hm_Stop()`	Stop the homing operation. (6040_h, 6041_h)	
Function Group "Touch Probe" related functions		
`enableTouchProbe1()`	Enable the touch probe 1. ($60B8_h$, $60B9_h$, $60D0_h$)	
`enableTouchProbe2()`	Enable the touch probe 2. ($60B8_h$, $60B9_h$, $60D0_h$)	
`disableTouchProbe1()`	Disable the touch probe 1. ($60B8_h$, $60B9_h$)	
`disableTouchProbe2()`	Disable the touch probe 2. ($60B8_h$, $60B9_h$)	
`isTouchProbe1ValueReady()`	Check if a positive or negative edge has occurred on the touch probe 1 signal. ($60B9_h$)	O[1]
`isTouchProbe2ValueReady()`	Check if a positive or negative edge has occurred on the touch probe 2 signal. ($60B9_h$)	O[1]
`readTouchProbe1Value()`	Read the touch probe position 1. ($60BA_h$, $60BB_h$)	O[1]
`readTouchProbe2Value()`	Read the touch probe position 2. ($60BC_h$, $60BD_h$)	O[1]
Low-level functions for mode-specific flow control		
`setHaltBit()`	Set the halt bit in the controlword. (6040_h)	O
`isTargetReached()`	Check if the target has reached. (6041_h)	O
`setModeSpecificBit4()`	Set the bit 4 in the controlword. (6040_h)	O
`setModeSpecificBit5()`	Set the bit 5 in the controlword. (6040_h)	O
`setModeSpecificBit6()`	Set the bit 6 in the controlword. (6040_h)	O
`checkModeSpecificBit12()`	Check the value of bit 12 in the statusword. (6041_h)	O
`checkModeSpecificBit13()`	Check the value of bit 13 in the statusword. (6041_h)	O

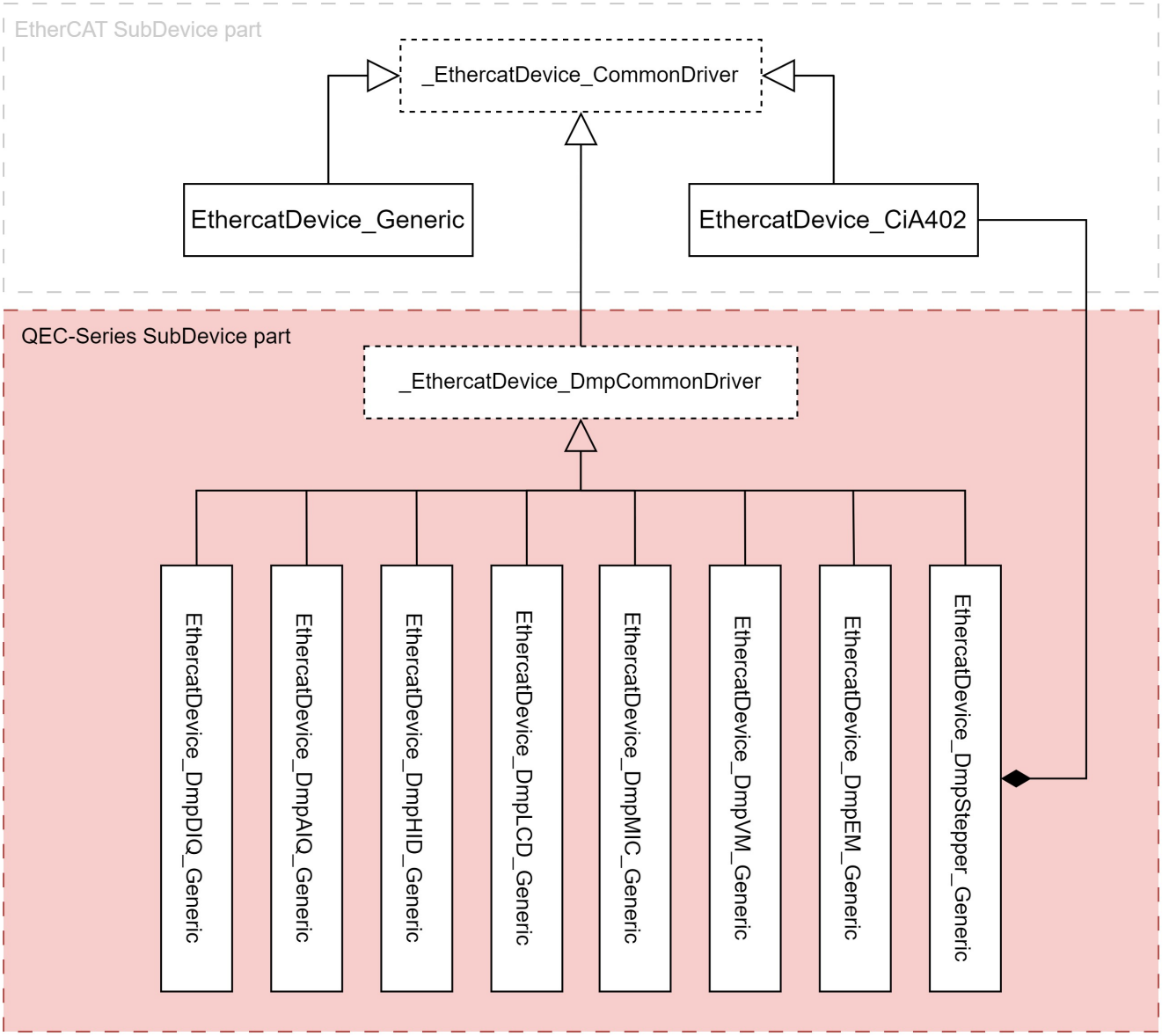
- **Note 1**：若相關物件已對應至 PDO，此函式可於 Callback 函式中使用。
- **Note 2**：當此函式於 Callback 函式中使用時，將忽略 timeout 參數，且不會等待實際值與設定值一致。

如需更詳細的資訊與 API 函式使用說明，請參閱 EtherCAT CiA402 APIs。

2.3 QEC-Series SubDevice

QEC-Series SubDevice 部分提供了專為 ICOP 的 QEC 系列從站裝置設計的專屬函式，讓使用者能以更直觀且簡潔的方式進行程式撰寫。

QEC-Series SubDevice 和 EtherCAT SubDevice 部分之間的主要類別關係是關聯，QEC-Series SubDevice 依賴 EtherCAT SubDevice 部分。如下圖所示，**_EthercatDevice_DmpCommonDriver** 與 **_EthercatDevice_CommonDriver** 之間存在關聯關係。

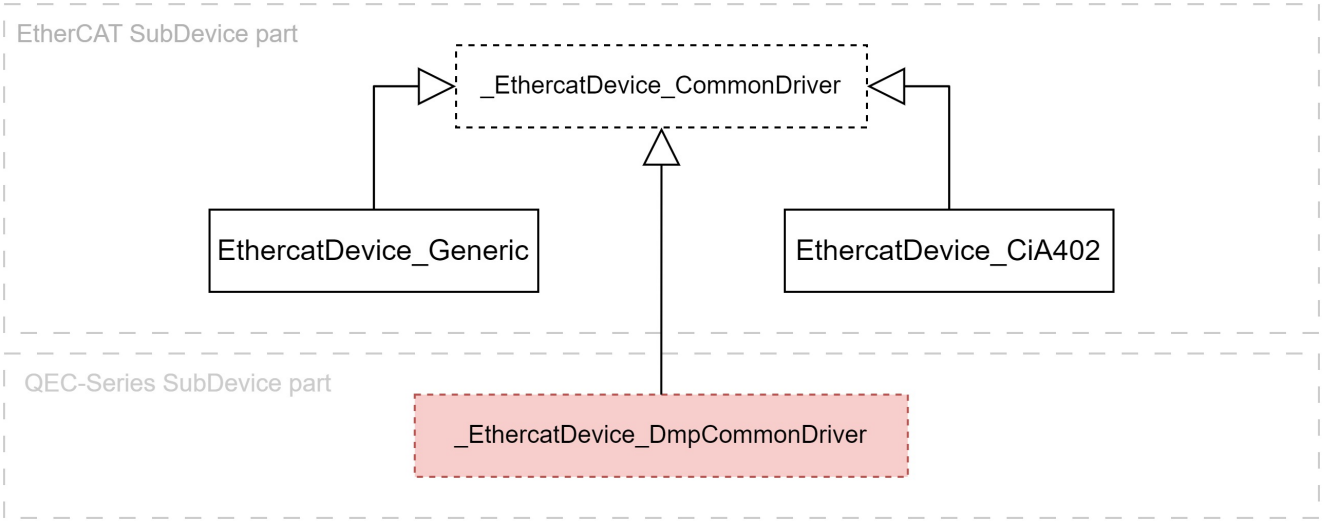


類別

- _EthercatDevice_DmpCommonDriver
- EthercatDevice_DmpDIQ_Generic
- EthercatDevice_DmpAIQ_Generic
- EthercatDevice_DmpCIQ_Generic
- EthercatDevice_DmpHID_Generic
- EthercatDevice_DmpLCD_Generic
- EthercatDevice_DmpMIC_Generic
- EthercatDevice_DmpVM_Generic
- EthercatDevice_DmpEM_Generic
- EthercatDevice_DmpStepper_Generic

2.3.1 _EthercatDevice_DmpCommonDriver

_EthercatDevice_DmpCommonDriver 是一個抽象類別，提供 ICOP 所開發的 EtherCAT 從站裝置專屬功能的存取函式。這些功能包括：系統監控 (溫度、電壓、電流)、訂單資訊、平均故障間隔時間 (MTBF) 等。
_EthercatDevice_DmpCommonDriver 的類別關係如下圖所示：



- *_EthercatDevice_DmpCommonDriver* 繼承自 *_EthercatDevice_CommonDriver*.

警告：禁止使用此類別建立物件。

基礎類別

- _EthercatDevice_CommonDriver

函式分類：

- 系統監控函式
- MTBF 函式
- 訂單資訊函式

函式：

函式名稱	說明	Callback 中呼叫
系統監控相關函式		
getSystemTemperature()	取得系統溫度	O[1,2]
getSystemPowerVoltage()	取得系統電壓	O[1,2]
getSystemPowerCurrent()	取得系統電流	O[1,2]
getPeripheralPowerVoltage()	取得周邊電壓	O[1,2]

getPeripheralPowerCurrent()	取得周邊電流	O[1,2]
tryToGetSystemTemperature()	以非阻塞方式嘗試取得系統溫度	O
tryToGetSystemPowerVoltage()	以非阻塞方式嘗試取得系統電壓	O
tryToGetSystemPowerCurrent()	以非阻塞方式嘗試取得系統電流	O
tryToGetPeripheralPowerVoltage()	以非阻塞方式嘗試取得周邊電壓	O
tryToGetPeripheralPowerCurrent()	以非阻塞方式嘗試取得周邊電流	O
MTBF 相關函式		
getWorkingHours()	取得累積運行時間	O[1]
getBootTimes()	取得開機次數	O[1]
tryToGetWorkingHours()	以非阻塞方式嘗試取得累積運行時間	O
tryToGetBootTimes()	以非阻塞方式嘗試取得開機次數	O
Order information related functions		
getCustomer()	取得客戶資訊	
getOrderNumber()	取得訂單號碼	
getInvoiceNumber()	取得發票號碼	
getDeliveryDate()	取得出貨日期	

- **Note 1**：是否能於回呼函式中使用，需視函式使用方式與對應條件而定。
- **Note 2**：此函式包含浮點數運算，不可在禁用 FPU (浮點運算單元) 的 Callback 函式中呼叫。更多關於禁用 FPU 的說明，請參閱 Callback 函式 章節。

系統監控函式

所有搭載 MCU 的 QEC 系列 EtherCAT 從站裝置皆提供 CoE 物件，用於獲取系統監控資訊，包括：**系統溫度、系統電壓、系統電流、周邊電壓、周邊電流**。因此，本函式庫提供對應的系統監控函式，讓使用者能即時掌握系統狀態，並據以評估是否有潛在異常或故障徵兆。

函式：

- getSystemTemperature()
- getSystemPowerVoltage()
- getSystemPowerCurrent()
- getPeripheralPowerVoltage()
- getPeripheralPowerCurrent()
- tryToGetSystemTemperature()
- tryToGetSystemPowerVoltage()
- tryToGetSystemPowerCurrent()
- tryToGetPeripheralPowerVoltage()
- tryToGetPeripheralPowerCurrent()

getSystemTemperature()

說明

取得系統溫度。

語法

```
double getSystemTemperature();
```

參數

無。

傳回值

返回系統溫度。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 後才能呼叫。

此函式可以是阻塞式或非阻塞式。

若為阻塞式，則不能在 callback 函式中呼叫。 若為非阻塞式，僅能在支援 FPU 的 callback 函式中呼叫，並必須搭配 `tryToGetSystemTemperature()` 使用。關於支援 FPU 的 callback 函式，請參考 Callback 函式。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11DT0H slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);

  Serial.print("System Temperature: ");
  Serial.println(slave.getSystemTemperature());
}
void loop() {
  // ...
}
```

getSystemPowerVoltage()

說明

取得系統電壓 (Vs)。

語法

```
double getSystemPowerVoltage();
```

參數

無。

傳回值

返回系統電壓。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 後才能呼叫。

此函式可以是阻塞式或非阻塞式。

若為阻塞式，則不能在 callback 函式中呼叫。 若為非阻塞式，僅能在支援 FPU 的 callback 函式中呼叫，並必須搭配 `tryToGetSystemPowerVoltage()` 使用。關於支援 FPU 的 callback 函式，請參考 Callback 函式。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11DT0H slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);

  Serial.print("System Voltage: ");
  Serial.println(slave.getSystemPowerVoltage());
}
void loop() {
  // ...
}
```

getSystemPowerCurrent()

說明

取得系統電流 (Is)。

語法

`double getSystemPowerCurrent();`

參數

無。

傳回值

返回系統電流。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 後才能呼叫。

此函式可以是阻塞式或非阻塞式。

若為阻塞式，則不能在 callback 函式中呼叫。 若為非阻塞式，僅能在支援 FPU 的 callback 函式中呼叫，並必須搭配 tryToGetSystemPowerCurrent() 使用。關於支援 FPU 的 callback 函式，請參考 Callback 函式。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11DT0H slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);

  Serial.print("System Current: ");
  Serial.println(slave.getSystemPowerCurrent());
}
void loop() {
  // ...
}
```

getPeripheralPowerVoltage()

說明

取得外部設備電壓 (Vp)。

語法

```
double getPeripheralPowerVoltage();
```

參數

無。

傳回值

返回外部設備電壓。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 後才能呼叫。

此函式可以是阻塞式或非阻塞式。

若為阻塞式，則不能在 callback 函式中呼叫。 若為非阻塞式，僅能在支援 FPU 的 callback 函式中呼叫，並必須搭配 tryToGetPeripheralPowerVoltage() 使用。關於支援 FPU 的 callback 函式，請參考 Callback 函式。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11DT0H slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);

  Serial.print("Peripheral Voltage: ");
  Serial.println(slave.getPeripheralPowerVoltage());
}
void loop() {
  // ...
}
```

getPeripheralPowerCurrent()

說明

取得外部設備電流 (Ip)。

語法

```
double getPeripheralPowerCurrent();
```

參數

無。

傳回值

返回外部設備電流。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 後才能呼叫。

此函式可以是阻塞式或非阻塞式。

若為阻塞式，則不能在 callback 函式中呼叫。 若為非阻塞式，僅能在支援 FPU 的 callback 函式中呼叫，並必須搭配 tryToGetPeripheralPowerCurrent() 使用。關於支援 FPU 的 callback 函式，請參考 Callback 函式。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11DT0H slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);

  Serial.print("Peripheral Current: ");
  Serial.println(slave.getPeripheralPowerCurrent());
}
void loop() {
  // ...
}
```

tryToGetSystemTemperature()

說明

以非阻塞方式嘗試取得系統溫度。若回傳 true，表示讀取程序已完成，必須立即呼叫 `getSystemTemperature()` 以取得數值；若回傳 false，則表示讀取程序尚未完成。

語法

```
bool tryToGetSystemTemperature();
```

參數

無。

傳回值

回傳一個布林值，表示非阻塞讀取程序是否已完成：

- true：已完成
- false：進行中

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，且可在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11DT0H slave;
double SystemTemperature;

void CyclicCallback() {
  if (slave.tryToGetSystemTemperature())
    SystemTemperature = slave.getSystemTemperature();
}

void setup() {
  Serial.begin(115200);
  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  Serial.print("System Temperature: "); Serial.println(SystemTemperature);
}
```

tryToGetSystemPowerVoltage()

說明

以非阻塞方式嘗試取得系統電壓。若回傳 true，表示讀取程序已完成，必須立即呼叫 `getSystemPowerVoltage()` 以取得數值；若回傳 false，則表示讀取程序尚未完成。

語法

```
bool tryToGetSystemPowerVoltage();
```

參數

無。

傳回值

回傳一個布林值，表示非阻塞讀取程序是否已完成：

- true：已完成
- false：進行中

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，且可在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11DT0H slave;
double SystemVoltage;

void CyclicCallback() {
  if (slave.tryToGetSystemPowerVoltage())
    SystemVoltage = slave.getSystemPowerVoltage();
}

void setup() {
  Serial.begin(115200);
  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  Serial.print("System Voltage: ");  Serial.println(SystemVoltage);
}
```

tryToGetSystemPowerCurrent()

說明

以非阻塞方式嘗試取得系統電流。若回傳 true，表示讀取程序已完成，必須立即呼叫 `getSystemPowerCurrent()` 以取得數值；若回傳 false，則表示讀取程序尚未完成。

語法

```
bool tryToGetSystemPowerCurrent();
```

參數

無。

傳回值

回傳一個布林值，表示非阻塞讀取程序是否已完成：

- true：已完成
- false：進行中

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，且可在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11DT0H slave;
double SystemCurrent;

void CyclicCallback() {
  if (slave.tryToGetSystemPowerCurrent())
    SystemCurrent = slave.getSystemPowerCurrent();
}

void setup() {
  Serial.begin(115200);
  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}
void loop() {
  Serial.print("System Current: ");  Serial.println(SystemCurrent);
}
```

tryToGetPeripheralPowerVoltage()

說明

以非阻塞方式嘗試取得外部電壓。若回傳 true，表示讀取程序已完成，必須立即呼叫 `getPeripheralPowerVoltage()` 以取得數值；若回傳 false，則表示讀取程序尚未完成。

語法

```
bool tryToGetPeripheralPowerVoltage();
```

參數

無。

傳回值

回傳一個布林值，表示非阻塞讀取程序是否已完成：

- true：已完成
- false：進行中

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，且可在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11DT0H slave;
double PeripheralVoltage;

void CyclicCallback() {
  if (slave.tryToGetPeripheralPowerVoltage())
    PeripheralVoltage = slave.getPeripheralPowerVoltage();
}

void setup() {
  Serial.begin(115200);
  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}
void loop() {
  Serial.print("Peripheral Voltage: ");  Serial.println(PeripheralVoltage);
}
```

tryToGetPeripheralPowerCurrent()

說明

以非阻塞方式嘗試取得外部電流。若回傳 true，表示讀取程序已完成，必須立即呼叫 `getPeripheralPowerCurrent()` 以取得數值；若回傳 false，則表示讀取程序尚未完成。

語法

```
bool tryToGetPeripheralPowerCurrent();
```

參數

無。

傳回值

回傳一個布林值，表示非阻塞讀取程序是否已完成：

- true：已完成
- false：進行中

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，且可在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11DT0H slave;
double PeripheralCurrent;

void CyclicCallback() {
  if (slave.tryToGetPeripheralPowerCurrent())
    PeripheralCurrent = slave.getPeripheralPowerCurrent();
}

void setup() {
  Serial.begin(115200);
  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}
void loop() {
  Serial.print("Peripheral Current: "); Serial.println(PeripheralCurrent);
}
```

MTBF 函式

MTBF 是「平均故障間隔時間」(Mean Time Between Failures) 的縮寫，是一項衡量系統或元件可靠性的指標，用來評估兩次故障之間的平均運行時間。其計算方式為：將總運行時間除以期間內發生的故障次數。所得結果為一個平均值，可用於估算系統或元件的預期使用壽命。MTBF 是一項實用的指標，可用來追蹤系統與元件的可靠性，並有助於辨識潛在的設計缺陷或製造瑕疵。它也可用來制定預防性維護的時程規劃。

所有搭載 MCU 的 QEC 系列 EtherCAT 從站裝置皆提供 CoE 物件來獲取 MTBF 相關資訊。因此，本函式庫提供相關函式，讓使用者可以取得這些 MTBF 資訊，並提供給製造商用於評估與判斷裝置的壽命與故障狀況。

函式：

- getWorkingHours()
- getBootTimes()
- tryToGetWorkingHours()
- tryToGetBootTimes()

getWorkingHours()

說明

取得目前的工作時數。

語法

```
int getWorkingHours();
```

參數

無。

傳回值

回傳目前的工作時數。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 後才能呼叫。

此函式可以是阻塞式或非阻塞式。

若為阻塞式，則不能在 callback 函式中呼叫。 若為非阻塞式，則可在 callback 函式中呼叫，並必須搭配 `tryToGetWorkingHours()` 使用。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11DT0H slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);

  Serial.print("Working Hours: ");
  Serial.println(slave.getWorkingHours());
}

void loop() {
  // ...
}
```

getBootTimes()

說明

取得目前的開機次數。

語法

```
int getBootTimes();
```

參數

無。

傳回值

回傳目前的開機次數。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 後才能呼叫。

此函式可以是阻塞式或非阻塞式。

若為阻塞式，則不能在 callback 函式中呼叫。 若為非阻塞式，則可在 callback 函式中呼叫，並必須搭配 `tryToGetBootTimes()` 使用。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11DT0H slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);

  Serial.print("Boot Times: ");
  Serial.println(slave.getBootTimes());
}

void loop() {
  // ...
}
```

tryToGetWorkingHours()

說明

以非阻塞方式嘗試取得工作時數。若回傳 true，表示讀取程序已完成，此時應立即呼叫 `getWorkingHours()` 以取得數值；若回傳 false，則表示讀取程序尚未完成。

語法

```
bool tryToGetWorkingHours();
```

參數

無。

傳回值

回傳一個布林值，表示非阻塞讀取程序是否已完成：

- true：已完成
- false：進行中

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，且可在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11DT0H slave;
int WorkingHours;

void CyclicCallback() {
  if (slave.tryToGetWorkingHours())
    WorkingHours = slave.getWorkingHours();
}

void setup() {
  Serial.begin(115200);
  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  Serial.print("Working Hours: ");  Serial.println(WorkingHours);
}
```

tryToGetBootTimes()

說明

以非阻塞方式嘗試取得開機次數。若回傳 true，表示讀取程序已完成，此時應立即呼叫 `getBootTimes()` 以取得數值；若回傳 false，則表示讀取程序尚未完成。

語法

```
bool tryToGetBootTimes();
```

參數

無。

傳回值

回傳一個布林值，表示非阻塞讀取程序是否已完成：

- true：已完成
- false：進行中

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，且可在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11DT0H slave;
int BootTimes;

void CyclicCallback() {
  if (slave.tryToGetBootTimes())
    BootTimes = slave.getBootTimes();
}

void setup() {
  Serial.begin(115200);
  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  Serial.print("Boot Times: "); Serial.println(BootTimes);
}
```

訂單資訊函式

所有搭載 MCU 的 QEC 系列 EtherCAT 從站裝置皆提供 CoE 物件，用於取得客戶訂單相關資訊，這些資訊在出貨前即已預先寫入裝置中。
因此，本函式庫提供函式以讀取這些客戶訂單資訊，方便在需要時進行查詢。

函式：

- getCustomer()
- getOrderNumber()
- getInvoiceNumber()
- getDeliveryDate()

getCustomer()

說明

取得客戶資訊。

語法

```
char *getCustomer(char *buffer = NULL);
```

參數

- *[out] char *buffer*
  字串資料緩衝區，請確保陣列大小至少為 7。如果未提供此參數，將使用內部資料緩衝區。

傳回值

回傳指向客戶資訊字串的指標。

備註

此函式必須在成功執行 EthercatMaster::begin() 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11DT0H slave;
char Customer[7];

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);

  Serial.print("Customer: ");
  Serial.println(slave.getCustomer());
  Serial.print("Customer: ");
  Serial.println(slave.getCustomer(Customer));
  Serial.print("Customer: ");
  Serial.println(Customer);
}

void loop() {
  // ...
}
```

getOrderNumber()

說明

取得訂單編號。

語法

```
char *getOrderNumber(char *buffer = NULL);
```

參數

- *[out] char *buffer*
  字串資料緩衝區，請確保陣列大小至少為 9。如果未提供此參數，將使用內部資料緩衝區。

傳回值

回傳指向訂單編號字串的指標。

備註

此函式必須在成功執行 EthercatMaster::begin() 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11DT0H slave;
char OrderNumber[9];

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);

  Serial.print("Order Number: ");
  Serial.println(slave.getOrderNumber());
  Serial.print("Order Number: ");
  Serial.println(slave.getOrderNumber(OrderNumber));
  Serial.print("Order Number: ");
  Serial.println(OrderNumber);
}

void loop() {
  // ...
}
```

getInvoiceNumber()

說明

取得發票號碼。

語法

```
char *getInvoiceNumber(char *buffer = NULL);
```

參數

- *[out] char *buffer*
  字串資料緩衝區，請確保陣列大小至少為 12。若未提供此參數，將使用內部資料緩衝區。

傳回值

回傳指向發票號碼字串的指標。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11DT0H slave;
char InvoiceNumber[12];

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);

  Serial.print("Invoice Number: ");
  Serial.println(slave.getInvoiceNumber());
  Serial.print("Invoice Number: ");
  Serial.println(slave.getInvoiceNumber(InvoiceNumber));
  Serial.print("Invoice Number: ");
  Serial.println(InvoiceNumber);
}

void loop() {
  // ...
}
```

getDeliveryDate()

說明

取得出貨日期。

語法

```
char *getDeliveryDate(char *buffer = NULL);
```

參數

- *[out] char *buffer*
  字串資料緩衝區，請確保陣列大小至少為 5。若未提供此參數，將使用內部資料緩衝區。

傳回值

回傳指向出貨日期字串的指標。

備註

此函式必須在成功執行 EthercatMaster::begin() 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

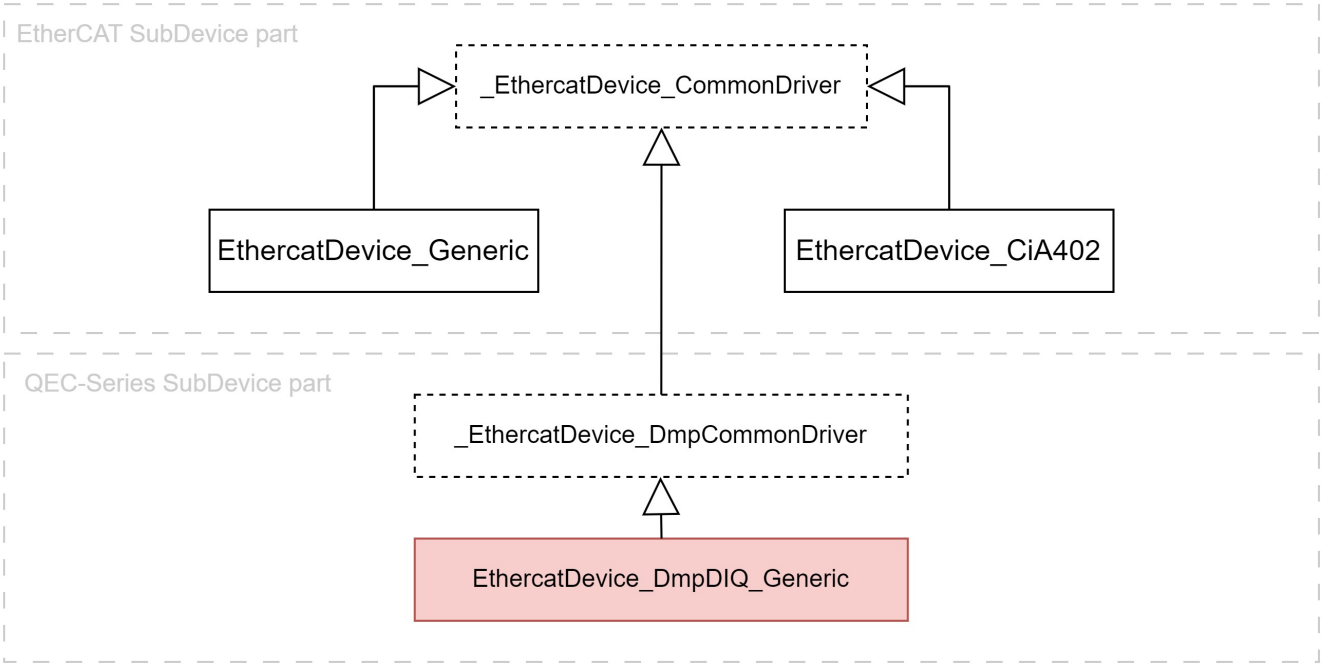
範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11DT0H slave;
char DeliveryDate[5];

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);

  Serial.print("Delivery Date: ");
  Serial.println(slave.getDeliveryDate());
  Serial.print("Delivery Date: ");
  Serial.println(slave.getDeliveryDate(DeliveryDate));
  Serial.print("Delivery Date: ");
  Serial.println(DeliveryDate);
}

void loop() {
  // ...
}
```

2.3.2 EthercatDevice_DmpDIQ_Generic

EthercatDevice_DmpDIQ_Generic 是由 ICOP 專為數位輸入/輸出（Digital I/O）EtherCAT 從站模組所開發的從站裝置類別。它提供了數位輸入、數位輸出等功能的 API。

EthercatDevice_DmpDIQ_Generic 的類別關係如以下圖所示：



- *EthercatDevice_DmpDIQ_Generic* 繼承自 *_EthercatDevice_DmpCommonDriver*.

基礎類別：

- _EthercatDevice_CommonDriver

衍生類別：

類別名稱	VID	PID	Inputs	Outputs	MCU	DC	斷線偵測
EthercatDevice_QECR00D0FS	0x00000bc3	0x0086d303	0	16	O		
EthercatDevice_QECR00D0FH	0x00000bc3	0x0086d30A	0	16	O	O	
EthercatDevice_QECR00D0TL	0x00000bc3	0x0086d327	0	32			
EthercatDevice_QECR00D0TH	0x00000bc3	0x0086d801	0	32	O	O	
EthercatDevice_QECR00DF0S	0x00000bc3	0x0086d30D	16	0	O		
EthercatDevice_QECR00DF0D	0x00000bc3	0x0086d300	16	0	O		O
EthercatDevice_QECR00DF0H	0x00000bc3	0x0086d30B	16	0	O	O	
EthercatDevice_QECR00DT0L	0x00000bc3	0x0086d323	32	0			
EthercatDevice_QECR00DT0H	0x00000bc3	0x0086d701	32	0	O	O	

EthercatDevice_QECR00D88S	0x00000bc3	0x0086d309	8	8	O		
EthercatDevice_QECR00D88D	0x00000bc3	0x0086d301	8	8	O		O
EthercatDevice_QECR00D88H	0x00000bc3	0x0086d30F	8	8	O	O	
EthercatDevice_QECR00DC4D	0x00000bc3	0x0086d304	12	4	O		O
EthercatDevice_QECR00D4CD	0x00000bc3	0x0086d302	4	12	O		O
EthercatDevice_QECR11D0FS	0x00000bc3	0x0086d0d4	0	16	O		
EthercatDevice_QECR11D0FH	0x00000bc3	0x0086d305	0	16	O	O	
EthercatDevice_QECR11D0TL	0x00000bc3	0x0086d324	0	32			
EthercatDevice_QECR11D0TH	0x00000bc3	0x0086d800	0	32	O	O	
EthercatDevice_QECR11DF0S	0x00000bc3	0x0086d30E	16	0	O		
EthercatDevice_QECR11DF0D	0x00000bc3	0x0086d0d2	16	0	O		O
EthercatDevice_QECR11DF0H	0x00000bc3	0x0086d306	16	0	O	O	
EthercatDevice_QECR11DT0L	0x00000bc3	0x0086d320	32	0			
EthercatDevice_QECR11DT0H	0x00000bc3	0x0086d700	32	0	O	O	
EthercatDevice_QECR11D88S	0x00000bc3	0x0086d0d5	8	8	O		
EthercatDevice_QECR11D88D	0x00000bc3	0x0086d307	8	8	O		O
EthercatDevice_QECR11D88H	0x00000bc3	0x0086d308	8	8	O	O	

函式分類：

- 初始化
- 數位 I/O
- 斷線偵測

函式：

函式名稱	說明	Callback 中呼叫
初始化相關函式		
attach()	初始化此 EtherCAT 從站裝置類別的物件	
detach()	解除初始化此 EtherCAT 從站裝置類別的物件	
數位 I/O 存取函式		
digitalWrite()	寫入指定數位輸出腳位的狀態	O
digitalRead()	讀取指定數位輸入腳位的狀態	O
digitalWriteAll()	寫入所有腳位的數位輸出狀態	O
digitalReadAll()	讀取所有腳位的數位輸入狀態	O
斷線偵測函式		
startBrokenWireTest()	啟動斷線偵測測試	

初始化函式

適用於 EthercatDevice_DmpDIQ_Generic 類別的初始化相關函式。

函式：

- attach()
- detach()

attach()

說明

此函式的行為與 `EthercatDevice Generic::attach()` 相同。

詳細的說明、語法、參數與傳回值，請參閱該章節。

範例

適用於 EthercatDevice_QECR00D0FH。

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00D0FH slave;

void setup() {
    master.begin();
    slave.attach(0, master);
    master.start();
}
void loop() {
    // ...

}
```

detach()

說明

此函式的行為與 `EthercatDevice Generic::detach()` 相同。

詳細的說明、語法、參數與傳回值，請參閱該章節。

範例

適用於 EthercatDevice_QECR00D0FH。

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00D0FH slave;

void setup() {
  master.begin();
  slave.attach(0, master);

  delay(3000);

  slave.detach();
  master.end();
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

數位 I/O 函式

適用於 EthercatDevice_DmpDIQ_Generic 類別的數位輸入與數位輸出功能函式。

函式：

- digitalWrite()
- digitalRead()
- digitalWriteAll()
- digitalReadAll()

digitalWrite()

說明

將 HIGH 或 LOW 的數位值寫入指定的 EtherCAT 從站數位輸出腳位。

語法

```
int digitalWrite(int pin, uint8_t value);
```

參數

- *[in] int pin*
  EtherCAT 從站的數位輸出腳位編號。
- *[in] uint8_t value*
  數位邏輯電平，值為 HIGH 或 LOW。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00D0FH slave;

void setup() {
    master.begin();
    slave.attach(0, master);
    master.start();
}
void loop() {
    slave.digitalWrite(0, HIGH);
    delay(4000);
    slave.digitalWrite(0, LOW);
    delay(1000);
}
```

digitalRead()

說明

讀取指定 EtherCAT 從站數位輸入腳位的值。

語法

```
int digitalRead(int pin);
```

參數

- *[in] int pin*
  EtherCAT 從站的數位輸出腳位編號。

傳回值

傳回指定腳位的數位邏輯電平，值為 HIGH 或 LOW。

備註

本函式必須在成功執行 EthercatMaster::start() 並在 EthercatMaster::stop() 之前呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00D88D slave;

void setup() {
  Serial.begin(115200);
  master.begin();
  slave.attach(0, master);
  master.start();
}

void loop() {
  slave.digitalWrite(0, HIGH);
  delay(1000);
  Serial.print("DI0: "); Serial.println(slave.digitalRead(0));

  slave.digitalWrite(0, LOW);
  delay(1000);
  Serial.print("DI0: "); Serial.println(slave.digitalRead(0));
}
```

digitalWriteAll()

說明

同時寫入 EtherCAT 從站所有數位輸出腳位的狀態。

語法

```
int digitalWriteAll(uint32_t value);
```

參數

- *[in] uint32_t value*

  此參數為一個 32 位元的無符號整數，用以指定要寫入所有數位輸出腳位的值。每一個位元對應一個數位輸出腳位，對應如下：:

  - 第 0 位元對應數位輸出腳位 0。
  - 第 1 位元對應數位輸出腳位 1。
  - 依此類推至第 31 位元對應腳位 31。

  位元值為 1 表示將該腳位設為 HIGH，值為 0 則為 LOW。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00D88D slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  master.start();
}

void loop() {
  slave.digitalWriteAll(0x55555555);
  delay(1000);
  slave.digitalWriteAll(0xAAAAAAAA);
  delay(1000);
}
```

digitalReadAll()

說明

同時讀取 EtherCAT 從站所有數位輸入腳位的狀態。

語法

```
uint32_t digitalReadAll();
```

參數

無。

傳回值

傳回一個 32 位元的無符號整數，代表所有數位輸入腳位的組合狀態。傳回值中的每個位元對應一個數位輸入腳位，對應如下：

- 第 0 位元表示數位輸入腳位 0。
- 第 1 位元表示數位輸入腳位 1。
- 依此類推至第 31 位元表示腳位 31。

值為 1 表示對應的腳位目前為 HIGH，值為 0 則表示為 LOW。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"
EthercatMaster master;
EthercatDevice_QECR00D88D slave;

void setup() {
  Serial.begin(115200);
  master.begin();
  slave.attach(0, master);
  master.start();
}
void loop() {
  slave.digitalWriteAll(0x55555555);
  delay(1000);
  Serial.print("DI: "); Serial.println(slave.digitalReadAll());

  slave.digitalWriteAll(0xAAAAAAAA);
  delay(1000);
  Serial.print("DI: "); Serial.println(slave.digitalReadAll());
}
```

斷線偵測函式

適用於 EthercatDevice_DmpDIQ_Generic 類別的斷線偵測功能。

函式：

- startBrokenWireTest()

startBrokenWireTest()

說明

啟動指定 EtherCAT 從站輸入腳位的斷線檢測測試。

語法

```
uint16_t startBrokenWireTest(uint16_t bitMask, uint32_t timeout_ms = 1000);
```

參數

- *[in] uint16_t bitmask*

  此參數為 16 位元無符號整數，用以指定哪些輸入腳位要納入斷線檢測測試。bitMask 中的每一個位元對應到一個輸入腳位，對應如下：

  - 位元 0 表示數位輸入腳位 0。
  - 位元 1 表示數位輸入腳位 1。
  - 依此類推至位元 15 表示腳位 15。

  值為 1 表示該腳位將被包含在測試中，值為 0 則表示不包含。

  警告： 必須確保此參數中設定為 1 的位元對應的腳位在 EtherCAT 從站上確實存在，否則將導致測試無法執行。

- *[in] uint32_t timeout_ms*

  逾時時間 (毫秒)。

傳回值

傳回一個 16 位元無符號整數，代表斷線檢測測試的結果。傳回值中的每個位元對應一個參與測試的輸入腳位，解釋如下：

- 0：該腳位未參與測試或被偵測為斷線。
- 1：該腳位參與測試，且未偵測到斷線。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為阻塞式，不可於 callback 函式中呼叫。

範例

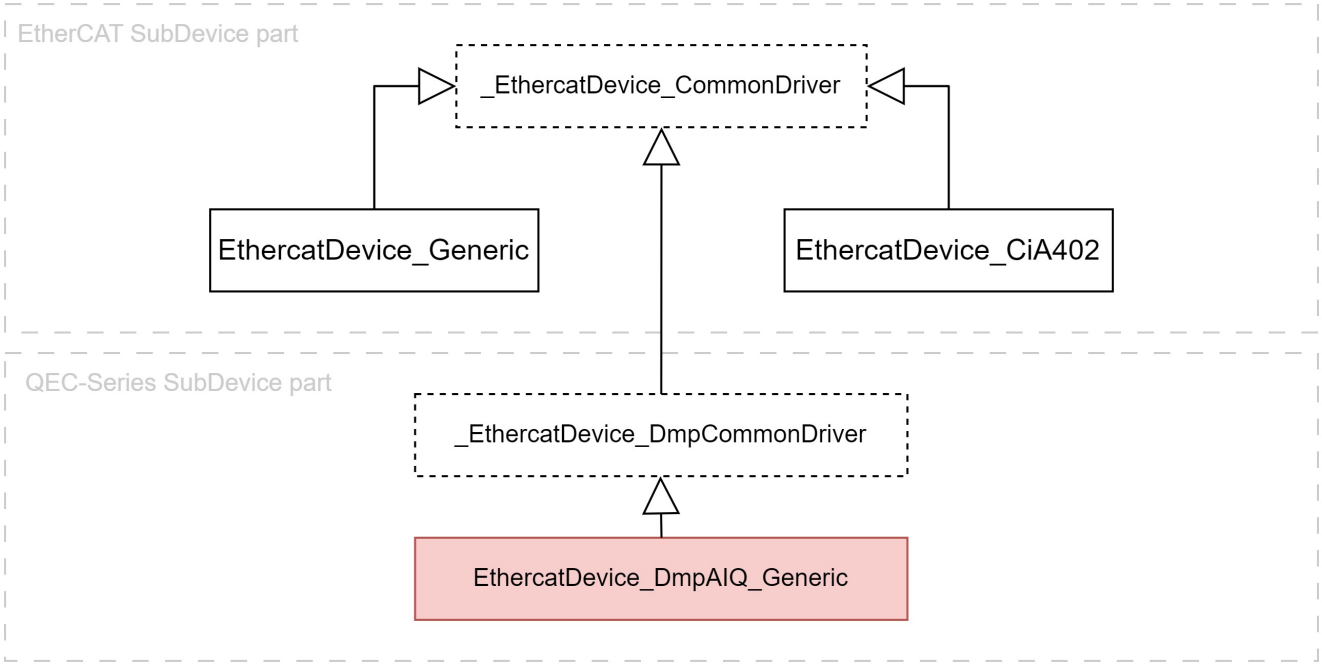
```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00D88D slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  master.start();
```

```
  Serial.print(slave.startBrokenWireTest(0xFF));
}

void loop() {
  // ...
}
```

2.3.3 EthercatDevice_DmpAIQ_Generic

EthercatDevice_DmpAIQ_Generic 是由 ICOP 專為類比輸入/輸出（Analog I/O）EtherCAT 從站模組所開發的從站裝置類別。它提供了類比輸入、類比輸出等功能的 API。

EthercatDevice_DmpAIQ_Generic 的類別關係如以下圖所示：



- *EthercatDevice_DmpAIQ_Generic* 繼承自 *_EthercatDevice_DmpCommonDriver*.

基礎類別：

- _EthercatDevice_CommonDriver

衍生類別：

類別名稱	Vendor ID	Product Code	Input Channels	Output Channels
EthercatDevice_QECR11A44S	0x00000bc3	0x0086d880	4	4

函式分類：

- 初始化
- 類比輸出存取函式
- 類比輸入存取函式

函式：

函式名稱	說明	Callback 中呼叫
初始化相關函式		
attach()	初始化此 EtherCAT 從站裝置類別的物件	
detach()	解除初始化此 EtherCAT 從站裝置類別的物件	
類比輸出存取函式		
analogWrite()	寫入指定類比輸出通道的值	O
voltageWrite()	寫入指定類比輸出通道的電壓值	O[1]
currentWrite()	寫入指定類比輸出通道的電流值	O[1]
類比輸入存取函式		
analogRead()	讀取指定類比輸入通道的值	O
voltageRead()	讀取指定類比輸入通道的電壓值	O[1]

- **Note 1**：此函式包含浮點數運算，因此不可在禁用 FPU（浮點運算單元）的 Callback 函式中呼叫。如需更多關於禁用 FPU Callback 函式的說明，請參閱 Callback 函式 章節。

初始化函式

適用於 EthercatDevice_DmpAIQ_Generic 類別的初始化相關函式。

函式：

- attach()
- detach()

attach()

說明

此函式的行為與 `EthercatDevice Generic::attach()` 相同。

詳細的說明、語法、參數與傳回值，請參閱該章節。

範例

適用於 EthercatDevice_QECR11A44S。

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11A44S slave;

void setup(void) {
    master.begin();
    slave.attach(0, master);
    master.start();
}
void loop() {
    // ...

}
```

detach()

說明

此函式的行為與 `EthercatDevice Generic::detach()` 相同。

詳細的說明、語法、參數與傳回值，請參閱該章節。

範例

適用於 EthercatDevice_QECR11A44S。

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11A44S slave;

void setup() {
  master.begin();
  slave.attach(0, master);

  delay(3000);

  slave.detach();
  master.end();
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

類比輸出存取函式

適用於 EthercatDevice_DmpAIQ_Generic 類別的類比輸出操作函式。

函式：

- analogWrite()
- voltageWrite()
- currentWrite()

analogWrite()

說明

將值寫入 EtherCAT 從站上指定的類比輸出通道。

語法

```
int analogWrite(int ch, int32_t value);
```

參數

- *[in] int ch*
  EtherCAT 從站上指定的類比輸出通道。
- *[in] int32_t value*
  要寫入的類比輸出值。此參數為 32 位元帶符號整數，根據該通道的模式設定，此值會線性對應為實際的輸出：
  - 電壓模式 (Voltage Mode)：對應關係為：0 映射為 0V、2^{31} 映射為 64V、-2^{31} 映射為 -64V。例如：2^{25} 對應約為 1V。
  - 電流模式 (Current Mode)：對應關係為：0 映射為 0A、2^{31} 映射為 16A、-2^{31} 映射為 -16A。例如：2^{27} 對應約為 1A。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11A44S slave;

int voltage = 5;

void setup() {
  master.begin();
  slave.attach(0, master);
  master.start();
}

void loop() {
  slave.analogWrite(0, voltage << 25);
  delay(1000);
  slave.analogWrite(0, 0);
```

```
  delay(1000);
  slave.analogWrite(0, -1 * (voltage << 25));
  delay(1000);
  slave.analogWrite(0, 0);
  delay(1000);
}
```

voltageWrite()

說明

將電壓值寫入 EtherCAT 從站上指定的類比輸出通道。本函式用於指定該通道已設定為電壓模式 (Voltage Mode)。

語法

```
int voltageWrite(int ch, double voltage);
```

參數

- *[in] int ch*
  EtherCAT 從站上指定的類比輸出通道。
- *[in] double voltage*
  所需輸出的電壓值，單位為伏特 (V)。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

此函式為非阻塞式，不可在未啟用 FPU 的 callback 函式中呼叫。有關 FPU 禁用之 callback 函式的更多詳細資訊，請參考 Callback 函式。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11A44S slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  master.start();
}

void loop() {
  slave.voltageWrite(0, 10.0);
  delay(1000);

  slave.voltageWrite(0, 0);
  delay(1000);

  slave.voltageWrite(0, -10.0);
```

```
  delay(1000);

  slave.voltageWrite(0, 0);
  delay(1000);
}
```

currentWrite()

說明

將電流值寫入 EtherCAT 從站上指定的類比輸出通道。本函式用於指定該通道已設定為電流模式 (Current Mode)。

語法

```
int currentWrite(int ch, double current);
```

參數

- *[in] int ch*
  EtherCAT 從站上指定的類比輸出通道。
- *[in] double current*
  所需輸出的電流值，單位為安培 (A)。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

此函式為非阻塞式，不可在未啟用 FPU 的 callback 函式中呼叫。有關 FPU 禁用之 callback 函式的更多詳細資訊，請參考 Callback 函式。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11A44S slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  master.start();
}

void loop() {
  slave.currentWrite(0, 0.02);
  delay(1000);

  slave.currentWrite(0, 0);
  delay(1000);

  slave.currentWrite(0, 0.01);
```

```
  delay(1000);

  slave.currentWrite(0, 0);
  delay(1000);
}
```

類比輸入存取函式

適用於 EthercatDevice_DmpAIQ_Generic 類別的類比輸入操作函式。

函式：

- analogRead()
- voltageRead()

analogRead()

說明

讀取 EtherCAT 從站上指定的類比輸入通道的數值。

語法

```
int analogRead(int ch);
```

參數

- *[in] int ch*
  EtherCAT 從站上指定的類比輸入通道。

傳回值

傳回一個 32 位元的帶符號整數，其範圍為 -2^{31} 到 2^{31}，作為類比輸入的數值。該數值會線性對應至實際的輸入電壓 (單位為伏特)：0 對應 0V、2^{31} 對應 64V、-2^{31} 對應 -64V。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11A44S slave;

void setup() {
  Serial.begin(115200);
  master.begin();
  slave.attach(0, master);
  master.start();
}

void loop() {
  Serial.print("AIO: ");
  Serial.println(slave.analogRead(0));
  delay(1000);
}
```

voltageRead()

說明

讀取 EtherCAT 從站上指定類比輸入通道的電壓值。

語法

```
double voltageRead(int ch);
```

參數

- *[in] int ch*
  EtherCAT 從站上指定的類比輸入通道。

傳回值

傳回以伏特 (V) 為單位的電壓值 (double 型別)。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

此函式為非阻塞式，不可在未啟用 FPU 的 callback 函式中呼叫。有關 FPU 禁用之 callback 函式的更多詳細資訊，請參考 Callback 函式。

範例

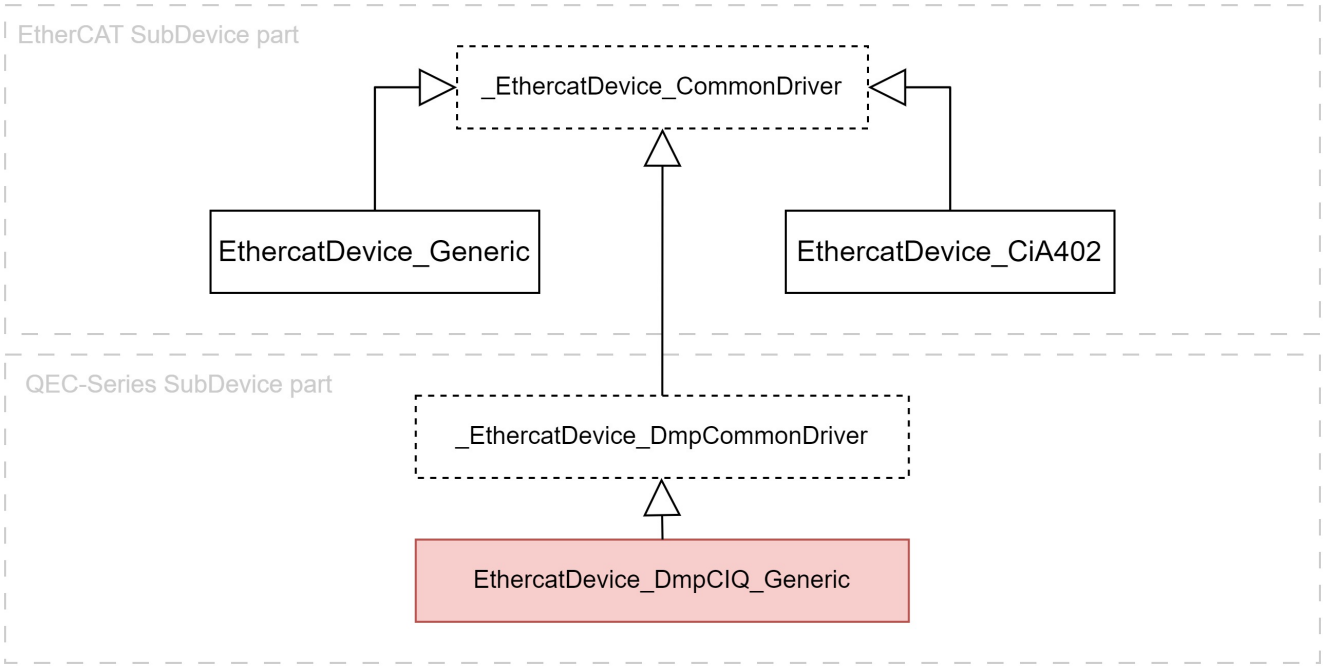
```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11A44S slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  master.start();
}

void loop() {
  Serial.print("AIO: ");
  Serial.println(slave.voltageRead(0));
  delay(1000);
}
```

2.3.4 EthercatDevice_DmpCIQ_Generic

EthercatDevice_DmpCIQ_Generic 是由 ICOP 專為複合型 I/O（Compound I/O）EtherCAT 從站模組所開發的從站裝置類別，整合了數位 I/O、類比 I/O 以及 RS232/RS485 通訊功能。

RS232 和 **RS485** 是電氣規範，定義了透過實體電纜實現 UART 通訊的電壓等級、訊號時序和連接器引腳排列。 ***UART (Universal Asynchronous Receiver Transmitter)*** 是一種非同步串列資料通訊的硬體介面標準。它定義了串列通訊的資料位元、起始位元和停止位元、奇偶校驗位元和波特率的格式。 UART 通常用於連接微控制器、電腦和其他設備以進行資料交換。該設備具有一個 UART 端口，可在 RS232 或 RS485 模式之間自由切換，以滿足用戶的應用需求。

EthercatDevice_DmpCIQ_Generic 的類別關係如下圖所示：



- *EthercatDevice_DmpCIQ_Generic* 繼承自 *_EthercatDevice_DmpCommonDriver*.

基礎類別：

- _EthercatDevice_DmpCommonDriver

衍生類別：

類別名稱	Vendor ID	Product Code	DI	DO	AI	AO	UART
EthercatDevice_QECR11CFFG	0x00000bc3	0x0086d903	16	16	1	1	1

函式分類：

- 初始化
- 控制
- 數位 I/O 存取函式
- 類比 I/O 存取函式
- UART 存取函式

函式：

函式名稱	說明	Callback 中呼叫
初始化相關函式		
attach()	初始化此 EtherCAT 從站裝置類別的物件	
detach()	解除初始化此 EtherCAT 從站裝置類別的物件	
控制相關函式		
update()	更新各功能的狀態機與內部變數	O
數位 I/O 存取函式		
digitalWrite()	寫入指定數位輸出腳位的狀態	O
digitalRead()	讀取指定數位輸入腳位的狀態	O
digitalWriteAll()	寫入所有腳位的數位輸出狀態	O
digitalReadAll()	讀取所有腳位的數位輸入狀態	O
類比 I/O 存取函式		
analogWrite()	寫入指定類比輸出通道的數值	O
voltageWrite()	寫入指定通道的電壓值	O[1]
currentWrite()	寫入指定通道的電流值	O[1]
analogRead()	讀取指定類比輸入通道的數值	O
voltageRead()	讀取指定通道的電壓值	O[1]
UART 存取函式		
uartIs485()	檢查指定 UART 埠是否為 RS485 模式	O
uartSetBaud()	設定鮑率（baud rate）	
uartSetFormat()	設定 UART 格式	
uartSetFlowControl()	設定流量控制模式	
uartGetRTS()	取得 RTS 控制訊號的目前狀態	O
uartGetCTS()	取得 CTS 控制訊號的目前狀態	O
uartGetDTR()	取得 DTR 控制訊號的目前狀態	O
uartGetDSR()	取得 DSR 控制訊號的目前狀態	O
uartSetRTS()	控制 RTS 訊號	O
uartSetDTR()	控制 DTR 訊號	O
uartClearFIFO()	清除 TX 與 RX 的硬體 FIFO	

uartClearTxQueue()	清除軟體傳送 FIFO	O
uartClearRxQueue()	清除軟體接收 FIFO	O
uartQueryTxQueue()	查詢軟體傳送 FIFO 的目前位元組數	O
uartQueryRxQueue()	查詢軟體接收 FIFO 的目前位元組數	O
uartTxQueueEmpty()	檢查軟體傳送 FIFO 是否為空	O
uartRxQueueEmpty()	檢查軟體接收 FIFO 是否為空	O
uartTxQueueFull()	檢查軟體傳送 FIFO 是否已滿	O
uartRxQueueFull()	檢查軟體接收 FIFO 是否已滿	O
uartSend()	傳送多位元組資料	O
uartWrite()	傳送單一位元組資料	O
uartReceive()	接收多位元組資料	O
uartRead()	讀取單一位元組資料	O

- **Note 1**：此函式包含浮點數運算，因此不可在禁用 FPU（浮點運算單元）的 Callback 函式中呼叫。如需更多關於禁用 FPU Callback 函式的說明，請參閱 Callback 函式 章節。

初始化函式

適用於 EthercatDevice_DmpCIQ_Generic 類別的初始化相關函式。

函式：

- attach()
- detach()

attach()

說明

此函式的行為與 `EthercatDevice Generic::attach()` 相同。

詳細的說明、語法、參數與傳回值，請參閱該章節。

範例

適用於 EthercatDevice_QECR11CFFG。

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave;

void setup(void) {
    master.begin();
    slave.attach(0, master);
    master.start();
}
void loop() {
    // ...

}
```

detach()

說明

此函式的行為與 `EthercatDevice Generic::detach()` 相同。

詳細的說明、語法、參數與傳回值，請參閱該章節。

範例

適用於 EthercatDevice_QECR11CFFG。

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave;

void setup() {
  master.begin();
  slave.attach(0, master);

  delay(3000);

  slave.detach();
  master.end();
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

控制函式

適用於 EthercatDevice_DmpCIQ_Generic 類別的控制相關函式。

函式：

- update()

update()

說明

更新 EtherCAT 從站上各功能的狀態機與內部變數。

語法

```
int update();
```

參數

無。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  master.start();
}

void loop() {
  slave.update();
  // ...
}
```

數位 I/O 函式

適用於 EthercatDevice_DmpCIQ_Generic 類別的數位輸入/輸出存取函式。

函式：

- digitalWrite()
- digitalRead()
- digitalWriteAll()
- digitalReadAll()

digitalWrite()

說明

將 HIGH 或 LOW 的數位值寫入指定的 EtherCAT 從站數位輸出腳位。

語法

```
int digitalWrite(int pin, uint8_t value);
```

參數

- *[in] int pin*
  EtherCAT 從站的數位輸出腳位編號。
- *[in] uint8_t value*
  數位邏輯電平，值為 HIGH 或 LOW。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave;

void setup() {
    master.begin();
    slave.attach(0, master);
    master.start();
}
void loop() {
    slave.digitalWrite(0, HIGH);
    delay(4000);
    slave.digitalWrite(0, LOW);
    delay(1000);
}
```

digitalRead()

說明

讀取指定 EtherCAT 從站數位輸入腳位的值。

語法

```
int digitalRead(int pin);
```

參數

- *[in] int pin*
  EtherCAT 從站的數位輸出腳位編號。

傳回值

傳回指定腳位的數位邏輯電平，值為 HIGH 或 LOW。

備註

本函式必須在成功執行 EthercatMaster::start() 並在 EthercatMaster::stop() 之前呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave;

void setup() {
  Serial.begin(115200);
  master.begin();
  slave.attach(0, master);
  master.start();
}

void loop() {
  slave.digitalWrite(0, HIGH);
  delay(1000);
  Serial.print("DI0: "); Serial.println(slave.digitalRead(0));

  slave.digitalWrite(0, LOW);
  delay(1000);
  Serial.print("DI0: "); Serial.println(slave.digitalRead(0));
}
```

digitalWriteAll()

說明

同時寫入 EtherCAT 從站所有數位輸出腳位的狀態。

語法

```
int digitalWriteAll(uint32_t value);
```

參數

- *[in] uint32_t value*

  此參數為一個 32 位元的無符號整數，用以指定要寫入所有數位輸出腳位的值。每一個位元對應一個數位輸出腳位，對應如下：:

  - 第 0 位元對應數位輸出腳位 0。
  - 第 1 位元對應數位輸出腳位 1。
  - 依此類推至第 31 位元對應腳位 31。

  位元值為 1 表示將該腳位設為 HIGH，值為 0 則為 LOW。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  master.start();
}

void loop() {
  slave.digitalWriteAll(0x55555555);
  delay(1000);
  slave.digitalWriteAll(0xAAAAAAAA);
  delay(1000);
}
```

digitalReadAll()

說明

同時讀取 EtherCAT 從站所有數位輸入腳位的狀態。

語法

```
uint32_t digitalReadAll();
```

參數

無。

傳回值

傳回一個 32 位元的無符號整數，代表所有數位輸入腳位的組合狀態。傳回值中的每個位元對應一個數位輸入腳位，對應如下：

- 第 0 位元表示數位輸入腳位 0。
- 第 1 位元表示數位輸入腳位 1。
- 依此類推至第 31 位元表示腳位 31。

值為 1 表示對應的腳位目前為 HIGH，值為 0 則表示為 LOW。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"
EthercatMaster master;
EthercatDevice_QECR11CFFG slave;

void setup() {
  Serial.begin(115200);
  master.begin();
  slave.attach(0, master);
  master.start();
}
void loop() {
  slave.digitalWriteAll(0x55555555);
  delay(1000);
  Serial.print("DI: "); Serial.println(slave.digitalReadAll());

  slave.digitalWriteAll(0xAAAAAAAA);
  delay(1000);
  Serial.print("DI: "); Serial.println(slave.digitalReadAll());
}
```

類比 I/O 函式

適用於 EthercatDevice_DmpCIQ_Generic 類別的類比輸入/輸出存取函式。

函式：

- analogWrite()
- voltageWrite()
- currentWrite()
- analogRead()
- voltageRead()

analogWrite()

說明

將值寫入 EtherCAT 從站上指定的類比輸出通道。

語法

```
int analogWrite(int ch, int32_t value);
```

參數

- *[in] int ch*
  EtherCAT 從站上指定的類比輸出通道。
- *[in] int32_t value*
  要寫入的類比輸出值。此參數為 32 位元帶符號整數，根據該通道的模式設定，此值會線性對應為實際的輸出：
  - 電壓模式 (Voltage Mode)：對應關係為：0 映射為 0V、2^{31} 映射為 64V、-2^{31} 映射為 -64V。例如：2^{25} 對應約為 1V。
  - 電流模式 (Current Mode)：對應關係為：0 映射為 0A、2^{31} 映射為 16A、-2^{31} 映射為 -16A。例如：2^{27} 對應約為 1A。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave;

int voltage = 5;

void setup() {
  master.begin();
  slave.attach(0, master);
  master.start();
}

void loop() {
  slave.analogWrite(0, voltage << 25);
  delay(1000);
  slave.analogWrite(0, 0);
```

```
  delay(1000);
  slave.analogWrite(0, -1 * (voltage << 25));
  delay(1000);
  slave.analogWrite(0, 0);
  delay(1000);
}
```

voltageWrite()

說明

將電壓值寫入 EtherCAT 從站上指定的類比輸出通道。本函式用於指定該通道已設定為電壓模式 (Voltage Mode)。

語法

```
int voltageWrite(int ch, double voltage);
```

參數

- *[in] int ch*
  EtherCAT 從站上指定的類比輸出通道。
- *[in] double voltage*
  所需輸出的電壓值，單位為伏特 (V)。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

此函式為非阻塞式，不可在未啟用 FPU 的 callback 函式中呼叫。有關 FPU 禁用之 callback 函式的更多詳細資訊，請參考 Callback 函式。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  master.start();
}

void loop() {
  slave.voltageWrite(0, 10.0);
  delay(1000);

  slave.voltageWrite(0, 0);
  delay(1000);

  slave.voltageWrite(0, -10.0);
```

```
  delay(1000);

  slave.voltageWrite(0, 0);
  delay(1000);
}
```

currentWrite()

說明

將電流值寫入 EtherCAT 從站上指定的類比輸出通道。本函式用於指定該通道已設定為電流模式 (Current Mode)。

語法

```
int currentWrite(int ch, double current);
```

參數

- *[in] int ch*
  EtherCAT 從站上指定的類比輸出通道。
- *[in] double current*
  所需輸出的電流值，單位為安培 (A)。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

此函式為非阻塞式，不可在未啟用 FPU 的 callback 函式中呼叫。有關 FPU 禁用之 callback 函式的更多詳細資訊，請參考 Callback 函式。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  master.start();
}

void loop() {
  slave.currentWrite(0, 0.02);
  delay(1000);

  slave.currentWrite(0, 0);
  delay(1000);

  slave.currentWrite(0, 0.01);
```

```
  delay(1000);

  slave.currentWrite(0, 0);
  delay(1000);
}
```

analogRead()

說明

讀取 EtherCAT 從站上指定的類比輸入通道的數值。

語法

```
int analogRead(int ch);
```

參數

- *[in] int ch*
  EtherCAT 從站上指定的類比輸入通道。

傳回值

傳回一個 32 位元的帶符號整數，其範圍為 -2^{31} 到 2^{31}，作為類比輸入的數值。該數值會線性對應至實際的輸入電壓 (單位為伏特)：0 對應 0V、2^{31} 對應 64V、-2^{31} 對應 -64V。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  master.start();
}

void loop() {
  Serial.print("AIO: ");
  Serial.println(slave.analogRead(0));
  delay(1000);
}
```

voltageRead()

說明

讀取 EtherCAT 從站上指定類比輸入通道的電壓值。

語法

```
double voltageRead(int ch);
```

參數

- *[in] int ch*
  EtherCAT 從站上指定的類比輸入通道。

傳回值

傳回以伏特 (V) 為單位的電壓值 (double 型別)。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

此函式為非阻塞式，不可在未啟用 FPU 的 callback 函式中呼叫。有關 FPU 禁用之 callback 函式的更多詳細資訊，請參考 Callback 函式。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  master.start();
}

void loop() {
  Serial.print("AIO: ");
  Serial.println(slave.voltageRead(0));
  delay(1000);
}
```

UART 函式

適用於 EthercatDevice_DmpCIQ_Generic 類別的 UART 存取函式。

函式：

- uartIs485()
- uartSetBaud()
- uartSetFormat()
- uartSetFlowControl()
- uartGetRTS()
- uartGetCTS()
- uartGetDTR()
- uartGetDSR()
- uartSetRTS()
- uartSetDTR()
- uartClearFIFO()
- uartClearTxQueue()
- uartClearRxQueue()
- uartQueryTxQueue()
- uartQueryRxQueue()
- uartTxQueueEmpty()
- uartRxQueueEmpty()
- uartTxQueueFull()
- uartRxQueueFull()
- uartSend()
- uartWrite()
- uartReceive()
- uartRead()

uartIs485()

說明

檢查 EtherCAT 從站上指定的 UART 埠是否處於 RS485 模式。

語法

```
int uartIs485(int dev);
```

參數

- *[in] int dev*

  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。

傳回值

傳回指定 UART 埠是否為 RS485 模式：

- 1 表示為 RS485 模式。
- 0 表示非 RS485 模式。

若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  Serial.print("RS485 mode: "); Serial.println(slave.uartIs485(0));
}

void loop() {
  // ...
}
```

uartSetBaud()

說明

設定 EtherCAT 從站上指定 UART 埠的鮑率 (Baud Rate)。

語法

```
int uartSetBaud(int dev, uint32_t baud);
```

參數

- *[in] int dev*
  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。
- *[in] uint32_t baud*
  欲設定的鮑率數值。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.uartSetBaud(0, 115200);
  // ...
}

void loop() {
  // ...
}
```

uartSetFormat()

說明

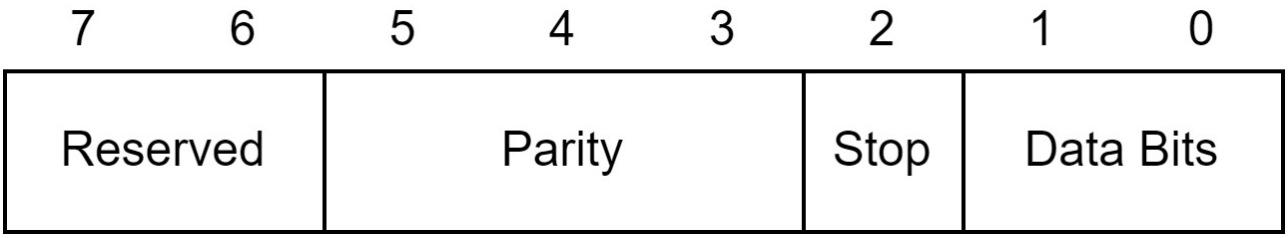
設定 EtherCAT 從站上指定 UART 埠的資料格式。

語法

```
int uartSetFormat(int dev, uint8_t format);
```

參數

- *[in] int dev*
  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。
- *[in] uint8_t format*
  欲設定的 UART 資料格式。
  此參數的位元定義如下：



Data Bits：這些位元定義正在傳送和接收的資料的字長。

定義	數值	說明
ECAT_UART_BYTESIZE5	0x00	5 data bits.
ECAT_UART_BYTESIZE6	0x01	6 data bits.
ECAT_UART_BYTESIZE7	0x02	7 data bits.
ECAT_UART_BYTESIZE8	0x03	8 data bits.

Stop：此位元選擇要傳送的停止位數。

定義	數值	說明
ECAT_UART_STOPBIT1	0x00	One stop bit.
ECAT_UART_STOPBIT2	0x04	Two stop bits (1.5 with 5-bit data).

Parity：這些位元選擇執行奇偶校驗控制的方式。

定義	數值	說明
ECAT_UART_NOPARITY	0x00	No parity bit.
ECAT_UART_ODDPARITY	0x08	Odd parity.
ECAT_UART_EVENPARITY	0x18	Even parity.
ECAT_UART_MARKPARITY	0x28	The parity bit exists and is always 1.
ECAT_UART_SPACEPARITY	0x38	The parity bit exists and is always 0.

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.uartSetFormat(0, ECAT_UART_BYTESIZE8 + ECAT_UART_NOPARITY +
ECAT_UART_STOPBIT1);
  // ...
}

void loop() {
  // ...
}
```

uartSetFlowControl()

說明

設定 EtherCAT 從站上指定 UART 埠的流控模式。

語法

```
int uartSetFlowControl(int dev, int control);
```

參數

- *[in] int dev*
  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。
- *[in] int control*
  欲設定的流控模式。選項如下：
  - ***ECAT_UART_NO_CONTROL***：停用流控功能。
  - ***ECAT_UART_RTS_CTS***：使用 RTS/CTS 硬體流控，常用於 RS232 傳輸。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.uartSetFlowControl(0, ECAT_UART_RTS_CTS);
}

void loop() {
  // ...
}
```

uartGetRTS()

說明

取得 EtherCAT 從站上指定 UART 埠的 RTS 控制訊號目前狀態。

語法

```
int uartGetRTS(int dev);
```

參數

- *[in] int dev*

  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。

傳回值

傳回 RTS 控制訊號的目前狀態。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  slave.uartSetRTS(0, 1);
  Serial.print("RTS: ");
  Serial.println(slave.uartGetRTS(0));
```

```
  delay(1000);

  slave.uartSetRTS(0, 0);
  Serial.print("RTS: ");
  Serial.println(slave.uartGetRTS(0));
  delay(500);
  // ...
}
```

uartGetCTS()

說明

取得 EtherCAT 從站上指定 UART 埠的 CTS 控制訊號目前狀態。

語法

```
int uartGetCTS(int dev);
```

參數

- *[in] int dev*

  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。

傳回值

傳回 CTS 控制訊號的目前狀態。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  Serial.print("CTS: ");
  Serial.println(slave.uartGetCTS(0));
  delay(1000);
```

```
  // ...
}
```

uartGetDTR()

說明

取得 EtherCAT 從站上指定 UART 埠的 DTR 控制訊號目前狀態。

語法

```
int uartGetDTR(int dev);
```

參數

- *[in] int dev*

  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。

傳回值

傳回 DTR 控制訊號的目前狀態。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  slave.uartSetDTR(0, 1);
  Serial.print("DTR: ");
  Serial.println(slave.uartGetDTR(0));
```

```
  delay(1000);

  slave.uartSetDTR(0, 0);
  Serial.print("DTR: ");
  Serial.println(slave.uartGetDTR(0));
  delay(500);
  // ...
}
```

uartGetDSR()

說明

取得 EtherCAT 從站上指定 UART 埠的 DSR 訊號目前狀態。

語法

```
int uartGetDSR(int dev);
```

參數

- *[in] int dev*

  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。

傳回值

傳回 DSR 訊號的目前狀態。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  Serial.print("DSR: ");
  Serial.println(slave.uartGetDSR(0));
  delay(1000);
```

```
  // ...
}
```

uartSetRTS()

說明

控制 EtherCAT 從站上指定 UART 埠的 RTS 訊號。

語法

```
int uartSetRTS(int dev, uint8_t value);
```

參數

- *[in] int dev*
  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。
- *[in] unit8_t value*
  要設定的 RTS 訊號值。
  0：表示 RTS 訊號為非作用狀態
  1 到 255：表示 RTS 訊號為作用狀態

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  slave.uartSetRTS(0, 1);
```

```
  delay(1000);
  slave.uartSetRTS(0, 0);
  delay(500);
  // ...
}
```

uartSetDTR()

說明

控制 EtherCAT 從站上指定 UART 埠的 DTR 訊號。

語法

```
int uartSetDTR(int dev, uint8_t value);
```

參數

- *[in] int dev*
  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。
- *[in] unit8_t value*
  要設定的 DTR 訊號值。
  0：表示 DTR 訊號為非作用狀態
  1 到 255：表示 DTR 訊號為作用狀態

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  slave.uartSetDTR(0, 1);
```

```
  delay(1000);
  slave.uartSetDTR(0, 0);
  delay(500);
  // ...
}
```

uartClearFIFO()

說明

清除 EtherCAT 從站上指定 UART 埠的 TX 與 RX 的 FIFO 緩衝區。

語法

```
int uartClearFIFO(int dev);
```

參數

- *[in] int dev*

  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.uartClearFIFO(0);
}

void loop() {
  // ...
}
```

uartClearTxQueue()

說明

清除本程式庫中 EtherCAT 從站指定的 UART 埠的軟體 TX 的 FIFO 緩衝區。

語法

```
int uartClearTxQueue(int dev);
```

參數

- *[in] int dev*

  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.uartClearTxQueue(0);
}

void loop() {
  // ...
}
```

uartClearRxQueue()

說明

清除本程式庫中 EtherCAT 從站指定的 UART 埠的軟體 RX 的 FIFO 緩衝區。

語法

```
int uartClearRxQueue(int dev);
```

參數

- *[in] int dev*

  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.uartClearRxQueue(0);
}

void loop() {
  // ...
}
```

uartQueryTxQueue()

說明

取得本程式庫中 EtherCAT 從站的指定 UART 埠之軟體 TX 的 FIFO 緩衝區中目前的位元組數。

語法

```
int uartQueryTxQueue(int dev);
```

參數

- *[in] int dev*

  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。

傳回值

傳回該 UART 埠的軟體 TX FIFO 中目前的位元組數。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  Serial.print("TX FIFO: ");
  Serial.println(slave.uartQueryTxQueue(0));
}

void loop() {
  // ...
}
```

uartQueryRxQueue()

說明

取得本程式庫中 EtherCAT 從站的指定 UART 埠之軟體 RX 的 FIFO 緩衝區中目前的位元組數。

語法

```
int uartQueryRxQueue(int dev);
```

參數

- *[in] int dev*

  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。

傳回值

傳回該 UART 埠的軟體 RX FIFO 中目前的位元組數。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  Serial.print("RX FIFO: ");
  Serial.println(slave.uartQueryRxQueue(0));
}

void loop() {
  // ...
}
```

uartTxQueueEmpty()

說明

檢查本函式庫中 EtherCAT 從站指定 UART 埠的軟體 TX FIFO 是否為空值。

語法

```
int uartTxQueueEmpty(int dev);
```

參數

- *[in] int dev*

  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。

傳回值

傳回指定 UART 埠的軟體 TX FIFO 是否為空。若該埠的 FIFO 為空，傳回值為 1。若不為空，傳回值為 0。

若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  Serial.print("TX FIFO is Empty: ");
  Serial.println(slave.uartTxQueueEmpty(0));
}
void loop() {
  // ...
}
```

uartRxQueueEmpty()

說明

檢查本函式庫中 EtherCAT 從站指定 UART 埠的軟體 RX FIFO 是否為空值。

語法

```
int uartRxQueueEmpty(int dev);
```

參數

- *[in] int dev*

  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。

傳回值

傳回指定 UART 埠的軟體 RX FIFO 是否為空。若該埠的 FIFO 為空，傳回值為 1。若不為空，傳回值為 0。

若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  Serial.print("RX FIFO is Empty: ");
  Serial.println(slave.uartRxQueueEmpty(0));
}
void loop() {
  // ...
}
```

uartTxQueueFull()

說明

檢查本函式庫中 EtherCAT 從站指定 UART 埠的軟體 TX FIFO 是否已滿。

語法

```
int uartTxQueueFull(int dev);
```

參數

- *[in] int dev*

  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。

傳回值

傳回指定 UART 埠的軟體 TX FIFO 是否已滿。若 FIFO 已滿，傳回值為 1。若未滿，傳回值為 0。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  Serial.print("TX FIFO is Full: ");
  Serial.println(slave.uartTxQueueFull(0));
}
void loop() {
  // ...
}
```

uartRxQueueFull()

說明

檢查本函式庫中 EtherCAT 從站指定 UART 埠的軟體 RX FIFO 是否已滿。

語法

```
int uartRxQueueFull(int dev);
```

參數

- *[in] int dev*

  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。

傳回值

傳回指定 UART 埠的軟體 RX FIFO 是否已滿。若 FIFO 已滿，傳回值為 1。若未滿，傳回值為 0。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  Serial.print("RX FIFO is Full: ");
  Serial.println(slave.uartRxQueueFull(0));
}
void loop() {
  // ...
}
```

uartSend()

說明

從 EtherCAT 從站指定的 UART 埠傳送多個位元組的資料。

語法

```
int uartSend(int dev, void *buf, size_t size);
```

參數

- *[in] int dev*
  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。
- *[in] void *buf*
  要傳送的資料緩衝區。
- *[in] size_t size*
  資料緩衝區的大小。

傳回值

傳回實際傳送的位元組數。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  char buffer[] = {"Hello world!"};

  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
```

```
  slave.uartSend(0, buffer, strlen(buffer));

  Serial.print("Sent: ");
  Serial.println(buffer);
}

void loop() {
  // ...
}
```

uartWrite()

說明

從 EtherCAT 從站指定的 UART 埠傳送一個位元組的資料。

語法

```
int uartWrite(int dev, uint8_t value);
```

參數

- *[in] int dev*
  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。
- *[in] unit8_t value*
  要傳送的一個位元組資料。

傳回值

傳回實際傳送的位元組數。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  slave.uartWrite(0, 'H');
  slave.uartWrite(0, 'e');
  slave.uartWrite(0, 'l');
  slave.uartWrite(0, 'l');
  slave.uartWrite(0, 'o');
```

```
}

void loop() {
  // ...
}
```

uartReceive()

說明

從 EtherCAT 從站指定的 UART 埠接收多個位元組的資料。

語法

```
int uartReceive(int dev, void *buf, size_t size);
```

參數

- *[in] int dev*
  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。
- *[in] void *buf*
  用於接收資料的緩衝區。
- *[in] size_t size*
  緩衝區的大小。

傳回值

傳回實際接收的位元組數。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave;

char buffer[256];
int rc;

void CyclicCallback() {
  slave.update();
}

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
```

```
  master.start();
}

void loop() {
  rc = slave.uartReceive(0, buffer, 256);
  for (int i = 0; i < rc; i++) {
    Serial.print(buffer[i]);
  }

  // ...
}
```

uartRead()

說明

從 EtherCAT 從站指定的 UART 埠讀取 1 個位元組的資料。

語法

```
int uartRead(int dev);
```

參數

- *[in] int dev*

  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。

傳回值

傳回 1 個位元組的資料。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  int ch = slave.uartRead(0);
  if (ch >= 0)
    Serial.print((char)ch);
```

```
  // ...
}
```

2.3.5 EthercatDevice_DmpHID_Generic

EthercatDevice_DmpHID_Generic 是由 ICOP 專為 QEC EtherCAT 從站 HID 模組所開發的從站裝置類別。它整合了 RS232/RS485、4x4 鍵盤、16x2 LCM、手輪 (MPG) 以及蜂鳴器 (Buzzer) 等功能。

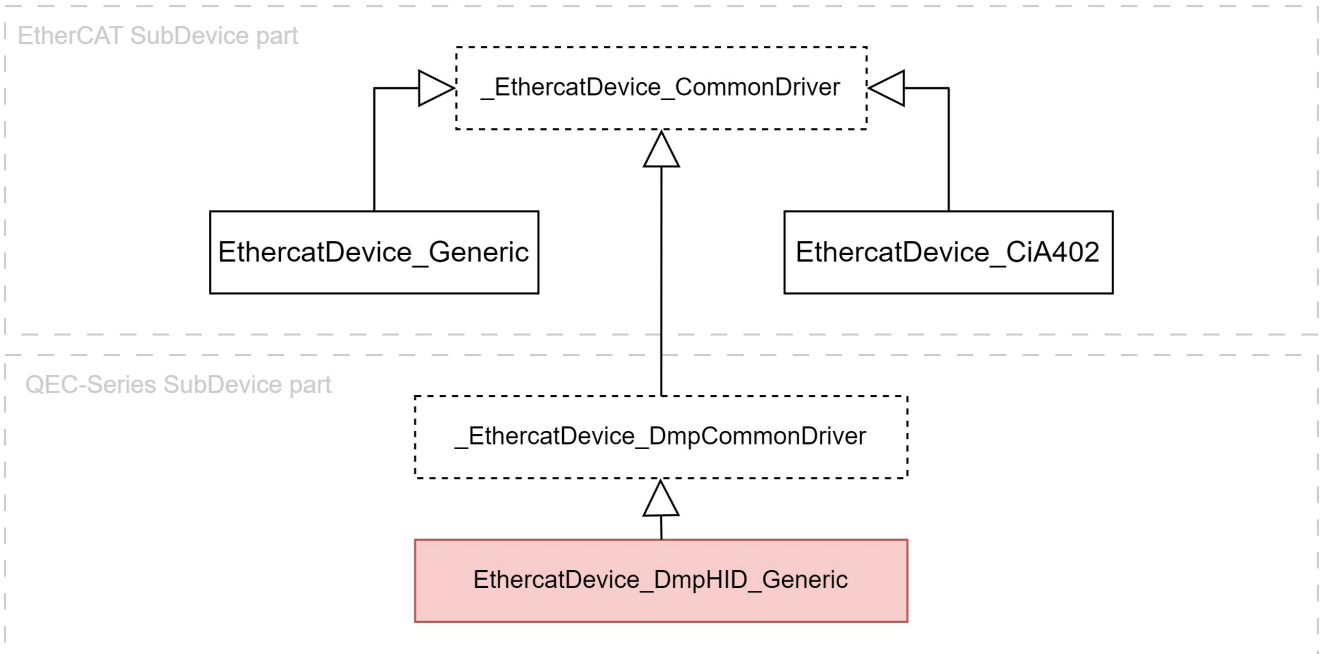
RS232 和 **RS485** 是電氣規範，定義了透過實體電纜實現 UART 通訊的電壓等級、訊號時序和連接器引腳排列。 ***UART (Universal Asynchronous Receiver Transmitter)*** 是一種非同步串列資料通訊的硬體介面標準。它定義了串列通訊的資料位元、起始位元和停止位元、奇偶校驗位元和波特率的格式。 UART 通常用於連接微控制器、電腦和其他設備以進行資料交換。該設備具有兩個 UART 端口，可在 RS232 或 RS485 模式之間自由切換，以滿足用戶的應用需求。

本裝置內建 **4x4 矩陣式鍵盤**，提供輸入數字、符號與字元的使用者介面。鍵盤排列為 4 列 4 行，每個按鍵對應特定字元或功能。

本裝置內建 **2x16 字元型液晶模組（LCM）**，可用於顯示資訊給使用者。此 LCM 採用矩陣式配置，共兩列，每列可顯示最多 16 個字元。

本裝置提供 **手輪 (MPG，Manual Pulse Generator) 介面**，可連接特定 MPG 裝置，實現機台或系統移動的精確控制。在本裝置中，用於計數手輪脈衝的介面稱為「編碼器 (Encoder)」。
此外，此介面也支援 倍率旋鈕（Ratio knob）、軸選旋鈕（Axis knob）、急停開關（Emergency Stop）、啟用開關（Enable Switch）等控制功能。

EthercatDevice_DmpHID_Generic 的類別關係如下圖所示：



- *EthercatDevice_DmpHID_Generic* 繼承自 *_EthercatDevice_DmpCommonDriver*.

基礎類別：

- _EthercatDevice_CommonDriver

衍生類別：

類別名稱	Vendor ID	Product Code	UART	Keypad	LCM	MPG
EthercatDevice_QECR11HU1S	0x00000bc3	0x0086d404	O			O
EthercatDevice_QECR11HU5S	0x00000bc3	0x0086d403	O			
EthercatDevice_QECR00HU5S	0x00000bc3	0x0086d400	O			
EthercatDevice_QECR11HU9S	0x00000bc3	0x0086d402	O	O	O	O
EthercatDevice_QECR00HU9S	0x00000bc3	0x0086d401	O	O	O	O

函式分類：

- 初始化
- 控制
- UART 存取函式
- LCM 存取函式
- Keypad 存取函式
- MPG 存取函式
- 蜂鳴器函式

函式：

函式名稱	說明	Callback 中呼叫
初始化相關函式		
attach()	初始化此 EtherCAT 從站裝置類別的物件。	
detach()	解除初始化此 EtherCAT 從站裝置類別的物件。	
控制相關函式		
update()	更新各功能模組的狀態機與內部變數。	O
UART 存取函式		
uartIs485()	檢查指定 UART 埠是否為 RS485 模式	O
uartSetBaud()	設定鮑率（baud rate）	
uartSetFormat()	設定 UART 格式	
uartSetFlowControl()	設定流量控制模式	
uartGetRTS()	取得 RTS 控制訊號的目前狀態	O
uartGetCTS()	取得 CTS 控制訊號的目前狀態	O
uartGetDTR()	取得 DTR 控制訊號的目前狀態	O
uartGetDSR()	取得 DSR 控制訊號的目前狀態	O

uartSetRTS()	控制 RTS 訊號	O
uartSetDTR()	控制 DTR 訊號	O
uartClearFIFO()	清除 TX 與 RX 的硬體 FIFO	
uartClearTxQueue()	清除軟體傳送 FIFO	O
uartClearRxQueue()	清除軟體接收 FIFO	O
uartQueryTxQueue()	查詢軟體傳送 FIFO 的目前位元組數	O
uartQueryRxQueue()	查詢軟體接收 FIFO 的目前位元組數	O
uartTxQueueEmpty()	檢查軟體傳送 FIFO 是否為空	O
uartRxQueueEmpty()	檢查軟體接收 FIFO 是否為空	O
uartTxQueueFull()	檢查軟體傳送 FIFO 是否已滿	O
uartRxQueueFull()	檢查軟體接收 FIFO 是否已滿	O
uartSend()	傳送多位元組資料	O
uartWrite()	傳送單一位元組資料	O
uartReceive()	接收多位元組資料	O
uartRead()	讀取單一位元組資料	O
LCM 存取函式		
lcmHeight()	取得 LCM 的高度	O
lcmWidth()	取得 LCM 的寬度	O
lcmWordWrap()	啟用或停用自動換行功能	O
lcmGotoXY()	移動游標至指定座標	O
lcmClear()	清除畫面	O
lcmPrint()	列印字串	O
lcmWrite()	列印單一字元	O
鍵盤存取函式		
keypadSetTimeout()	設定鍵盤輸入資料緩衝區的逾時時間	O
keypadClear()	清除鍵盤輸入資料緩衝區	O
keypadRead()	讀取一個鍵盤輸入字元	O
MPG 存取函式		
mpgSetCallback()	註冊 MPG 事件回呼函式	O
mpgSetNoiseFilter()	設定雜訊濾波器	
mpgInvertEncoderDirection()	反轉編碼器計數方向	
mpgWriteEncoderRaw()	寫入編碼器的原始計數值	
mpgWriteEncoder()	寫入編碼器的邏輯計數值	O
mpgReadEncoderRaw()	讀取編碼器的原始計數值	O
mpgReadEncoder()	讀取編碼器的邏輯計數值	O
mpgReadEmergencyStop()	讀取急停開關狀態	O
mpgReadEnableSwitch()	讀取啟用開關狀態	O

mpgReadAxis()	讀取軸選旋鈕的值	O
mpgReadRatio()	讀取倍率旋鈕的值	O
蜂鳴器函式		
buzzer()	發出指定頻率的聲音	O

初始化函式

適用於 EthercatDevice_DmpHID_Generic 類別的初始化相關函式。

函式：

- attach()
- detach()

attach()

說明

此函式的行為與 `EthercatDevice Generic::attach()` 相同。

詳細的說明、語法、參數與傳回值，請參閱該章節。

範例

適用於 EthercatDevice_QECR00HU9S。

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00HU9S slave;

void setup(void) {
    master.begin();
    slave.attach(0, master);
    master.start();
}

void loop() {
    // ...

}
```

detach()

說明

此函式的行為與 `EthercatDevice Generic::detach()` 相同。

詳細的說明、語法、參數與傳回值，請參閱該章節。

範例

適用於 EthercatDevice_QECR11HU9S。

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void setup() {
  master.begin();
  slave.attach(0, master);

  delay(3000);

  slave.detach();
  master.end();
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

控制函式

適用於 EthercatDevice_DmpHID_Generic 類別的控制相關函式。

函式：

- update()

update()

說明

更新 EtherCAT 從站上各功能的狀態機與內部變數。

語法

```
int update();
```

參數

無。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  master.start();
}

void loop() {
  slave.update();
  // ...
}
```

UART 函式

適用於 EthercatDevice_DmpHID_Generic 類別的 UART 存取函式。

函式：

- uartIs485()
- uartSetBaud()
- uartSetFormat()
- uartSetFlowControl()
- uartGetRTS()
- uartGetCTS()
- uartGetDTR()
- uartGetDSR()
- uartSetRTS()
- uartSetDTR()
- uartClearFIFO()
- uartClearTxQueue()
- uartClearRxQueue()
- uartQueryTxQueue()
- uartQueryRxQueue()
- uartTxQueueEmpty()
- uartRxQueueEmpty()
- uartTxQueueFull()
- uartRxQueueFull()
- uartSend()
- uartWrite()
- uartReceive()
- uartRead()

uartIs485()

說明

檢查 EtherCAT 從站上指定的 UART 埠是否處於 RS485 模式。

語法

```
int uartIs485(int dev);
```

參數

- *[in] int dev*

  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。1 表示 COM2。

傳回值

傳回指定 UART 埠是否為 RS485 模式：

- 1 表示為 RS485 模式。
- 0 表示非 RS485 模式。

若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  Serial.print("RS485 mode: ");
  Serial.println(slave.uartIs485(0));
}

void loop() {
  // ...
}
```

uartSetBaud()

說明

設定 EtherCAT 從站上指定 UART 埠的鮑率 (Baud Rate)。

語法

```
int uartSetBaud(int dev, uint32_t baud);
```

參數

- *[in] int dev*
  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。1 表示 COM2。
- *[in] uint32_t baud*
  欲設定的鮑率數值。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.uartSetBaud(0, 115200);
}

void loop() {
  // ...
}
```

uartSetFormat()

說明

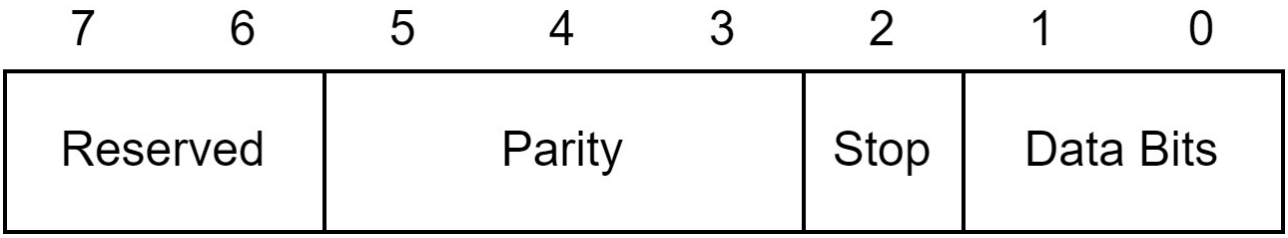
設定 EtherCAT 從站上指定 UART 埠的資料格式。

語法

```
int uartSetFormat(int dev, uint8_t format);
```

參數

- *[in] int dev*
  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。1 表示 COM2。
- *[in] uint8_t format*
  欲設定的 UART 資料格式。
  此參數的位元定義如下：



Data Bits：這些位元定義正在傳送和接收的資料的字長。

定義	數值	說明
ECAT_UART_BYTESIZE5	0x00	5 data bits.
ECAT_UART_BYTESIZE6	0x01	6 data bits.
ECAT_UART_BYTESIZE7	0x02	7 data bits.
ECAT_UART_BYTESIZE8	0x03	8 data bits.

Stop：此位元選擇要傳送的停止位數。

定義	數值	說明
ECAT_UART_STOPBIT1	0x00	One stop bit.
ECAT_UART_STOPBIT2	0x04	Two stop bits (1.5 with 5-bit data).

Parity：這些位元選擇執行奇偶校驗控制的方式。

定義	數值	說明
ECAT_UART_NOPARITY	0x00	No parity bit.
ECAT_UART_ODDPARITY	0x08	Odd parity.
ECAT_UART_EVENPARITY	0x18	Even parity.
ECAT_UART_MARKPARITY	0x28	The parity bit exists and is always 1.
ECAT_UART_SPACEPARITY	0x38	The parity bit exists and is always 0.

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.uartSetFormat(0, ECAT_UART_BYTESIZE8 + ECAT_UART_NOPARITY +
ECAT_UART_STOPBIT1);
}

void loop() {
  // ...
}
```

uartSetFlowControl()

說明

設定 EtherCAT 從站上指定 UART 埠的流控模式。

語法

```
int uartSetFlowControl(int dev, int control);
```

參數

- *[in] int dev*
  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。1 表示 COM2。
- *[in] int control*
  欲設定的流控模式。選項如下：
  - ***ECAT_UART_NO_CONTROL***：停用流控功能。
  - ***ECAT_UART_RTS_CTS***：使用 RTS/CTS 硬體流控，常用於 RS232 傳輸。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.uartSetFlowControl(0, ECAT_UART_RTS_CTS);
}

void loop() {
  // ...
}
```

uartGetRTS()

說明

取得 EtherCAT 從站上指定 UART 埠的 RTS 控制訊號目前狀態。

語法

```
int uartGetRTS(int dev);
```

參數

- *[in] int dev*

  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。1 表示 COM2。

傳回值

傳回 RTS 控制訊號的目前狀態。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  slave.uartSetRTS(0, 1);
  Serial.print("RTS: ");
```

```
  Serial.println(slave.uartGetRTS(0));
  delay(1000);

  slave.uartSetRTS(0, 0);
  Serial.print("RTS: ");
  Serial.println(slave.uartGetRTS(0));
  delay(500);
  // ...
}
```

uartGetCTS()

說明

取得 EtherCAT 從站上指定 UART 埠的 CTS 控制訊號目前狀態。

語法

```
int uartGetCTS(int dev);
```

參數

- *[in] int dev*

  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。1 表示 COM2。

傳回值

傳回 CTS 控制訊號的目前狀態。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  Serial.print("CTS: ");
  Serial.println(slave.uartGetCTS(0));
```

```
  delay(1000);
  // ...
}
```

uartGetDTR()

說明

取得 EtherCAT 從站上指定 UART 埠的 DTR 控制訊號目前狀態。

語法

```
int uartGetDTR(int dev);
```

參數

- *[in] int dev*

  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。1 表示 COM2。

傳回值

傳回 DTR 控制訊號的目前狀態。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  slave.uartSetDTR(0, 1);
  Serial.print("DTR: ");
```

```
  Serial.println(slave.uartGetDTR(0));
  delay(1000);

  slave.uartSetDTR(0, 0);
  Serial.print("DTR: ");
  Serial.println(slave.uartGetDTR(0));
  delay(500);
  // ...
}
```

uartGetDSR()

說明

取得 EtherCAT 從站上指定 UART 埠的 DSR 訊號目前狀態。

語法

```
int uartGetDSR(int dev);
```

參數

- *[in] int dev*

  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。1 表示 COM2。

傳回值

傳回 DSR 訊號的目前狀態。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  Serial.print("DSR: ");
  Serial.println(slave.uartGetDSR(0));
```

```
  delay(1000);
  // ...
}
```

uartSetRTS()

說明

控制 EtherCAT 從站上指定 UART 埠的 RTS 訊號。

語法

```
int uartSetRTS(int dev, uint8_t value);
```

參數

- *[in] int dev*
  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。1 表示 COM2。
- *[in] unit8_t value*
  要設定的 RTS 訊號值。
  0：表示 RTS 訊號為非作用狀態
  1 到 255：表示 RTS 訊號為作用狀態

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  slave.uartSetRTS(0, 1);
```

```
  delay(1000);
  slave.uartSetRTS(0, 0);
  delay(500);
  // ...
}
```

uartSetDTR()

說明

控制 EtherCAT 從站上指定 UART 埠的 DTR 訊號。

語法

```
int uartSetDTR(int dev, uint8_t value);
```

參數

- *[in] int dev*
  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。1 表示 COM2。
- *[in] unit8_t value*
  要設定的 DTR 訊號值。
  0：表示 DTR 訊號為非作用狀態
  1 到 255：表示 DTR 訊號為作用狀態

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  slave.uartSetDTR(0, 1);
```

```
  delay(1000);
  slave.uartSetDTR(0, 0);
  delay(500);
  // ...
}
```

uartClearFIFO()

說明

清除 EtherCAT 從站上指定 UART 埠的 TX 與 RX 的 FIFO 緩衝區。

語法

```
int uartClearFIFO(int dev);
```

參數

- *[in] int dev*

  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。1 表示 COM2。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.uartClearFIFO(0);
}

void loop() {
  // ...
}
```

uartClearTxQueue()

說明

清除本程式庫中 EtherCAT 從站指定的 UART 埠的軟體 TX 的 FIFO 緩衝區。

語法

```
int uartClearTxQueue(int dev);
```

參數

- *[in] int dev*

  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。1 表示 COM2。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.uartClearTxQueue(0);
}

void loop() {
  // ...
}
```

uartClearRxQueue()

說明

清除本程式庫中 EtherCAT 從站指定的 UART 埠的軟體 RX 的 FIFO 緩衝區。

語法

```
int uartClearRxQueue(int dev);
```

參數

- *[in] int dev*

  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。1 表示 COM2。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.uartClearRxQueue(0);
}

void loop() {
  // ...
}
```

uartQueryTxQueue()

說明

取得本程式庫中 EtherCAT 從站的指定 UART 埠之軟體 TX 的 FIFO 緩衝區中目前的位元組數。

語法

```
int uartQueryTxQueue(int dev);
```

參數

- *[in] int dev*

  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。1 表示 COM2。

傳回值

傳回該 UART 埠的軟體 TX FIFO 中目前的位元組數。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  Serial.print("TX FIFO: ");
  Serial.println(slave.uartQueryTxQueue(0));
}

void loop() {
  // ...
}
```

uartQueryRxQueue()

說明

取得本程式庫中 EtherCAT 從站的指定 UART 埠之軟體 RX 的 FIFO 緩衝區中目前的位元組數。

語法

```
int uartQueryRxQueue(int dev);
```

參數

- *[in] int dev*

  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。1 表示 COM2。

傳回值

傳回該 UART 埠的軟體 RX FIFO 中目前的位元組數。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  Serial.print("RX FIFO: ");
  Serial.println(slave.uartQueryRxQueue(0));
}

void loop() {
  // ...
}
```

uartTxQueueEmpty()

說明

檢查本函式庫中 EtherCAT 從站指定 UART 埠的軟體 TX FIFO 是否為空值。

語法

```
int uartTxQueueEmpty(int dev);
```

參數

- *[in] int dev*

  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。1 表示 COM2。

傳回值

傳回指定 UART 埠的軟體 TX FIFO 是否為空。若該埠的 FIFO 為空，傳回值為 1。若不為空，傳回值為 0。

若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  Serial.print("TX FIFO is Empty: ");
  Serial.println(slave.uartTxQueueEmpty(0));
}
void loop() {
  // ...
}
```

uartRxQueueEmpty()

說明

檢查本函式庫中 EtherCAT 從站指定 UART 埠的軟體 RX FIFO 是否為空值。

語法

```
int uartRxQueueEmpty(int dev);
```

參數

- *[in] int dev*

  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。1 表示 COM2。

傳回值

傳回指定 UART 埠的軟體 RX FIFO 是否為空。若該埠的 FIFO 為空，傳回值為 1。若不為空，傳回值為 0。

若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  Serial.print("RX FIFO is Empty: ");
  Serial.println(slave.uartRxQueueEmpty(0));
}
void loop() {
  // ...
}
```

uartTxQueueFull()

說明

檢查本函式庫中 EtherCAT 從站指定 UART 埠的軟體 TX FIFO 是否已滿。

語法

```
int uartTxQueueFull(int dev);
```

參數

- *[in] int dev*

  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。1 表示 COM2。

傳回值

傳回指定 UART 埠的軟體 TX FIFO 是否已滿。若 FIFO 已滿，傳回值為 1。若未滿，傳回值為 0。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  Serial.print("TX FIFO is Full: ");
  Serial.println(slave.uartTxQueueFull(0));
}
void loop() {
  // ...
}
```

uartRxQueueFull()

說明

檢查本函式庫中 EtherCAT 從站指定 UART 埠的軟體 RX FIFO 是否已滿。

語法

```
int uartRxQueueFull(int dev);
```

參數

- *[in] int dev*

  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。1 表示 COM2。

傳回值

傳回指定 UART 埠的軟體 RX FIFO 是否已滿。若 FIFO 已滿，傳回值為 1。若未滿，傳回值為 0。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  Serial.print("RX FIFO is Full: ");
  Serial.println(slave.uartRxQueueFull(0));
}
void loop() {
  // ...
}
```

uartSend()

說明

從 EtherCAT 從站指定的 UART 埠傳送多個位元組的資料。

語法

```
int uartSend(int dev, void *buf, size_t size);
```

參數

- *[in] int dev*
  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。1 表示 COM2。
- *[in] void *buf*
  要傳送的資料緩衝區。
- *[in] size_t size*
  資料緩衝區的大小。

傳回值

傳回實際傳送的位元組數。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  char buffer[] = {"Hello world!"};

  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
```

```
  slave.uartSend(0, buffer, strlen(buffer));

  Serial.print("Sent: ");
  Serial.println(buffer);
}

void loop() {
  // ...
}
```

uartWrite()

說明

從 EtherCAT 從站指定的 UART 埠傳送一個位元組的資料。

語法

```
int uartWrite(int dev, uint8_t value);
```

參數

- *[in] int dev*
  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。1 表示 COM2。
- *[in] unit8_t value*
  要傳送的一個位元組資料。

傳回值

傳回實際傳送的位元組數。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  slave.uartWrite(0, 'H');
  slave.uartWrite(0, 'e');
  slave.uartWrite(0, 'l');
  slave.uartWrite(0, 'l');
  slave.uartWrite(0, 'o');
```

```
}

void loop() {
  // ...
}
```

uartReceive()

說明

從 EtherCAT 從站指定的 UART 埠接收多個位元組的資料。

語法

```
int uartReceive(int dev, void *buf, size_t size);
```

參數

- *[in] int dev*
  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。1 表示 COM2。
- *[in] void *buf*
  用於接收資料的緩衝區。
- *[in] size_t size*
  緩衝區的大小。

傳回值

傳回實際接收的位元組數。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

char buffer[256];
int rc;

void CyclicCallback() {
  slave.update();
}

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
```

```
  master.start();
}

void loop() {
  rc = slave.uartReceive(0, buffer, 256);
  for (int i = 0; i < rc; i++) {
    Serial.print(buffer[i]);
  }

  // ...
}
```

uartRead()

說明

從 EtherCAT 從站指定的 UART 埠讀取 1 個位元組的資料。

語法

```
int uartRead(int dev);
```

參數

- *[in] int dev*

  EtherCAT 從站上的指定 UART 埠。0 表示 COM1。1 表示 COM2。

傳回值

傳回 1 個位元組的資料。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  int ch = slave.uartRead(0);
  if (ch >= 0)
```

```
    Serial.print((char)ch);


  // ...
}
```

LCM 函式

適用於 EthercatDevice_DmpHID_Generic 類別的 LCM 存取函式。

函式：

- lcmHeight()
- lcmWidth()
- lcmWordWrap()
- lcmGotoXY()
- lcmClear()
- lcmPrint()
- lcmWrite()

lcmHeight()

說明

取得 EtherCAT 從站上 LCM 的高度。

語法

`int lcmHeight();`

參數

無。

傳回值

傳回 LCM 的高度。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  Serial.print("LCM Height: ");
  Serial.println(slave.lcmHeight());
}

void loop() {
  // ...
}
```

lcmWidth()

說明

取得 EtherCAT 從站上 LCM 的寬度。

語法

`int lcmWidth();`

參數

無。

傳回值

傳回 LCM 的寬度。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  Serial.print("LCM Width: ");
  Serial.println(slave.lcmWidth());
}

void loop() {
  // ...
}
```

lcmWordWrap()

說明

啟用或停用 EtherCAT 從站上 LCM 的自動換行功能。

語法

```
int lcmWordWrap(bool wrap);
```

參數

- *[in] bool wrap*

  一個布林值，用來指定是否啟用自動換行功能：

  1. true：啟用自動換行
  2. false：停用自動換行

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

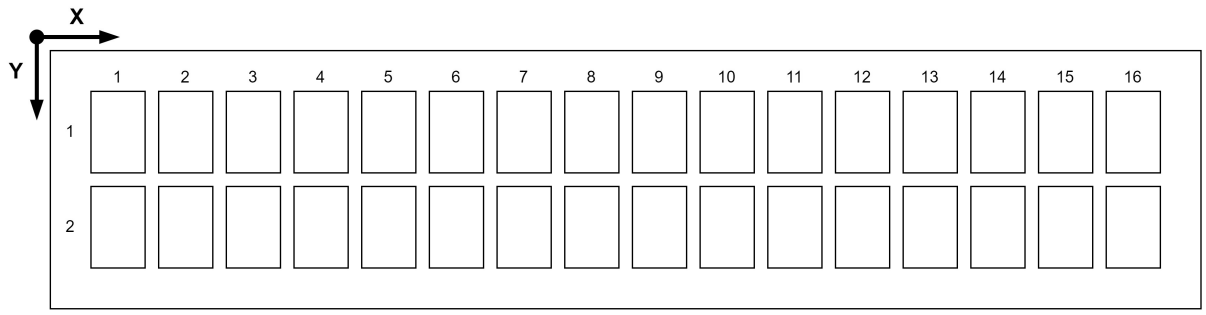
範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.lcmWordWrap(true);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  slave.lcmPrint("Hi, this is QEC HID slave.");
}

void loop() {
  // ...
}
```

lcmGotoXY()

說明

將 EtherCAT 從站上 LCM 的游標移動到指定的座標位置。



語法

```
int lcmGotoXY(int x, int y);
```

參數

- *[in] int x*
  LCM 上的 X 軸位置。
- *[in] int y*
  LCM 上的 Y 軸位置。

傳回值

回傳目前 LCM 游標的位置。游標位置 P_{cursor} 的計算公式如下，其中 W 為 LCM 的寬度：

$P_{cursor} = W \times (Y - 1) + (X - 1)$

若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  master.begin();
```

```
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  slave.lcmGotoXY(1, 1);
  slave.lcmPrint("Hello");
  slave.lcmGotoXY(8, 1);
  slave.lcmPrint("World!");
  slave.lcmGotoXY(2, 2);
  slave.lcmPrint("QEC HID slave");
}

void loop() {
  // ...
}
```

lcmClear()

說明

清除 EtherCAT 從站上 LCM 的畫面。

語法

```
int lcmClear();
```

參數

無。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  slave.lcmClear();
}

void loop() {
  // ...
}
```

lcmPrint()

說明

將指定的字串列印至 EtherCAT 從站的 LCM 螢幕上。

語法

```
int lcmPrint(const char *fmt, ...);
```

參數

- *[in] const char *fmt*
  要列印在 LCM 螢幕上的字串。這是一個指向以 null 結尾的字串指標，包含輸出的格式說明。格式字串遵循 C 語言中的 printf 函式格式，可插入變數與格式化選項。
- *[in] ...*
  這是一個可變參數，會根據 fmt 字串中的格式說明插入對應的變數值。

傳回值

傳回實際列印至 LCM 螢幕的字元數。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
  slave.lcmPrint("Hello world!");
}

void loop() {
  // ...
}
```

lcmWrite()

說明

將單一字元列印至 EtherCAT 從站的 LCM 螢幕上。

語法

```
int lcmWrite(char c);
```

參數

- *[in] char c*
  要列印至 LCM 螢幕的字元。

傳回值

傳回實際列印至 LCM 螢幕的字元數。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  slave.lcmWrite('H');
  slave.lcmWrite('e');
  slave.lcmWrite('l');
  slave.lcmWrite('l');
  slave.lcmWrite('o');
  slave.lcmWrite('!');
```

```
}

void loop() {
  // ...
}
```

Keypad 函式

適用於 EthercatDevice_DmpHID_Generic 類別的 Keypad 存取函式。

函式：

- keypadSetTimeout()
- keypadClear()
- keypadRead()

keypadSetTimeout()

說明

設定 EtherCAT 從站的鍵盤輸入資料緩衝區的逾時時間。如果在指定時間內未讀取到任何鍵盤輸入資料，系統將自動清除該資料緩衝區。預設逾時時間為 1000 毫秒。

語法

```
int keypadSetTimeout(uint32_t timeout_ms);
```

參數

- *[in] uint32_t timeout_ms*
  逾時時間（以毫秒為單位）。若此參數設為 0，則表示鍵盤輸入資料緩衝區不會自動清除。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.keypadSetTimeout(3000);
}

void loop() {
  // ...
}
```

keypadClear()

說明

清除 EtherCAT 從站鍵盤的輸入資料緩衝區。

語法

```
int keypadClear();
```

參數

無。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.keypadClear();
}

void loop() {
  // ...
}
```

keypadRead()

說明

從 EtherCAT 從站裝置的鍵盤讀取輸入字元。

語法

`int keypadRead();`

參數

無。

傳回值

讀取一個來自鍵盤的字元。若無可用資料，則傳回 '\0'。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，並可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  int ch;
  if ((ch = slave.keypadRead()) != '\0') {
```

```
    Serial.print((char)ch);
  }

  // ...
}
```

MPG 函式

適用於 EthercatDevice_DmpHID_Generic 類別的 MPG (Manual Pulse Generator) 存取函式。

函式：

- mpgSetCallback()
- mpgSetNoiseFilter()
- mpgInvertEncoderDirection()
- mpgWriteEncoderRaw()
- mpgWriteEncoder()
- mpgReadEncoderRaw()
- mpgReadEncoder()
- mpgReadEmergencyStop()
- mpgReadEnableSwitch()
- mpgReadAxis()
- mpgReadRatio()

mpgSetCallback()

說明

註冊 EtherCAT 從站裝置的 MPG 事件 callback 函式。

語法

```
int mpgSetCallback(void (*callback)(int));
```

參數

- *[in] void (*callback)(int)*
  要註冊的 MPG 事件 callback 函式，帶有一個整數型參數，用以表示事件類型。支援的 MPG 事件類型如下：

定義	代碼	說明
ECAT_MPG_AXIS_CHANGE	1	軸選旋鈕狀態改變
ECAT_MPG_RATIO_CHANGE	2	比例旋鈕狀態改變
ECAT_MPG_EMERGENCY_STOP_CHANGE	3	急停開關狀態改變
ECAT_MPG_ENABLE_SWITCH_CHANGE	4	啟動開關狀態改變
ECAT_MPG_ENCODER_CHANGE	5	編碼器的邏輯計數器改變

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞，且可在 callback 函式中呼叫。由於該回呼函式會在 `update()` 函式中觸發，如果 `update()` 是在中斷回呼中被呼叫，則使用時必須遵守特定限制。詳細內容請參閱 Callback 函式。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void MpgCallback(int event) {
  switch (event) {
    case ECAT_MPG_AXIS_CHANGE:
      Serial.print("MPG Axis changed. ");
      Serial.println(slave.mpgReadAxis());
      break;
    case ECAT_MPG_RATIO_CHANGE:
      Serial.print("MPG Ratio changed. ");
      Serial.println(slave.mpgReadRatio());
```

```
      break;
    case ECAT_MPG_EMERGENCY_STOP_CHANGE:
      Serial.print("MPG Emergency Stop changed ");
      Serial.println(slave.mpgReadEmergencyStop());
      break;
    case ECAT_MPG_ENABLE_SWITCH_CHANGE:
      Serial.print("MPG Enable Switch changed. ");
      Serial.println(slave.mpgReadEnableSwitch());
      break;
    case ECAT_MPG_ENCODER_CHANGE:
      Serial.print("MPG Encoder changed. ");
      Serial.println(slave.mpgReadEncoder());
      break;
  }
}
void setup() {
  Serial.begin(115200);
  master.begin();
  slave.attach(0, master);
  slave.mpgSetCallback(MpgCallback);
  master.start();
}
void loop() {
  slave.update();
}
```

mpgSetNoiseFilter()

說明

設定 EtherCAT 從站中 MPG 相關元件的雜訊濾波器，包括急停開關、軸選旋鈕與比例旋鈕的雜訊過濾時間。

語法

```
int mpgSetNoiseFilter(uint32_t time_us);
```

參數

- *[in] uint32_t time_us*
  The desired configuration time for the noise filter.

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.mpgSetNoiseFilter(10000);
  master.start();
}

void loop() {
  slave.update();
}
```

mpgInvertEncoderDirection()

說明

反轉 EtherCAT 從站裝置中 MPG 編碼器的計數方向。

語法

```
int mpgInvertEncoderDirection(bool invert);
```

參數

- *[in] bool invert*

  一個布林值，用於指定是否要反轉編碼器的計數方向。

  a. true：計數方向將被反轉。

  b. false：計數方向維持不變。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.mpgInvertEncoderDirection(true);
  master.start();
}

void loop() {
  slave.update();
  // ...
}
```

mpgWriteEncoderRaw()

說明

將原始資料寫入 EtherCAT 從站裝置中 MPG 的編碼器。

語法

```
int mpgWriteEncoderRaw(int32_t value);
```

參數

- `[in] int32_t value`
  要寫入編碼器的原始資料值。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.mpgWriteEncoderRaw(100);
  master.start();
}

void loop() {
  slave.update();
  // ...
}
```

mpgWriteEncoder()

說明

將邏輯計數器值寫入 EtherCAT 從站裝置中 MPG 的編碼器。

語法

```
int mpgWriteEncoder(int32_t value);
```

參數

- *[in] int32_t value*
  要寫入編碼器的邏輯計數器值。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.mpgWriteEncoder(100);
  master.start();
}

void loop() {
  slave.update();
  // ...
}
```

mpgReadEncoderRaw()

說明

讀取 EtherCAT 從站裝置中 MPG 編碼器的原始資料。

語法

```
int32_t mpgReadEncoderRaw();
```

參數

無。

傳回值

回傳編碼器的原始資料值。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

int32_t raw = 0;

void CyclicCallback() {
  slave.update();
}

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  int32_t newRaw = slave.mpgReadEncoderRaw();
```

```
  if (raw != newRaw) {
    raw = newRaw;
    Serial.print("Raw: ");
    Serial.println(raw);
  }
  // ...
}
```

mpgReadEncoder()

說明

讀取 EtherCAT 從站裝置中 MPG 編碼器的邏輯計數值。

語法

`int32_t mpgReadEncoder();`

參數

無。

傳回值

回傳一個 32 位元有號整數，表示邏輯計數器的值。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

int32_t encoder = 0;

void CyclicCallback() {
  slave.update();
}

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  int32_t newEncoder = slave.mpgReadEncoder();
```

```
  if (encoder != newEncoder) {
    encoder = newEncoder;
    Serial.print("Encoder: ");
    Serial.println(encoder);
  }

  // ...
}
```

mpgReadEmergencyStop()

說明

讀取 EtherCAT 從站裝置中 MPG 的緊急停止訊號值。

語法

```
int mpgReadEmergencyStop();
```

參數

無。

傳回值

回傳緊急停止訊號的值：

- 1 表示緊急停止訊號為 高電位 (HIGH)
- 0 表示緊急停止訊號為 低電位 (LOW)

若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  Serial.print("Emergency Stop: ");
```

```
  Serial.println(slave.mpgReadEmergencyStop());
  delay(1000);
  // ...
}
```

mpgReadEnableSwitch()

說明

讀取 EtherCAT 從站裝置中 MPG 的啟用開關訊號值。

語法

```
int mpgReadEnableSwitch();
```

參數

無。

傳回值

回傳啟用開關的訊號值：

- 1 表示啟用開關訊號為 高電位 (HIGH)
- 0 表示啟用開關訊號為 低電位 (LOW)

若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  Serial.print("Enable: ");
```

```
  Serial.println(slave.mpgReadEnableSwitch());
  delay(1000);
  // ...
}
```

mpgReadAxis()

說明

讀取 EtherCAT 從站裝置中 MPG 的軸選擇旋鈕值。

語法

```
int mpgReadAxis();
```

參數

無。

傳回值

回傳 MPG 軸旋鈕的目前值：

- 0: off.
- 1: X-axis.
- 2: Y-axis.
- 3: Z-axis.
- 4: 4-axis.
- 5: 5-axis.
- 6: 6-axis.

若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
```

```
  master.start();
}

void loop() {
  Serial.print("Axis: ");
  Serial.println(slave.mpgReadAxis());
  delay(1000);
  // ...
}
```

mpgReadRatio()

說明

讀取 EtherCAT 從站裝置中 MPG 的倍率旋鈕設定值。

語法

```
int mpgReadRatio();
```

參數

無。

傳回值

回傳 MPG 的倍率設定值：

- 0: 1x.
- 1: 10x.
- 2: 100x.

若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
```

```
  Serial.print("Ratio: ");
  Serial.println(slave.mpgReadRatio());
  delay(1000);
  // ...
}
```

Buzzer 函式

適用於 EthercatDevice_DmpHID_Generic 類別的蜂鳴器 (Buzzer) 存取函式。

函式：

- buzzer()

buzzer()

說明

從 EtherCAT 從站的蜂鳴器發出指定頻率的聲音。

語法

```
int buzzer(uint32_t hz, uint32_t duration_ms = 0);
```

參數

- *[in] uint32_t hz*
  聲音的頻率（單位為 Hz）。若此參數設為 0，表示停止發聲。若頻率值超過 100,000（100K），本函式庫將自動限制為 100K。
- *[in] uint32_t duration_ms*
  聲音持續時間（單位為毫秒）。若此參數省略或設為 0，聲音將持續不斷，直到下一次呼叫停止為止。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

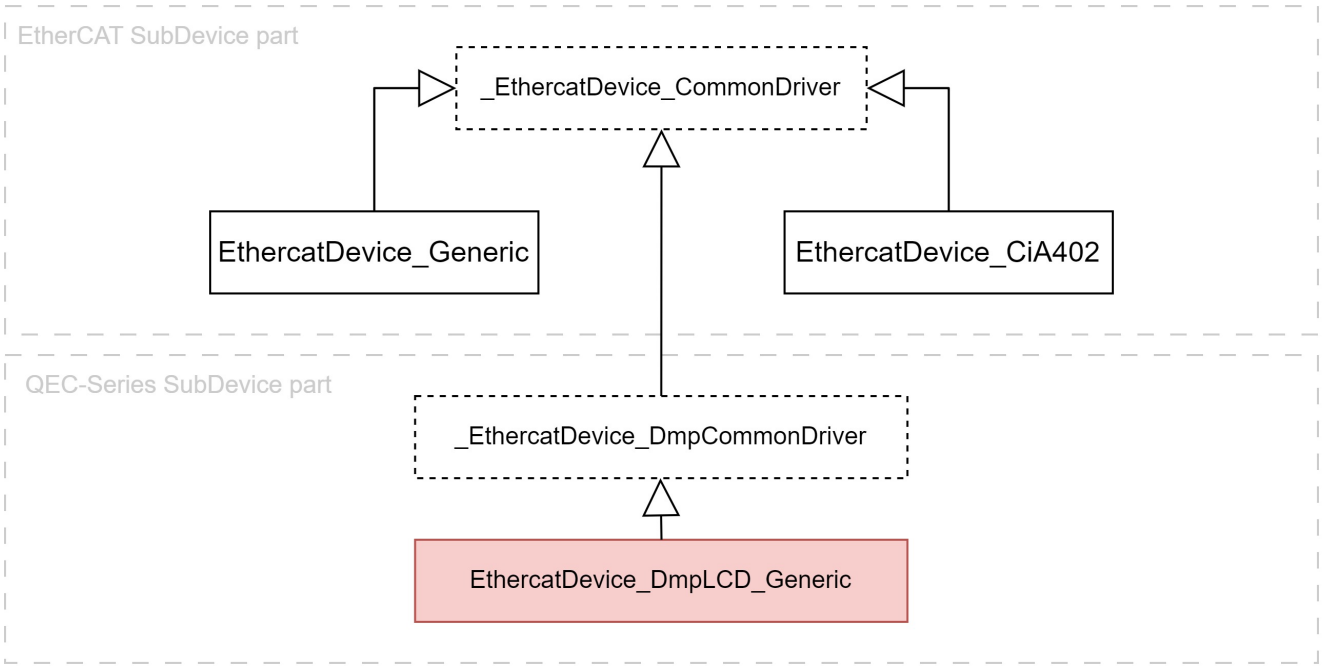
```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  slave.buzzer(3000);
  delay(1000);
  slave.buzzer(500);
```

```
  delay(1000);
  slave.buzzer(0);
  delay(1000);
}

void loop() {
  // ...
}
```

2.3.6 EthercatDevice_DmpLCD_Generic

EthercatDevice_DmpLCD_Generic 是 ICOP 為 QEC 系列 LCD EtherCAT 從站模組所專門開發的從站裝置類別。它提供一組通用的圖形繪製基本函式（如：點、線、圓、矩形、文字等），可用於各類顯示器上進行圖形繪製。此外，它也提供與觸控螢幕相關的操作函式，用來判斷觸控螢幕上被觸碰的位置。

EthercatDevice_DmpLCD_Generic 的類別關係如下圖所示：



- *EthercatDevice_DmpLCD_Generic* 繼承自 *_EthercatDevice_DmpCommonDriver*.

基礎類別：

- _EthercatDevice_CommonDriver

衍生類別：

類別名稱	Vendor ID	Product Code
EthercatDevice_QECR11UN01	0x00000bc3	0x0086d103
EthercatDevice_QECR00UN01	0x00000bc3	0x0086d100

函式分類：

- 初始化
- 控制
- LCD
- 觸控螢幕

函式：

函式名稱	說明	Callback 中呼叫
初始化相關函式		
attach()	初始化此 EtherCAT 從站裝置類別的物件	
detach()	解除初始化此 EtherCAT 從站裝置類別的物件	
控制相關函式		
update()	更新各功能模組的狀態機與內部變數	O
LCD 存取函式		
lcdInit()	初始化 LCD 模組	
lcdFlush()	等待所有圖形指令執行完成	
lcdWidth()	取得顯示器的寬度	O
lcdHeight()	取得顯示器的高度	O
lcdGetRotation()	取得顯示器的旋轉角度	O
lcdSetRotation()	設定顯示器的旋轉角度	O
lcdDrawPixel()	繪製單一像素	O
lcdDrawFastHLine()	繪製水平線	O
lcdDrawFastVLine()	繪製垂直線	O
lcdDrawLine()	繪製任意斜率的線條	O
lcdFillRect()	繪製並填滿矩形	O
lcdDrawRect()	繪製矩形	O
lcdFillCircle()	繪製並填滿圓形	O
lcdDrawCircle()	繪製圓形	O
lcdFillTriangle()	繪製並填滿三角形	O
lcdDrawTriangle()	繪製三角形	O
lcdFillRoundRect()	繪製並填滿圓角矩形	O
lcdDrawRoundRect()	繪製圓角矩形	O
lcdFillScreen()	填滿整個 LCD 畫面	O
lcdSetAddrWindow()	設定繪圖區域，座標視窗	O
lcdPushColors()	傳送一組 16-bit 顏色資料陣列進行繪圖	O
lcdSetTextCursor()	將文字游標移至指定像素位置	O
lcdSetTextWrap()	啟用或停用文字自動換行功能	O
lcdSetTextColor()	設定要列印文字的顏色	O
lcdSetTextSize()	設定要列印文字的大小	O
lcdPrint()	列印文字字串	O
觸控螢幕函式		
isTouched()	取得目前觸控點	O
touchX()	讀取觸控點的 X 軸座標	O

touchY()	讀取觸控點的 Y 軸座標	O
touchCalibration()	執行觸控校正	O

LCD 支援模組：

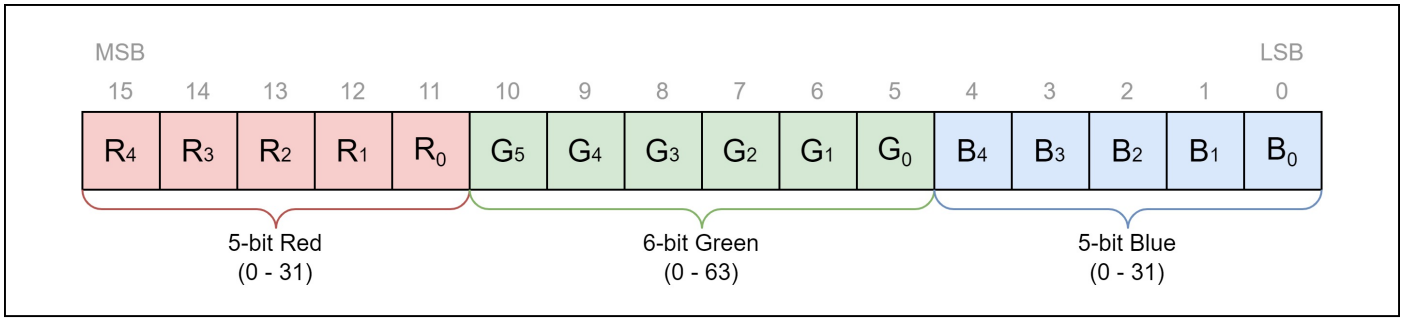
ID	LCM Driver IC	Resolution	Xp	Yp	Xm	Ym
0	ILI9341	240 X 320	D9	A2	A3	D8
1	ILI9341	240 X 320	D6	A1	A2	D7
2	ILI9488	320 X 480	D8	A3	A2	D9
3	ILI9486	320 X 480	X	X	X	X
4	HX8347-I(T)	240 X 320	D9	A2	A3	D8
5	HX8347-D	240 X 320	D9	A2	A3	D8

關於 RGB565

RGB565 是一種用來表示影像像素色彩資訊的色彩格式。它使用 16 位元 (2 個位元組) 來編碼單一像素的顏色資料。名稱「RGB565」代表了三原色 (紅、綠、藍) 在這 16 位元中的分配方式：

- **紅色 (R)**: 5 bits
- **綠色 (G)**: 6 bits
- **藍色 (B)**: 5 bits

這種格式將更多的位元分配給綠色，是因為人眼對綠色的變化比紅色和藍色更敏感。



由於每個色彩分量所分配的位元較少，RGB565 能呈現的色彩數量較 24-bit RGB 或 16-bit RGB 少。它可以表示紅色和藍色的 $2^5 = 32$ 種可能值，以及綠色的 $2^6 = 64$ 種可能值，總共為 $32 \times 64 \times 32 = 65{,}536$ 種顏色。

以下是一些常見 RGB565 顏色代碼，便於參考與測試：

顏色	代碼	顏色樣本
Black	0x0000	
Blue	0x001F	
Red	0xF800	
Green	0x07E0	
Cyan	0x07FF	
Magenta	0xF81F	
Yellow	0xFFE0	
White	0xFFFF	

RGB888 轉換為 RGB565

從 24 位元 RGB 轉換為 RGB565 涉及一個稱為顏色量化的過程。與 24 位元和 16 位元 RGB 之間的轉換類似，此過程會減少顏色數量並為它們分配 RGB565 調色板中最接近的可用顏色。這可能會造成一些色彩損失，但對於某些應用來說，色彩準確度和效率之間的權衡可能是可以接受的。以下是用 C 語言將 RGB888 轉換為 RGB565 的範例。

```
uint16_t rgb888_to_rgb565(uint8_t r, uint8_t g, uint8_t b)
{
    return ((r >> 3) << 11) | ((g >> 2) << 5) | (b >> 3);
}
```

初始化函式

適用於 EthercatDevice_DmpLCD_Generic 類別的初始化相關函式。

函式：

- attach()
- detach()

attach()

說明

此函式的行為與 `EthercatDevice Generic::attach()` 相同。

詳細的說明、語法、參數與傳回值，請參閱該章節。

範例

適用於 EthercatDevice_QECR11UN01.

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11UN01 slave;

void setup(void) {
    master.begin();
    slave.attach(0, master);
}
void loop() {
  // ...

}
```

detach()

說明

此函式的行為與 `EthercatDevice Generic::detach()` 相同。

詳細的說明、語法、參數與傳回值，請參閱該章節。

範例

適用於 EthercatDevice_QECR00UN01.

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00UN01 slave;

void setup() {
  master.begin();
  slave.attach(0, master);

  delay(3000);

  slave.detach();
  master.end();
}

void loop() {
  // ...
}
```

控制函式

適用於 EthercatDevice_DmpLCD_Generic 類別的控制相關函式。

函式：

- update()

update()

說明

更新 EtherCAT 從站上各功能的狀態機與內部變數。

語法

```
int update();
```

參數

無。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00UN01 slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  master.start();
}

void loop() {
  slave.update();
  // ...
}
```

LCD 函式

適用於 EthercatDevice_DmpLCD_Generic 類別的 LCD 存取函式。

函式：

- lcdInit()
- lcdFlush()
- lcdWidth()
- lcdHeight()
- lcdGetRotation()
- lcdSetRotation()
- lcdDrawPixel()
- lcdDrawFastHLine()
- lcdDrawFastVLine()
- lcdDrawLine()
- lcdFillRect()
- lcdDrawRect()
- lcdFillCircle()
- lcdDrawCircle()
- lcdFillTriangle()
- lcdDrawTriangle()
- lcdFillRoundRect()
- lcdDrawRoundRect()
- lcdFillScreen()
- lcdSetAddrWindow()
- lcdPushColors()
- lcdSetTextCursor()
- lcdSetTextColor()
- lcdSetTextSize()
- lcdSetTextWrap()
- lcdPrint()

lcdInit()

說明

初始化 EtherCAT 從站上的 LCD 模組。

語法

```
int lcdInit(uint16_t lcd_id = ECAT_LCD_UNKNOWN_ID);
```

參數

- *[in] uint16_t lcd_id*

  欲初始化的 LCD 模組 ID。支援的 LCD 模組與其觸控腳位（Xp、Yp、Xm、Ym）配置如下表所示

定義	ID	IC	解析度	Xp	Yp	Xm	Ym
ECAT_LCD_ILI9341_1	0	ILI9341	240 X 320	D9	A2	A3	D8
ECAT_LCD_ILI9341_2	1	ILI9341	240 X 320	D6	A1	A2	D7
ECAT_LCD_ILI9488_1	2	ILI9488	320 X 480	D8	A3	A2	D9
ECAT_LCD_ILI9486_1	3	ILI9486	320 X 480	-	-	-	-
ECAT_LCD_HX8347I_1	4	HX8347-I(T)	240 X 320	D9	A2	A3	D8
ECAT_LCD_HX8347D_1	5	HX8347-D	240 X 320	D9	A2	A3	D8
ECAT_LCD_UNKNOWN_ID	0xFFFF	-	-	-	-	-	-

  若此參數設定為 ECAT_LCD_UNKNOWN_ID，則會從 EtherCAT 從站裝置的儲存空間中載入 LCD 模組 ID 與校正參數。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00UN01 slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.lcdInit(ECAT_LCD_ILI9341_1);
  // ...
}
```

```
void loop() {
  // ...
}
```

lcdFlush()

說明

等待 EtherCAT 從站上的 LCD 所有圖形指令執行完成。

語法

```
int lcdFlush();
```

參數

無。

傳回值

回傳 1：表示所有圖形指令皆已執行完成。回傳 0：表示仍有圖形指令尚未完成執行。

若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00UN01 slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.lcdInit(ECAT_LCD_ILI9341_1);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  // Graphics commands.
  // ...
  slave.lcdFlush();
}
```

lcdWidth()

說明

取得 EtherCAT 從站上 LCD 顯示器的寬度。此寬度會依照 `lcdSetRotation()` 的設定而旋轉變化。

語法

```
int16_t lcdWidth();
```

參數

無。

傳回值

回傳 LCD 的顯示寬度。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00UN01 slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  slave.lcdInit(ECAT_LCD_ILI9341_1);

  Serial.print("LCD Width: ");
  Serial.println(slave.lcdWidth());
}

void loop() {
  // ...
}
```

lcdHeight()

說明

取得 EtherCAT 從站上 LCD 顯示器的高度。此高度會依照 `lcdSetRotation()` 的設定而旋轉變化。

語法

```
int16_t lcdHeight();
```

參數

無。

傳回值

回傳 LCD 的顯示高度。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00UN01 slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  slave.lcdInit(ECAT_LCD_ILI9341_1);

  Serial.print("LCD Height: ");
  Serial.println(slave.lcdHeight());
}

void loop() {
  // ...
}
```

lcdGetRotation()

說明

取得 EtherCAT 從站上 LCD 顯示器目前的顯示旋轉方向。

語法

```
uint8_t lcdGetRotation();
```

參數

無。

傳回值

回傳 LCD 顯示器目前的旋轉方向。關於旋轉模式的詳細資訊，請參考 `lcdSetRotation()`。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

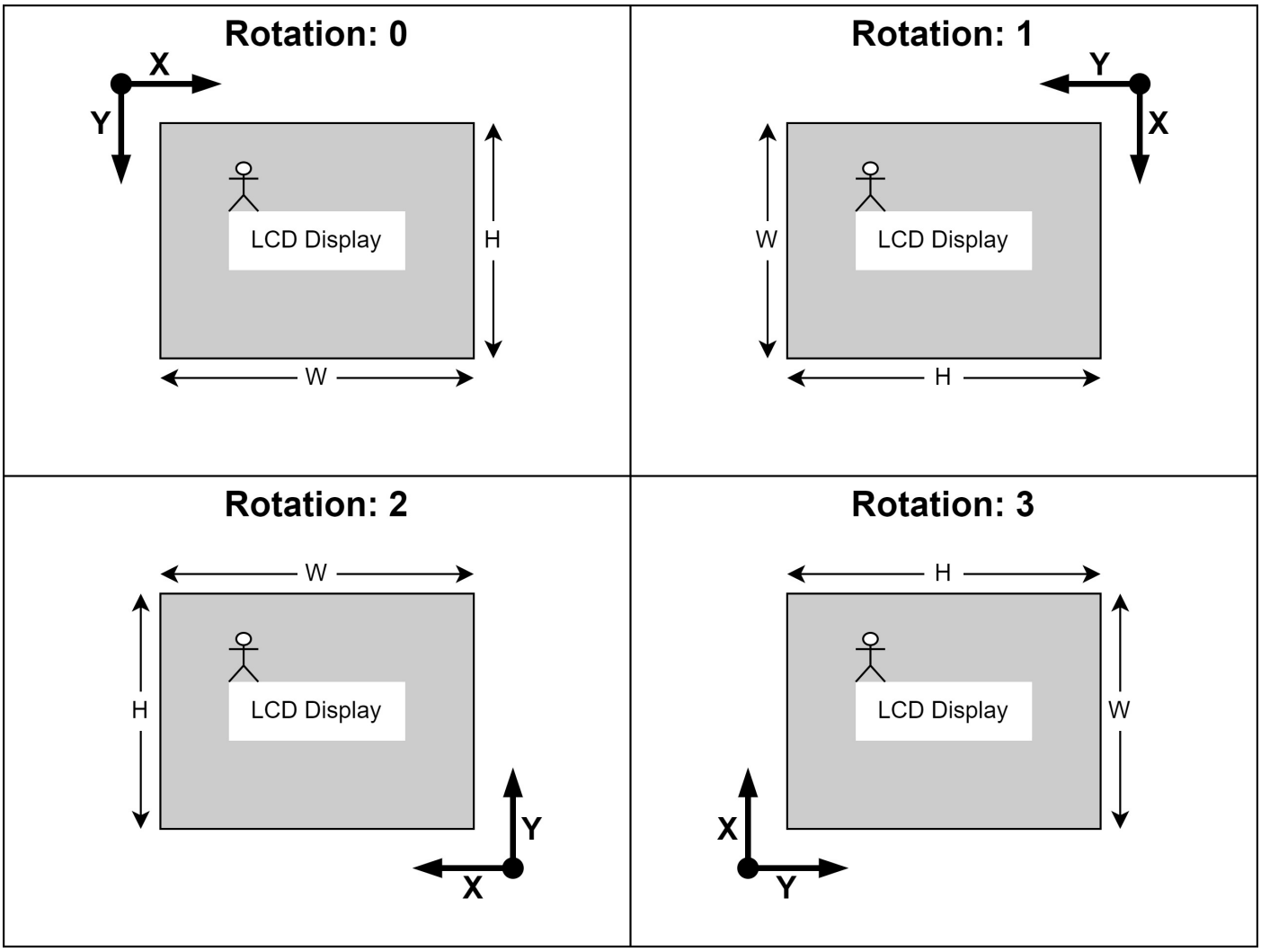
範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00UN01 slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  slave.lcdInit(ECAT_LCD_ILI9341_1);

  Serial.print("LCD Rotation: ");
  Serial.println(slave.lcdGetRotation());
}

void loop() {
  // ...
}
```

lcdSetRotation()

說明

設定 EtherCAT 從站上 LCD 顯示器的旋轉方向。關於旋轉模式的詳細說明，請參考下圖。



語法

```
int lcdSetRotation(uint8_t x);
```

參數

- `[in] uint8_t x`
  欲設定的旋轉模式。由於僅支援四種旋轉模式，本函式會對輸入參數 x 進行位元運算，只保留最低的兩個位元，其餘清為 0。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

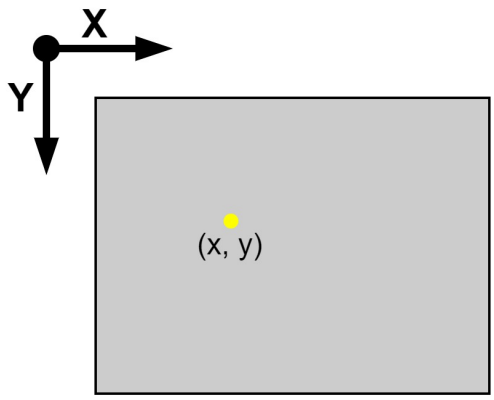
```
#include "Ethercat.h"
```

```
EthercatMaster master;
EthercatDevice_QECR00UN01 slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.lcdInit(ECAT_LCD_ILI9341_1);
  slave.lcdSetRotation(1);
}

void loop() {
  // ...
}
```

lcdDrawPixel()

說明

在 EtherCAT 從站的 LCD 顯示器上繪製指定顏色的單一像素點。



語法

```
int lcdDrawPixel(int16_t x, int16_t y, uint16_t color);
```

參數

- *[in] int16_t x*
  欲繪製像素的 X 軸座標。
- *[in] int16_t y*
  欲繪製像素的 Y 軸座標。
- *[in] uint16_t color*
  欲繪製像素的顏色，為 16 位元無號整數，使用 RGB565 格式來編碼顏色資訊。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

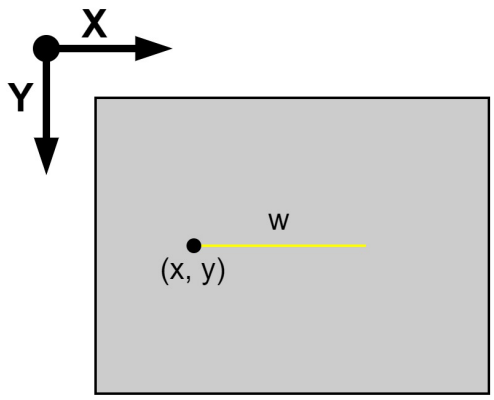
```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00UN01 slave;

void CyclicCallback() {
  slave.update();
}
```

```
void setup() {
  master.begin();
  slave.attach(0, master);
  slave.lcdInit(ECAT_LCD_ILI9341_1);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  slave.lcdDrawPixel(100, 100, 0xFFE0);
}

void loop() {
  // ...
}
```

lcdDrawFastHLine()

說明

在 EtherCAT 從站的 LCD 顯示器上繪製一條指定顏色的水平線。



語法

```
int lcdDrawFastHLine(int16_t x, int16_t y, int16_t w, uint16_t color);
```

參數

- *[in] int16_t x*
  水平線起點的 X 軸座標。
- *[in] int16_t y*
  水平線起點的 Y 軸座標。
- *[in] int16_t w*
  水平線的長度。
- *[in] uint16_t color*
  欲繪製像素的顏色，為 16 位元無號整數，使用 RGB565 格式來編碼顏色資訊。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

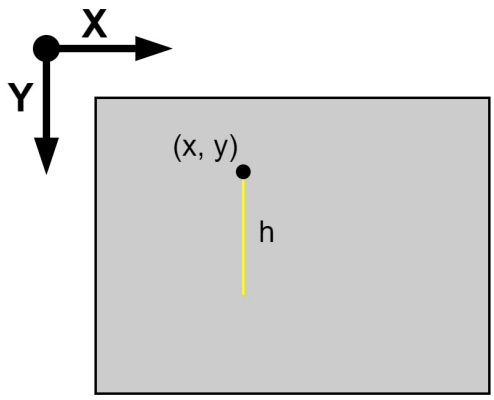
```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00UN01 slave;

void CyclicCallback() {
  slave.update();
```

```
}

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.lcdInit(ECAT_LCD_ILI9341_1);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  slave.lcdDrawFastHLine(100, 100, 100, 0xFFE0);
}

void loop() {
  // ...
}
```

lcdDrawFastVLine()

說明

在 EtherCAT 從站的 LCD 顯示器上繪製一條指定顏色的垂直線。



語法

```
int lcdDrawFastVLine(int16_t x, int16_t y, int16_t h, uint16_t color);
```

參數

- *[in] int16_t x*
  垂直線起點的 X 軸座標。
- *[in] int16_t y*
  垂直線起點的 Y 軸座標。
- *[in] int16_t h*
  垂直線的長度。
- *[in] uint16_t color*
  欲繪製像素的顏色，為 16 位元無號整數，使用 RGB565 格式來編碼顏色資訊。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

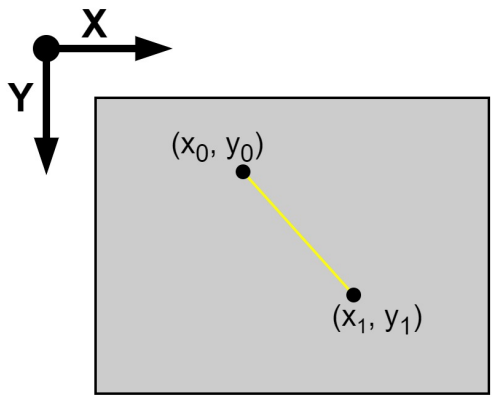
```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00UN01 slave;

void CyclicCallback() {
  slave.update();
```

```
}

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.lcdInit(ECAT_LCD_ILI9341_1);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  slave.lcdDrawFastVLine(100, 100, 100, 0xFFE0);
}

void loop() {
  // ...
}
```

lcdDrawLine()

說明

在 EtherCAT 從站的 LCD 顯示器上繪製一條指定顏色的直線。



語法

```
int lcdDrawLine(int16_t x0, int16_t y0, int16_t x1, int16_t y1, uint16_t color);
```

參數

- *[in] int16_t x0*
  線段起點的 X 軸座標。
- *[in] int16_t y0*
  線段起點的 Y 軸座標。
- *[in] int16_t x1*
  線段終點的 X 軸座標。
- *[in] int16_t y1*
  線段終點的 Y 軸座標。
- *[in] uint16_t color*
  欲繪製像素的顏色，為 16 位元無號整數，使用 RGB565 格式來編碼顏色資訊。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

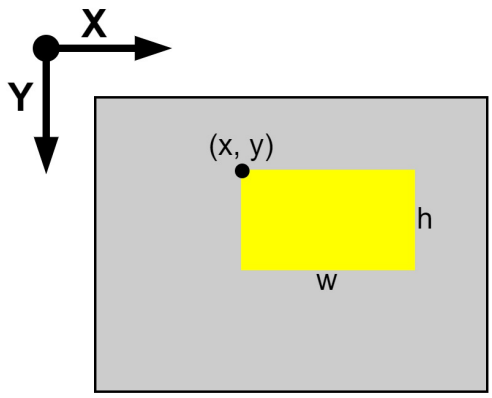
```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00UN01 slave;
```

```
void CyclicCallback() {
  slave.update();
}

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.lcdInit(ECAT_LCD_ILI9341_1);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  slave.lcdDrawLine(100, 100, 200, 200, 0xFFE0);
}

void loop() {
  // ...
}
```

lcdFillRect()

說明

在 EtherCAT 從站的 LCD 顯示器上繪製一個指定顏色的矩形框，並以相同顏色填滿該矩形。



語法

```
int lcdFillRect(int16_t x, int16_t y, int16_t w, int16_t h, uint16_t color);
```

參數

- *[in] int16_t x*
  矩形起點的 X 軸座標。
- *[in] int16_t y*
  矩形起點的 Y 軸座標。
- *[in] int16_t w*
  欲繪製矩形的寬度。
- *[in] int16_t h*
  欲繪製矩形的高度。
- *[in] uint16_t color*
  欲繪製像素的顏色，為 16 位元無號整數，使用 RGB565 格式來編碼顏色資訊。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

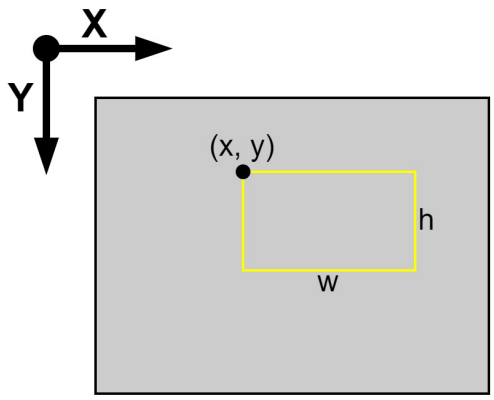
```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00UN01 slave;
```

```
void CyclicCallback() {
  slave.update();
}

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.lcdInit(ECAT_LCD_ILI9341_1);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  slave.lcdFillRect(100, 100, 100, 100, 0xFFE0);
}

void loop() {
  // ...
}
```

lcdDrawRect()

說明

在 EtherCAT 從站的 LCD 顯示器上，以指定的顏色繪製矩形框線（不填滿內容）。



語法

```
int lcdDrawRect(int16_t x, int16_t y, int16_t w, int16_t h, uint16_t color);
```

參數

- *[in] int16_t x*
  矩形起點的 X 軸座標。
- *[in] int16_t y*
  矩形起點的 Y 軸座標。
- *[in] int16_t w*
  欲繪製矩形的寬度。
- *[in] int16_t h*
  欲繪製矩形的高度。
- *[in] uint16_t color*
  欲繪製像素的顏色，為 16 位元無號整數，使用 RGB565 格式來編碼顏色資訊。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

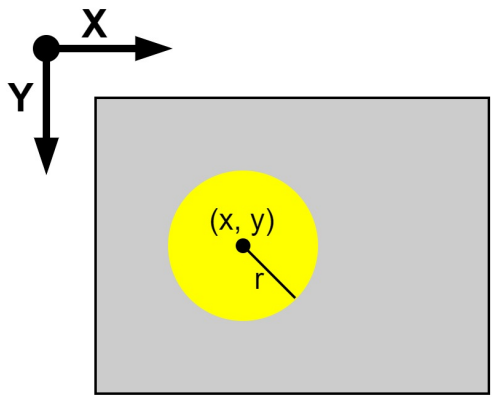
```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00UN01 slave;
```

```
void CyclicCallback() {
  slave.update();
}

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.lcdInit(ECAT_LCD_ILI9341_1);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  slave.lcdDrawRect(100, 100, 100, 100, 0xFFE0);
}

void loop() {
  // ...
}
```

lcdFillCircle()

說明

在 EtherCAT 從站的 LCD 顯示器上，以指定的顏色繪製圓形邊框，並以相同顏色填滿圓形內部。



語法

```
int lcdFillCircle(int16_t x, int16_t y, int16_t r, uint16_t color);
```

參數

- *[in] int16_t x*
  圓心的 X 軸座標。
- *[in] int16_t y*
  圓心的 Y 軸座標。
- *[in] int16_t r*
  圓的半徑。
- *[in] uint16_t color*
  欲繪製像素的顏色，為 16 位元無號整數，使用 RGB565 格式來編碼顏色資訊。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

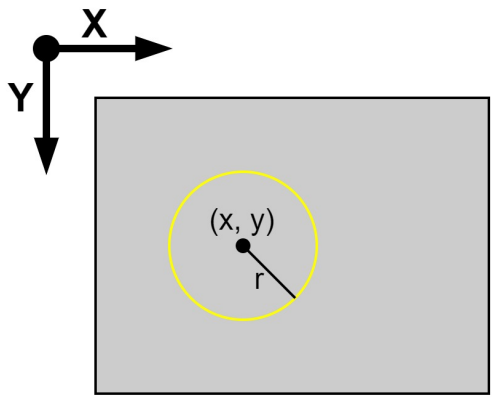
```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00UN01 slave;

void CyclicCallback() {
  slave.update();
```

```
}

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.lcdInit(ECAT_LCD_ILI9341_1);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  slave.lcdFillCircle(100, 100, 50, 0xFFE0);
}

void loop() {
  // ...
}
```

lcdDrawCircle()

說明

在 EtherCAT 從站的 LCD 顯示器上，以指定的顏色繪製圓形邊框。



語法

```
int lcdDrawCircle(int16_t x, int16_t y, int16_t r, uint16_t color);
```

參數

- *[in] int16_t x*
  圓心的 X 軸座標。
- *[in] int16_t y*
  圓心的 Y 軸座標。
- *[in] int16_t r*
  圓的半徑。
- *[in] uint16_t color*
  欲繪製像素的顏色，為 16 位元無號整數，使用 RGB565 格式來編碼顏色資訊。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

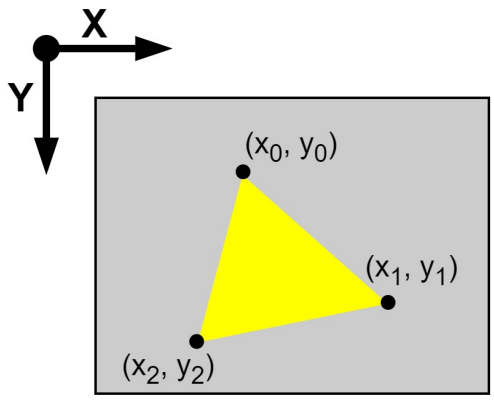
```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00UN01 slave;

void CyclicCallback() {
  slave.update();
```

```
}

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.lcdInit(ECAT_LCD_ILI9341_1);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  slave.lcdDrawCircle(100, 100, 50, 0xFFE0);
}

void loop() {
  // ...
}
```

lcdFillTriangle()

說明

在 EtherCAT 從站的 LCD 顯示器上，以指定的顏色繪製三角形邊框，並以相同顏色填滿三角形。



語法

```
int lcdFillTriangle(int16_t x0, int16_t y0, int16_t x1, int16_t y1, int16_t x2, int16_t y2, uint16_t color);
```

參數

- *[in] int16_t x0*
  第 1 個頂點的 X 軸位置
- *[in] int16_t y0*
  第 1 個頂點的 Y 軸位置
- *[in] int16_t x1*
  第 2 個頂點的 X 軸位置
- *[in] int16_t y1*
  第 2 個頂點的 Y 軸位置
- *[in] int16_t x2*
  第 3 個頂點的 X 軸位置
- *[in] int16_t y2*
  第 3 個頂點的 Y 軸位置
- *[in] uint16_t color*
  欲繪製像素的顏色，為 16 位元無號整數，使用 RGB565 格式來編碼顏色資訊。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

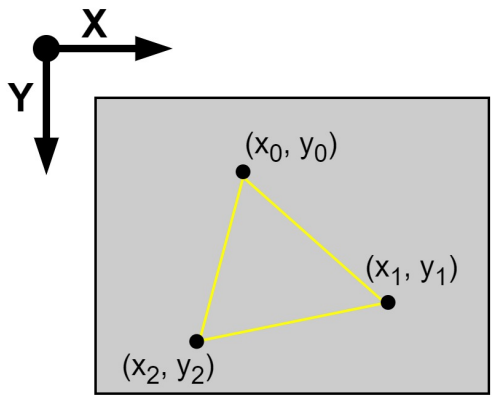
範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00UN01 slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.lcdInit(ECAT_LCD_ILI9341_1);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  slave.lcdFillTriangle(100, 100, 200, 200, 80, 220, 0xFFE0);
}

void loop() {
  // ...
}
```

lcdDrawTriangle()

說明

在 EtherCAT 從站的 LCD 顯示器上，以指定的顏色繪製三角形的邊框。



語法

```
int lcdDrawTriangle(int16_t x0, int16_t y0, int16_t x1, int16_t y1, int16_t x2, int16_t y2, uint16_t color);
```

參數

- *[in] int16_t x0*
  第 1 個頂點的 X 軸位置
- *[in] int16_t y0*
  第 1 個頂點的 Y 軸位置
- *[in] int16_t x1*
  第 2 個頂點的 X 軸位置
- *[in] int16_t y1*
  第 2 個頂點的 Y 軸位置
- *[in] int16_t x2*
  第 3 個頂點的 X 軸位置
- *[in] int16_t y2*
  第 3 個頂點的 Y 軸位置
- *[in] uint16_t color*
  欲繪製像素的顏色，為 16 位元無號整數，使用 RGB565 格式來編碼顏色資訊。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

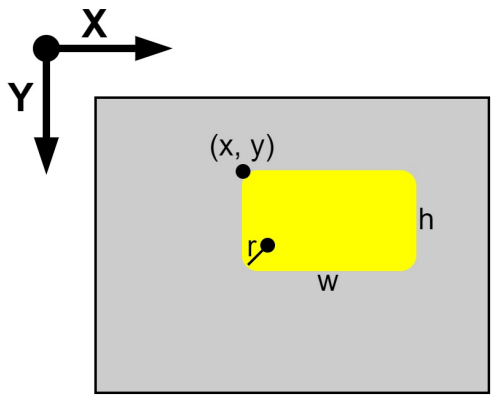
範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00UN01 slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.lcdInit(ECAT_LCD_ILI9341_1);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  slave.lcdDrawTriangle(100, 100, 200, 200, 80, 220, 0xFFE0);
}

void loop() {
  // ...
}
```

lcdFillRoundRect()

說明

在 EtherCAT 從站的 LCD 顯示器上，以指定的顏色繪製一個圓角矩形的邊框，並以相同顏色填滿。



語法

```
int lcdFillRoundRect(int16_t x, int16_t y, int16_t w, int16_t h, int16_t r, uint16_t color);
```

參數

- *[in] int16_t x*
  圓角矩形起點的 X 軸位置。
- *[in] int16_t y*
  圓角矩形起點的 Y 軸位置。
- *[in] int16_t w*
  圓角矩形的寬度。
- *[in] int16_t h*
  圓角矩形的高度。
- *[in] int16_t r*
  圓角的半徑。
- *[in] uint16_t color*
  欲繪製像素的顏色，為 16 位元無號整數，使用 RGB565 格式來編碼顏色資訊。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

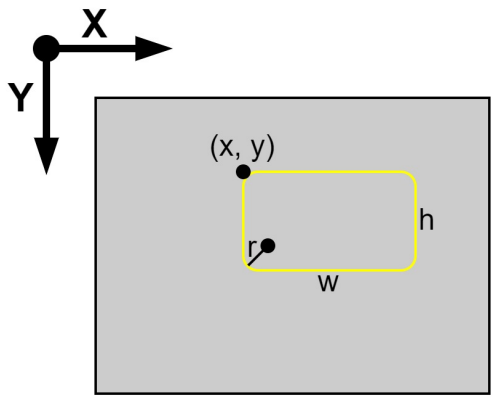
範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00UN01 slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.lcdInit(ECAT_LCD_ILI9341_1);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  slave.lcdFillRoundRect(100, 100, 100, 100, 20, 0xFFE0);
}

void loop() {
  // ...
}
```

lcdDrawRoundRect()

說明

在 EtherCAT 從站的 LCD 顯示器上，以指定的顏色繪製一個圓角矩形的邊框。



語法

```
int lcdDrawRoundRect(int16_t x, int16_t y, int16_t w, int16_t h, int16_t r, uint16_t color);
```

參數

- *[in] int16_t x*
  圓角矩形起點的 X 軸位置。
- *[in] int16_t y*
  圓角矩形起點的 Y 軸位置。
- *[in] int16_t w*
  圓角矩形的寬度。
- *[in] int16_t h*
  圓角矩形的高度。
- *[in] int16_t r*
  圓角的半徑。
- *[in] uint16_t color*
  欲繪製像素的顏色，為 16 位元無號整數，使用 RGB565 格式來編碼顏色資訊。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"
```

```
EthercatMaster master;
EthercatDevice_QECR00UN01 slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.lcdInit(ECAT_LCD_ILI9341_1);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  slave.lcdDrawRoundRect(100, 100, 100, 100, 20, 0xFFE0);
}

void loop() {
  // ...
}
```

lcdFillScreen()

說明

將 EtherCAT 從站上的整個 LCD 顯示區域，以指定顏色填滿。

語法

```
int lcdFillScreen(uint16_t color);
```

參數

- *[in] uint16_t color*
  欲繪製像素的顏色，為 16 位元無號整數，使用 RGB565 格式來編碼顏色資訊。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

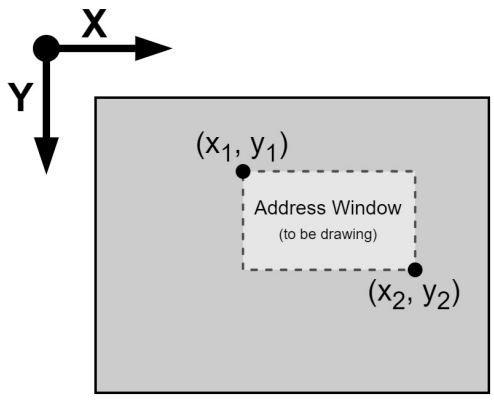
範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00UN01 slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.lcdInit(ECAT_LCD_ILI9341_1);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  slave.lcdFillScreen(0xFFE0);
}

void loop() {
  // ...
}
```

lcdSetAddrWindow()

說明

設定 EtherCAT 從站之 LCD 顯示區的繪圖視窗。後續的像素繪製操作將僅在此區域內執行。需與 `lcdPushColors()` 函式搭配使用。



語法

```
int lcdSetAddrWindow(int x1, int y1, int x2, int y2);
```

參數

- *[in] int x1*
  視窗左上角的 X 軸位置。
- *[in] int y1*
  視窗左上角的 Y 軸位置。
- *[in] int x2*
  視窗右下角的 X 軸位置，x2 必須大於或等於 x1 才是有效視窗。
- *[in] int y2*
  視窗右下角的 Y 軸位置，y2 必須大於或等於 y1 才是有效視窗。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

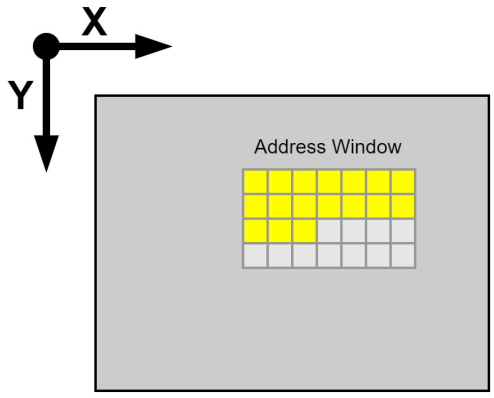
```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00UN01 slave;

uint16_t buffer[256];
```

```
void CyclicCallback() {
  slave.update();
}

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.lcdInit(ECAT_LCD_ILI9341_1);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  for (int i = 0; i < 128; i++)
    buffer[i] = 0xFFE0;
  for (int i = 0; i < 128; i++)
    buffer[i + 128] = 0xF800;

  slave.lcdSetAddrWindow(100, 100, 115, 115);
  slave.lcdPushColors(&buffer[0], 128, true);
  slave.lcdPushColors(&buffer[128], 128, false);
}

void loop() {
  // ...
}
```

lcdPushColors()

說明

將一組 16 位元色彩值陣列傳送至 EtherCAT 從站之 LCD 顯示器以進行像素繪圖。本函式需配合先前呼叫的 `lcdSetAddrWindow()` ，用以定義繪圖區域。



語法

```
int lcdPushColors(uint16_t *data, uint8_t len, bool first);
```

參數

- *[in] uint16_t *data*
  指向 16 位元色彩值陣列的指標。陣列中的每個元素代表一個像素的顏色，採用 RGB565 格式。
- *[in] uint8_t len*
  陣列中的色彩值長度，即預計繪製的像素數量。
  若像素數超過已設定的 LCD 視窗區域，超出部分將從視窗的左上角重新開始繪製。
- *[in] bool first*
  若設為 true，會將目前繪圖位置重置為 LCD 視窗的左上角；若設為 false，則會從當前繪圖位置繼續。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

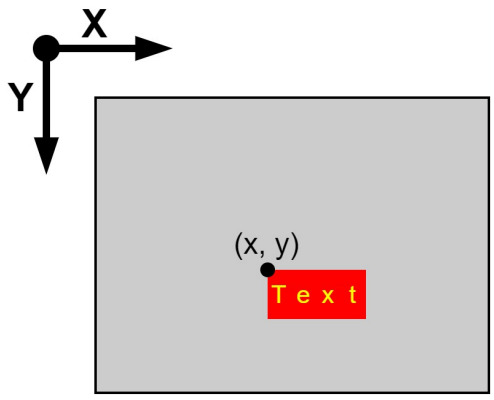
```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00UN01 slave;
```

```
uint16_t buffer[256];

void CyclicCallback() {
  slave.update();
}

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.lcdInit(ECAT_LCD_ILI9341_1);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  for (int i = 0; i < 128; i++)
    buffer[i] = 0xFFE0;
  for (int i = 0; i < 128; i++)
    buffer[i + 128] = 0xF800;

  slave.lcdSetAddrWindow(100, 100, 115, 115);
  slave.lcdPushColors(&buffer[0], 128, true);
  slave.lcdPushColors(&buffer[128], 128, false);
}

void loop() {
  // ...
}
```

lcdSetTextCursor()

說明

將 EtherCAT 從站 LCD 顯示器上的文字游標移動至指定的像素位置。後續的文字輸出將從此位置開始顯示。



語法

```
int lcdSetTextCursor(int16_t x, int16_t y);
```

參數

- *[in] int16_t x*
  文字游標的 X 軸位置（像素座標）。
- *[in] int16_t y*
  文字游標的 Y 軸位置（像素座標）。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

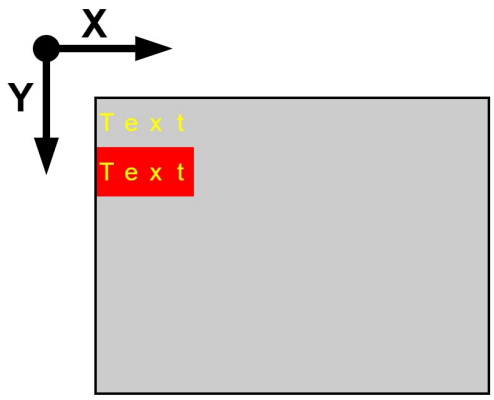
```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00UN01 slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
```

```
  master.begin();
  slave.attach(0, master);
  slave.lcdInit(ECAT_LCD_ILI9341_1);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  slave.lcdPrint("Hello World!\n");
  slave.lcdSetTextCursor(100, 100);
  slave.lcdPrint("Hello World!\n");
}

void loop() {
  // ...
}
```

lcdSetTextColor()

說明

設定 EtherCAT 從站 LCD 顯示器上要列印文字的字體顏色，亦可選擇性地設定文字背景色。



語法

```
int lcdSetTextColor(uint16_t color);
int lcdSetTextColor(uint16_t color, uint16_t background);
```

參數

- *[in] uint16_t color*
  要列印文字的字體顏色。為 16 位元無號整數，使用 RGB565 格式來編碼顏色資訊。
- *[in] uint16_t background*
  要列印文字的背景顏色。為 16 位元無號整數，使用 RGB565 格式來編碼顏色資訊。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00UN01 slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
```

```
  master.begin();
  slave.attach(0, master);
  slave.lcdInit(ECAT_LCD_ILI9341_1);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  slave.lcdSetTextColor(0xFFE0);
  slave.lcdPrint("Hello World!\n");
  slave.lcdSetTextColor(0xFFE0, 0xF800);
  slave.lcdPrint("Hello World!\n");
}

void loop() {
  // ...
}
```

lcdSetTextSize()

說明

設定在 EtherCAT 從站的 LCD 顯示器上列印文字的字型大小倍率。

預設的字型大小倍率為 1，對應的字型為 6 x 8 像素，表示文字寬度為 6 像素，高度為 8 像素。

您可以將倍率設為 2、3 或更高來放大字型。每增加一級倍率，文字會變得更大。

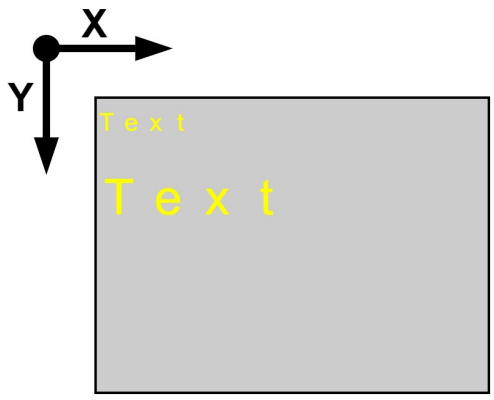
範例：

- s = 2: 12 x 16 pixels.
- s = 3: 18 x 24 pixels.
- s = 4: 24 x 32 pixels.
- 以此類推

除了等比例放大外，也可以分別設定寬度與高度的倍率，讓文字橫向或縱向拉伸，而不影響另一個方向。

範例：

- w = 1, h = 2: 6 x 16 pixels.
- w = 2, h = 1: 12 x 8 pixels.
- w = 2, h = 4: 12 x 32 pixels.



語法

```
int lcdSetTextSize(uint8_t s);
int lcdSetTextSize(uint8_t w, uint8_t h);
```

參數

- *[in] uint8_t s*
  要列印文字的字型整體倍率。
- *[in] uint8_t w*
  要列印文字的字型寬度倍率。
- *[in] uint8_t h*
  要列印文字的字型高度倍率。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

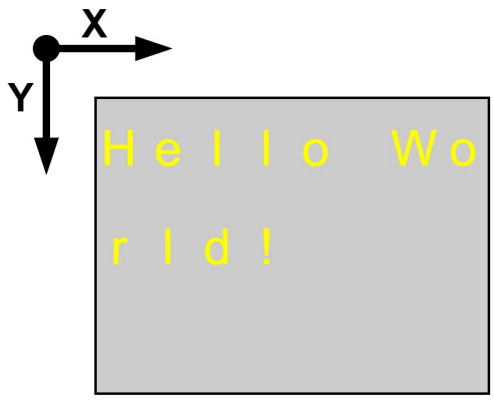
範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00UN01 slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.lcdInit(ECAT_LCD_ILI9341_1);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  slave.lcdSetTextSize(2);
  slave.lcdPrint("Hello World!\n");
  slave.lcdSetTextSize(2, 4);
  slave.lcdPrint("Hello World!\n");
}

void loop() {
  // ...
}
```

lcdSetTextWrap()

說明

設定當文字超出 LCD 顯示器寬度時，是否自動換行至下一行，或是直接裁切顯示。



語法

```
int lcdSetTextWrap(bool wrap);
```

參數

- *[in] bool wrap*
  用來控制是否啟用自動換行的布林值。
  - true：啟用自動換行。
  - false：停用自動換行（超出部分將被裁切）。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

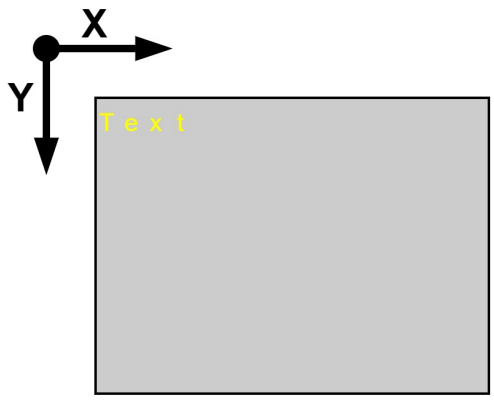
```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00UN01 slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  master.begin();
  slave.attach(0, master);
```

```
  slave.lcdInit(ECAT_LCD_ILI9341_1);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  slave.lcdSetTextWrap(true);
  slave.lcdPrint("Hello World!\n");
}

void loop() {
  // ...
}
```

lcdPrint()

說明

將指定的字串列印至 EtherCAT SubDevice 上的 LCD 顯示器。



語法

```
int lcdPrint(const char *fmt, ...);
```

參數

- *[in] const char *fmt*
  要顯示在 LCD 上的字串。這是一個指向以 null 結尾的字串指標，包含用於輸出的格式化說明。格式字串遵循 C 語言中的 printf 函式格式，允許插入變數與格式選項。
- *[in] ...*
  可變參數，用於依據 fmt 格式字串中的格式化指定插入對應的值。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00UN01 slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
```

```
  master.begin();
  slave.attach(0, master);
  slave.lcdInit(ECAT_LCD_ILI9341_1);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  slave.lcdPrint("Hello World!\n");
}

void loop() {
  // ...
}
```

觸控螢幕函式

適用於 EthercatDevice_DmpLCD_Generic 類別的觸控螢幕存取函式。

函式：

- touchCalibration()
- touchX()
- touchY()
- isTouched()

touchCalibration()

說明

EtherCAT 從站上 LCD 模組的觸控螢幕校正程序。此函式為非阻塞函式，內含校正程序的狀態機。必須持續呼叫此函式直到其回傳非零值，表示校正程序已完成。

語法

```
int touchCalibration(int flag = 0);
```

參數

- *[in] int flag*
  若此參數值為 -1，則會取消目前的觸控螢幕校正程序，此時函式將回傳 1 表示已取消。其他數值則不會影響校正程序的進行。

傳回值

回傳目前觸控螢幕校正程序的狀態：

- 0：校正進行中。
- 1：校正成功。
- -1：校正失敗。

若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00UN01 slave;

int rc;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  slave.lcdInit(ECAT_LCD_ILI9341_1);

  while ((rc = slave.touchCalibration()) == 0);
  if (rc > 0)
```

```
    Serial.println("Touch Screen calibration successful.");
  else
    Serial.println("Touch Screen calibration failed.");
}

void loop() {
  // ...
}
```

touchX()

說明

讀取 EtherCAT 從站上 LCD 模組觸控螢幕上觸控點的 X 軸座標。座標會依照 `lcdSetRotation()` 的設定進行旋轉。

語法

```
int touchX(size_t point = 0);
```

參數

- *`[in] size_t point`*
  觸控點的序號。若裝置支援多點觸控，則可透過此參數讀取指定觸控點的 X 軸位置。
  0：第一個觸控點。
  1：第二個觸控點。
  以此類推。

傳回值

回傳觸控點的 X 軸座標。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00UN01 slave;

bool Touched = false;
int TouchX;
int TouchY;

void CyclicCallback() {
  if (!Touched && slave.isTouched() > 0) {
    Touched = true;
    TouchX = slave.touchX();
    TouchY = slave.touchY();
  }
}

void setup() {
  Serial.begin(115200);
```

```
  master.begin();
  slave.attach(0, master);
  slave.lcdInit(ECAT_LCD_ILI9341_1);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  if (Touched) {
    Serial.print("Touched. X: ");
    Serial.print(TouchX);
    Serial.print(", Y: ");
    Serial.println(TouchY);
    delay(500);
    Touched = false;
  }
  // ...
}
```

touchY()

說明

讀取 EtherCAT 從站中 LCD 模組觸控螢幕上觸控點的 Y 軸位置。該座標會依據 `lcdSetRotation()` 的設定進行旋轉。

語法

```
int touchY(size_t point = 0);
```

參數

- *`[in] size_t point`*
  觸控點的序號。如果裝置支援多點觸控，此參數可用來讀取指定觸控點的 Y 軸位置。
  0：第 1 個觸控點
  1：第 2 個觸控點
  以此類推

傳回值

回傳觸控點的 Y 軸位置。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00UN01 slave;

bool Touched = false;
int TouchX;
int TouchY;

void CyclicCallback() {
  if (!Touched && slave.isTouched() > 0) {
    Touched = true;
    TouchX = slave.touchX();
    TouchY = slave.touchY();
  }
}

void setup() {
  Serial.begin(115200);
```

```
  master.begin();
  slave.attach(0, master);
  slave.lcdInit(ECAT_LCD_ILI9341_1);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  if (Touched) {
    Serial.print("Touched. X: ");
    Serial.print(TouchX);
    Serial.print(", Y: ");
    Serial.println(TouchY);
    delay(500);
    Touched = false;
  }
  // ...
}
```

isTouched()

說明

取得 EtherCAT 從站中觸控螢幕上的觸控點數量。對於僅支援單點觸控的裝置，此函式可用來判斷螢幕是否正在被觸碰。

語法

```
int isTouched();
```

參數

無。

傳回值

回傳觸控點的數量。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

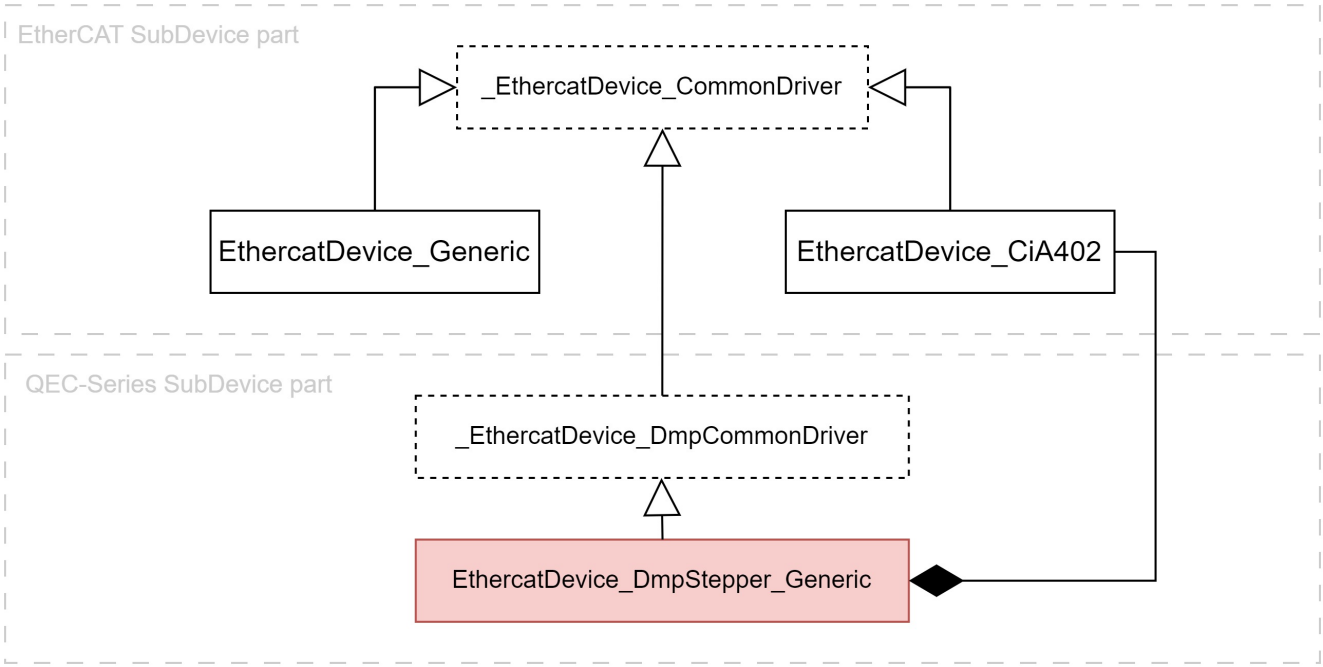
範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00UN01 slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  slave.lcdInit(ECAT_LCD_ILI9341_1);
  master.start();
}

void loop() {
  if (slave.isTouched() > 0) {
    Serial.println("Touched");
    delay(500);
  }
}
```

2.3.7 EthercatDevice_DmpStepper_Generic

EthercatDevice_DmpStepper_Generic 是 ICOP 為三軸步進馬達控制器 EtherCAT 從站裝置模組所開發的類別。該模組具備馬達驅動器、編碼器輸入及其他 CNC 相關功能。在馬達控制部分，除了支援 CiA 402 伺服模式，也支援三軸同步的 G-code 控制器模式。

EthercatDevice_DmpStepper_Generic 類別關係如下圖所示：



- EthercatDevice_DmpStepper_Generic 繼承自 _EthercatDevice_DmpCommonDriver.
- EthercatDevice_DmpStepper_Generic 組合了 EthercatDevice_CiA402.

基礎類別：

- _EthercatDevice_CommonDriver

衍生類別：

類別名稱	Vendor ID	Product Code
EthercatDevice_QECR11MP3S	0x00000bc3	0x0086d0d6
EthercatDevice_QECR00MP3S	0x00000bc3	0x0086d0d9

函式分類：

- 初始化
- 控制
- CiA 402
- Machine

- 編碼器
- 配置

函式：

函式名稱	說明	Callback 中呼叫
初始化相關函式		
attach()	初始化此 EtherCAT 從站裝置類別的物件	
detach()	解除初始化此 EtherCAT 從站裝置類別的物件	
cia402GetServo()	取得指定馬達的 CiA402 物件指標	O[1]
控制相關函式		
update()	更新狀態機與內部變數	O
attachInterrupt()	註冊事件 callback 函式	O
G-code 控制函式		
machineEnableSoftLimit()	啟用軟體極限功能	
machineDisableSoftLimit()	停用軟體極限功能	
machineIsEmergencyStopped()	檢查是否已緊急停止	O
machineSetEmergencyStop()	啟動緊急停止	O
machineClearEmergencyStop()	清除緊急停止狀態	O
machineIsLimitTouched()	檢查是否觸發極限開關	O
machineIsHomingAttained()	檢查是否完成歸零	O
machineIsServoOn()	檢查伺服是否開啟	O
machineIsMoving()	檢查機台是否正在移動	O
machineIsPositionErrorExceeded()	檢查是否超出位置誤差限制	O
machineServoOn()	開啟所有馬達伺服	O
machineServoOff()	關閉所有馬達伺服	O
machineSetHomingSpeed()	設定歸零速度	
machineStartHoming()	啟動歸零動作	O
machineGcode()	傳送 G-code 指令	O
machineActualPosition()	取得目前位置	O
編碼器存取函式		
encoderWrite()	設定編碼器計數器值	
encoderDirectionRead()	取得目前編碼器方向	O
encoderRead()	讀取編碼器計數器值	O
設定相關函式		
getDeviceMode()	取得裝置模式	
configDeviceMode()	設定裝置模式	
configCiA402MotorResolution()	設定 CiA 402 伺服模式下馬達的解析度	

configCiA402MotorPositionFeedbackSource()	設定馬達的位置回授來源	
configCiA402MotorPositionFeedbackScale()	設定馬達的位置回授比例	
configCiA402MotorPositionFeedbackOffset()	設定馬達的位置回授偏移值	
configCiA402MotorSelfStartingSpeed()	設定馬達的自起動速度	
configMachineAxisMapping()	設定機械軸與馬達的對應關係	
configMachineDefaultFeedrate()	設定預設進給速度	
configMachineDefaultHomingSpeed()	設定預設歸零速度	
configMachineHomingDirection()	設定歸零方向	
configMachineHomingPriority()	設定歸零優先順序	
configMachineMaxVelocity()	設定最大速度	
configMachineMaxAcceleration()	設定最大加速度	
configMachineSoftLimit()	設定軟體極限值	
configMachinePPU()	設定 PPU (Pulse Per Unit).	
configMachineAxisDirection()	設定馬達方向	
configMachinePositionFeedbackSource()	設定軸的位置回授來源	
configMachinePositionFeedbackScale()	設定軸的位置回授比例	
configMachinePositionFeedbackOffset()	設定軸的位置回授偏移值	
configMachineStepLossCompensationMode()	設定步進損失補償模式	
configMachineStepLossCompensationMaxError()	設定步進損失補償的最大誤差	
configMachineSelfStartingSpeed()	設定軸的自起動速度	
configMachineG54WorkOffset()	設定 G54 工作座標系偏移值	
configMachineG55WorkOffset()	設定 G55 工作座標系偏移值	
configMachineG56WorkOffset()	設定 G56 工作座標系偏移值	
configMachineG57WorkOffset()	設定 G57 工作座標系偏移值	
configMachineG58WorkOffset()	設定 G58 工作座標系偏移值	
configMachineG59WorkOffset()	設定 G59 工作座標系偏移值	
configEncoderMode()	設定編碼器模式	
configEncoderDigitalFilter()	設定編碼器數位濾波器	
configEncoderRange()	設定編碼器最大計數值	
configEncoderInputPolarity()	設定編碼器輸入腳位的極性	
configEncoderIndexReset()	啟用或停用索引訊號重置編碼器計數的功能	

- **Note 1**：此函數僅能在特定條件下在 callback 函數中呼叫。

初始化函式

EthercatDevice_DmpStepper_Generic 類別的初始化相關函式。

函式：

- attach()
- detach()
- cia402GetServo()

attach()

說明

此函式的行為與 `EthercatDevice Generic::attach()` 相同。

詳細的說明、語法、參數與傳回值，請參閱該章節。

範例

適用於 EthercatDevice_QECR11MP3S。

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S slave;

void setup() {
    master.begin();
    slave.attach(0, master);
    master.start();
}
void loop() {
  // ...

}
```

detach()

說明

此函式的行為與 `EthercatDevice Generic::detach()` 相同。

詳細的說明、語法、參數與傳回值，請參閱該章節。

範例

適用於 EthercatDevice_QECR11MP3S。

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S slave;

void setup() {
  master.begin();
  slave.attach(0, master);

  delay(3000);

  slave.detach();
  master.end();
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

cia402GetServo()

說明

取得 EtherCAT 從站中指定馬達的 EthercatDevice_CiA402 物件指標。

若 servo 參數為 NULL，首次呼叫此函式時將初始化一個內部的 CiA402 物件並回傳其指標；之後的呼叫將僅回傳此物件指標。

若 servo 參數不為 NULL 且內部物件已初始化，內部物件將被取消初始化，並使用輸入的 servo 參數來初始化新的 CiA402 物件。

語法

```
EthercatDevice_CiA402 *cia402GetServo(int motor, EthercatDevice_CiA402 *servo = NULL);
```

參數

- *[in] int motor*
  指定 EtherCAT 從站中的馬達編號。
  1：馬達 1。
  2：馬達 2。
  3：馬達 3。
- *[in] EthercatDevice_CiA402 *servo*
  使用者自行宣告的 EthercatDevice_CiA402 類別物件的指標。

傳回值

回傳 EthercatDevice_CiA402 類別的物件指標。

備註

此函式必須在成功執行 EthercatMaster::begin() 之後呼叫。 僅支援用於 EtherCAT 從站中的 CiA 402 模式。詳細資訊請參考 configDeviceMode()。

警告：禁止在 callback 函式中呼叫此函式。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S slave;
EthercatDevice_CiA402 motor1;
EthercatDevice_CiA402 motor2;

EthercatDevice_CiA402 *pMotor[3];

void setup() {
  master.begin();
  slave.attach(0, master);
  pMotor[0] = slave.cia402GetServo(1, &motor1);
```

```
  pMotor[1] = slave.cia402GetServo(2, &motor2);
  pMotor[2] = slave.cia402GetServo(3);
}

void loop() {
  // ...
}
```

控制函式

EthercatDevice_DmpStepper_Generic 類別的控制相關函式。

函式：

- update()
- attachInterrupt()

update()

說明

更新 EtherCAT 從站上各功能的狀態機與內部變數。

語法

```
int update();
```

參數

無。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  master.start();
}

void loop() {
  slave.update();
  // ...
}
```

attachInterrupt()

說明

註冊 EtherCAT 從站的事件回呼函式。

語法

```
int attachInterrupt(void (*callback)(int));
```

參數

- *[in] void (*callback)(int)*

  欲註冊的事件回呼函式，需帶有一個整數型參數，表示事件類型。支援的事件類型如下：

定義	代碼	說明
ECAT_EMERGENCY_STOPPED	1	發生緊急停止事件。
ECAT_MACHINE_X_AXIS_LIMIT_TOUCHED	2	觸發 X 軸限位開關。
ECAT_MACHINE_Y_AXIS_LIMIT_TOUCHED	3	觸發 Y 軸限位開關。
ECAT_MACHINE_Z_AXIS_LIMIT_TOUCHED	4	觸發 Z 軸限位開關。

  其餘事件類型可透過下列巨集進行判斷。由於這些事件的狀態儲存在 process data 中，且資料不具鎖存特性，建議於循環回呼中呼叫 `update()` 以處理事件並避免遺失。

定義	說明
IS_ECAT_ENCODER_1_INDEX_RESET(event)	編碼器 1 發生 Index Reset 事件。
IS_ECAT_ENCODER_2_INDEX_RESET(event)	編碼器 2 發生 Index Reset 事件。
IS_ECAT_ENCODER_3_INDEX_RESET(event)	編碼器 3 發生 Index Reset 事件。
IS_ECAT_ENCODER_X_INDEX_RESET(event)	編碼器 X 發生 Index Reset 事件。
IS_ECAT_ENCODER_Y_INDEX_RESET(event)	編碼器 Y 發生 Index Reset 事件。
IS_ECAT_ENCODER_Z_INDEX_RESET(event)	編碼器 Z 發生 Index Reset 事件。
IS_ECAT_ENCODER_1_OVERFLOW(event)	編碼器 1 發生 Overflow 事件。
IS_ECAT_ENCODER_2_OVERFLOW(event)	編碼器 2 發生 Overflow 事件。
IS_ECAT_ENCODER_3_OVERFLOW(event)	編碼器 3 發生 Overflow 事件。
IS_ECAT_ENCODER_X_OVERFLOW(event)	編碼器 X 發生 Overflow 事件。
IS_ECAT_ENCODER_Y_OVERFLOW(event)	編碼器 Y 發生 Overflow 事件。
IS_ECAT_ENCODER_Z_OVERFLOW(event)	編碼器 Z 發生 Overflow 事件。
IS_ECAT_ENCODER_1_UNDERFLOW(event)	編碼器 1 發生 Underflow 事件。
IS_ECAT_ENCODER_2_UNDERFLOW(event)	編碼器 2 發生 Underflow 事件。
IS_ECAT_ENCODER_3_UNDERFLOW(event)	編碼器 3 發生 Underflow 事件。
IS_ECAT_ENCODER_X_UNDERFLOW(event)	編碼器 X 發生 Underflow 事件。
IS_ECAT_ENCODER_Y_UNDERFLOW(event)	編碼器 Y 發生 Underflow 事件。
IS_ECAT_ENCODER_Z_UNDERFLOW(event)	編碼器 Z 發生 Underflow 事件。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

由於此回呼函式會在 `update()` 中被呼叫，若 `update()` 是在中斷回呼中執行，需遵守特定的使用限制。詳情請參見 Callback 函式。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S slave;

int emergency_stopped;
int x_limit_touched;
int y_limit_touched;
int z_limit_touched;
int encoder_index_reset[3];
int encoder_overflow[3];
int encoder_underflow[3];
int encoder_xyz_index_reset[3];
int encoder_xyz_overflow[3];
int encoder_xyz_underflow[3];

void Callback(int event) {
  switch (event) {
    case ECAT_EMERGENCY_STOPPED:
      emergency_stopped = 1;
      return;
    case ECAT_MACHINE_X_AXIS_LIMIT_TOUCHED:
      x_limit_touched = 1;
      return;
    case ECAT_MACHINE_Y_AXIS_LIMIT_TOUCHED:
      y_limit_touched = 1;
      return;
    case ECAT_MACHINE_Z_AXIS_LIMIT_TOUCHED:
      z_limit_touched = 1;
      return;
    default:
      break;
  }
```

```
  if (IS_ECAT_ENCODER_1_INDEX_RESET(event))
    encoder_index_reset[0] = 1;
  else if (IS_ECAT_ENCODER_2_INDEX_RESET(event))
    encoder_index_reset[1] = 1;
  else if (IS_ECAT_ENCODER_3_INDEX_RESET(event))
    encoder_index_reset[2] = 1;
  else if (IS_ECAT_ENCODER_1_OVERFLOW(event))
    encoder_overflow[0] = 1;
  else if (IS_ECAT_ENCODER_2_OVERFLOW(event))
    encoder_overflow[1] = 1;
  else if (IS_ECAT_ENCODER_3_OVERFLOW(event))
    encoder_overflow[2] = 1;
  else if (IS_ECAT_ENCODER_1_UNDERFLOW(event))
    encoder_underflow[0] = 1;
  else if (IS_ECAT_ENCODER_2_UNDERFLOW(event))
    encoder_underflow[1] = 1;
  else if (IS_ECAT_ENCODER_3_UNDERFLOW(event))
    encoder_underflow[2] = 1;

  if (IS_ECAT_ENCODER_X_INDEX_RESET(event))
    encoder_xyz_index_reset[0] = 1;
  else if (IS_ECAT_ENCODER_Y_INDEX_RESET(event))
    encoder_xyz_index_reset[1] = 1;
  else if (IS_ECAT_ENCODER_Z_INDEX_RESET(event))
    encoder_xyz_index_reset[2] = 1;
  else if (IS_ECAT_ENCODER_X_OVERFLOW(event))
    encoder_xyz_overflow[0] = 1;
  else if (IS_ECAT_ENCODER_Y_OVERFLOW(event))
    encoder_xyz_overflow[1] = 1;
  else if (IS_ECAT_ENCODER_Z_OVERFLOW(event))
    encoder_xyz_overflow[2] = 1;
  else if (IS_ECAT_ENCODER_X_UNDERFLOW(event))
    encoder_xyz_underflow[0] = 1;
  else if (IS_ECAT_ENCODER_Y_UNDERFLOW(event))
    encoder_xyz_underflow[1] = 1;
  else if (IS_ECAT_ENCODER_Z_UNDERFLOW(event))
    encoder_xyz_underflow[2] = 1;
}

void CyclicCallback() {
  slave.update();
```

```
}

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  slave.attachInterrupt(Callback);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  if (emergency_stopped) {
    emergency_stopped = 0;
    Serial.println("ECAT_EMERGENCY_STOPPED");
  }
  if (x_limit_touched) {
    x_limit_touched = 0;
    Serial.println("ECAT_MACHINE_X_AXIS_LIMIT_TOUCHED");
  }
  if (y_limit_touched) {
    y_limit_touched = 0;
    Serial.println("ECAT_MACHINE_Y_AXIS_LIMIT_TOUCHED");
  }
  if (z_limit_touched) {
    z_limit_touched = 0;
    Serial.println("ECAT_MACHINE_Z_AXIS_LIMIT_TOUCHED");
  }
  if (encoder_index_reset[0]) {
    encoder_index_reset[0] = 0;
    Serial.println("IS_ECAT_ENCODER_1_INDEX_RESET");
  }
  if (encoder_index_reset[1]) {
    encoder_index_reset[1] = 0;
    Serial.println("IS_ECAT_ENCODER_2_INDEX_RESET");
  }
  if (encoder_index_reset[2]) {
    encoder_index_reset[2] = 0;
    Serial.println("IS_ECAT_ENCODER_3_INDEX_RESET");
  }
  if (encoder_overflow[0]) {
```

```
    encoder_overflow[0] = 0;
    Serial.println("IS_ECAT_ENCODER_1_OVERFLOW");
  }
  if (encoder_overflow[1]) {
    encoder_overflow[1] = 0;
    Serial.println("IS_ECAT_ENCODER_2_OVERFLOW");
  }
  if (encoder_overflow[2]) {
    encoder_overflow[2] = 0;
    Serial.println("IS_ECAT_ENCODER_3_OVERFLOW");
  }
  if (encoder_underflow[0]) {
    encoder_underflow[0] = 0;
    Serial.println("IS_ECAT_ENCODER_1_UNDERFLOW");
  }
  if (encoder_underflow[1]) {
    encoder_underflow[1] = 0;
    Serial.println("IS_ECAT_ENCODER_2_UNDERFLOW");
  }
  if (encoder_underflow[2]) {
    encoder_underflow[2] = 0;
    Serial.println("IS_ECAT_ENCODER_3_UNDERFLOW");
  }
  if (encoder_xyz_index_reset[0]) {
    encoder_xyz_index_reset[0] = 0;
    Serial.println("IS_ECAT_ENCODER_X_INDEX_RESET");
  }
  if (encoder_xyz_index_reset[1]) {
    encoder_xyz_index_reset[1] = 0;
    Serial.println("IS_ECAT_ENCODER_Y_INDEX_RESET");
  }
  if (encoder_xyz_index_reset[2]) {
    encoder_xyz_index_reset[2] = 0;
    Serial.println("IS_ECAT_ENCODER_Z_INDEX_RESET");
  }
  if (encoder_xyz_overflow[0]) {
    encoder_xyz_overflow[0] = 0;
    Serial.println("IS_ECAT_ENCODER_X_OVERFLOW");
  }
  if (encoder_xyz_overflow[1]) {
    encoder_xyz_overflow[1] = 0;
    Serial.println("IS_ECAT_ENCODER_Y_OVERFLOW");
```

```
  }
  if (encoder_xyz_overflow[2]) {
    encoder_xyz_overflow[2] = 0;
    Serial.println("IS_ECAT_ENCODER_Z_OVERFLOW");
  }
  if (encoder_xyz_underflow[0]) {
    encoder_xyz_underflow[0] = 0;
    Serial.println("IS_ECAT_ENCODER_X_UNDERFLOW");
  }
  if (encoder_xyz_underflow[1]) {
    encoder_xyz_underflow[1] = 0;
    Serial.println("IS_ECAT_ENCODER_Y_UNDERFLOW");
  }
  if (encoder_xyz_underflow[2]) {
    encoder_xyz_underflow[2] = 0;
    Serial.println("IS_ECAT_ENCODER_Z_UNDERFLOW");
  }
  // ...
}
```

Machine 函式

適用於 EthercatDevice_DmpStepper_Generic 類別的機台控制函式。

函式：

- machineEnableSoftLimit()
- machineDisableSoftLimit()
- machineIsEmergencyStopped()
- machineSetEmergencyStop()
- machineClearEmergencyStop()
- machineIsLimitTouched()
- machineIsHomingAttained()
- machineIsServoOn()
- machineIsMoving()
- machineIsPositionErrorExceeded()
- machineServoOn()
- machineServoOff()
- machineSetHomingSpeed()
- machineStartHoming()
- machineGcode()
- machineActualPosition()

machineEnableSoftLimit()

說明

啟用 EtherCAT 從站機器的軟體限位功能。有關軟體限位的更多資訊，請參見 `configMachineSoftLimit()`。

語法

```
int machineEnableSoftLimit();
```

參數

無。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為阻塞式，不可於 callback 函式中呼叫。

此函式僅在 G-code 控制器模式下有效。詳細請參考 `configDeviceMode()`。

下列情況下此函式將無效：

- 機器處於 servo-off 狀態。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  slave.configMachineSoftLimit(ECAT_MACHINE_X_AXIS, -50, 50);
  slave.configMachineSoftLimit(ECAT_MACHINE_Y_AXIS, -50, 50);
  slave.configMachineSoftLimit(ECAT_MACHINE_Z_AXIS, -50, 50);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  slave.machineServoOn();
```

```
  while (slave.machineIsServoOn() == 0);

  slave.machineEnableSoftLimit();

  slave.machineGcode("G1 X-100 Y-100 Z-100");
  slave.machineGcode("G1 X100 Y100 Z100");
  delay(10);

  while (slave.machineIsMoving()) {
    Serial.print("Acutal Position => ");
    Serial.print("X: ");
    Serial.print(slave.machineActualPosition(ECAT_MACHINE_X_AXIS), 2);
    Serial.print(", Y: ");
    Serial.print(slave.machineActualPosition(ECAT_MACHINE_Y_AXIS), 2);
    Serial.print(", Z: ");
    Serial.println(slave.machineActualPosition(ECAT_MACHINE_Z_AXIS), 2);
    delay(10);
    // ...
  }

  slave.machineServoOff();
  while (slave.machineIsServoOn() == 1);
}

void loop() {
  // ...
}
```

machineDisableSoftLimit()

說明

停用 EtherCAT 從站機器的軟體限位功能。有關軟體限位的更多資訊，請參見 `configMachineSoftLimit()`。

語法

```
int machineDisableSoftLimit();
```

參數

無。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為阻塞式，不可於 callback 函式中呼叫。

此函式僅在 G-code 控制器模式下有效。詳細請參考 `configDeviceMode()`。

下列情況下此函式將無效：

- 機器處於 servo-off 狀態。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  slave.configMachineSoftLimit(ECAT_MACHINE_X_AXIS, -50, 50);
  slave.configMachineSoftLimit(ECAT_MACHINE_Y_AXIS, -50, 50);
  slave.configMachineSoftLimit(ECAT_MACHINE_Z_AXIS, -50, 50);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  slave.machineServoOn();
```

```
  while (slave.machineIsServoOn() == 0);

  slave.machineDisableSoftLimit();

  slave.machineGcode("G1 X-100 Y-100 Z-100");
  slave.machineGcode("G1 X100 Y100 Z100");
  delay(10);

  while (slave.machineIsMoving()) {
    Serial.print("Acutal Position => ");
    Serial.print("X: ");
    Serial.print(slave.machineActualPosition(ECAT_MACHINE_X_AXIS), 2);
    Serial.print(", Y: ");
    Serial.print(slave.machineActualPosition(ECAT_MACHINE_Y_AXIS), 2);
    Serial.print(", Z: ");
    Serial.println(slave.machineActualPosition(ECAT_MACHINE_Z_AXIS), 2);
    delay(10);
    // ...
  }

  slave.machineServoOff();
  while (slave.machineIsServoOn() == 1);
}

void loop() {
  // ...
}
```

machineIsEmergencyStopped()

說明

檢查 EtherCAT 從站上的機器是否處於緊急停止狀態。緊急停止可能由以下兩種主要情況觸發：

- 硬體緊急停止
  當實體的緊急停止開關被觸發時發生。這通常是一個設置於機台上的實體按鈕，用於在發生即時安全危險時立即停止機器運作。
- 使用者手動觸發的緊急停止
  當使用者呼叫 `machineSetEmergencyStop()` 函式時觸發。此函式通常透過軟體介面或控制面板進行手動操作以進入緊急停止狀態。

語法

```
int machineIsEmergencyStopped();
```

參數

無。

傳回值

回傳機器是否處於緊急停止狀態：

- 1：機器處於緊急停止狀態。
- 0：未處於緊急停止狀態。

若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為阻塞式，不可於 callback 函式中呼叫。

此函式僅在 G-code 控制器模式下有效。詳細請參考 `configDeviceMode()`。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S slave;

void setup() {
  Serial.begin(115200);

  master.begin();
  slave.attach(0, master);
  master.start();
}

void loop() {
  Serial.print("Emergency Stopped: ");
```

```
  Serial.println(slave.machineIsEmergencyStopped());
  delay(1000);
  //...
}
```

machineSetEmergencyStop()

說明

觸發 EtherCAT 從站上機器的緊急停止。此函式通常用於透過軟體介面或控制面板手動啟動緊急停止狀態。

語法

```
int machineSetEmergencyStop();
```

參數

無。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為阻塞式，不可於 callback 函式中呼叫。

此函式僅在 G-code 控制器模式下有效。詳細請參考 `configDeviceMode()`。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S slave;

void setup() {
  Serial.begin(115200);
  master.begin();
  slave.attach(0, master);
  master.start();

  slave.machineSetEmergencyStop();
  delay(10);
  Serial.print("Emergency Stopped: ");
  Serial.println(slave.machineIsEmergencyStopped());
}

void loop() {
  // ...
}
```

machineClearEmergencyStop()

說明

清除 EtherCAT 從站上機器的緊急停止狀態。

若要清除緊急停止狀態並恢復機器的正常運作，請確保實體的硬體緊急停止開關已解除或已復位為正常位置，再呼叫此函式。由於此函式為非阻塞式，並且需要持續呼叫 `update()` 函式以執行狀態機邏輯，因此可能需要一段時間才能完成動作。因此，像 `machineIsEmergencyStopped()` 這類相關狀態的回應也可能會有延遲。

語法

```
int machineClearEmergencyStop();
```

參數

無。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為阻塞式，不可於 callback 函式中呼叫。

此函式僅在 G-code 控制器模式下有效。詳細請參考 `configDeviceMode()`。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.machineClearEmergencyStop();
}

void loop() {
  slave.update();
  // ...
}
```

machineIsLimitTouched()

說明

檢查 EtherCAT 從站上指定機器軸的極限開關是否被觸發。

語法

```
int machineIsLimitTouched(int machine_axis);
```

參數

- *[in] int machine_axis*

  指定 EtherCAT 從站上的機器軸號：

定義	值	說明
ECAT_MACHINE_X_AXIS	0	X 軸，其對應由 configMachineAxisMapping() 決定。
ECAT_MACHINE_Y_AXIS	1	Y 軸，其對應由 configMachineAxisMapping() 決定。
ECAT_MACHINE_Z_AXIS	2	Z 軸，其對應由 configMachineAxisMapping() 決定。

傳回值

回傳該軸的極限開關是否被觸發。

- 1 表示該軸的極限開關被觸發。
- 0 表示未被觸發。

若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為阻塞式，不可於 callback 函式中呼叫。

此函式僅在 G-code 控制器模式下有效。詳細請參考 `configDeviceMode()`。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  Serial.begin(115200);
  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}
```

```
void loop() {
  Serial.print("Limit Switch => ");
  Serial.print("X: ");
  Serial.print(slave.machineIsLimitTouched(ECAT_MACHINE_X_AXIS));
  Serial.print(", Y: ");
  Serial.print(slave.machineIsLimitTouched(ECAT_MACHINE_Y_AXIS));
  Serial.print(", Z: ");
  Serial.println(slave.machineIsLimitTouched(ECAT_MACHINE_Z_AXIS));
  delay(1000);
  // ...
}
```

machineIsHomingAttained()

說明

檢查 EtherCAT 從站上的機器是否已完成原點復歸。

語法

```
int machineIsHomingAttained();
```

參數

無。

傳回值

回傳機器是否已完成原點復歸。

- 1 表示機器已完成原點復歸。
- 0 表示尚未完成。

若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為阻塞式，不可於 callback 函式中呼叫。

此函式僅在 G-code 控制器模式下有效。詳細請參考 `configDeviceMode()`。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  slave.machineServoOn();
  while (slave.machineIsServoOn() == 0);

  slave.machineStartHoming();
  delay(10);
  while (!slave.machineIsHomingAttained());
```

```
  // ..

  slave.machineServoOff();
  while (slave.machineIsServoOn() == 1);
}

void loop() {
  // ...
}
```

machineIsServoOn()

說明

檢查 EtherCAT 從站上的機器是否處於 Servo-on 狀態。

語法

```
int machineIsServoOn();
```

參數

無。

傳回值

回傳機器是否為 Servo-on 狀態。

- 1 表示機器為 Servo-on。
- 0 表示非 Servo-on。

若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為阻塞式，不可於 callback 函式中呼叫。

此函式僅在 G-code 控制器模式下有效。詳細請參考 `configDeviceMode()`。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  slave.machineServoOn();
  while (slave.machineIsServoOn() == 0);
  delay(1000);
  // ..
  slave.machineServoOff();
  while (slave.machineIsServoOn() == 1);
```

```
}

void loop() {
  // ...
}
```

machineIsMoving()

說明

檢查 EtherCAT 從站上的機器是否正在移動。

語法

```
int machineIsMoving();
```

參數

無。

傳回值

回傳機器是否正在移動。

- 1 表示機器正在移動。
- 0 表示未在移動。

若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為阻塞式，不可於 callback 函式中呼叫。

此函式僅在 G-code 控制器模式下有效。詳細請參考 `configDeviceMode()`。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  slave.machineServoOn();
  while (slave.machineIsServoOn() == 0);

  slave.machineGcode("G1 X0 Y0 Z0");
  slave.machineGcode("G1 X100 Y50 Z25");
  delay(10);
```

```
  while (slave.machineIsMoving());
  // ...

  slave.machineServoOff();
  while (slave.machineIsServoOn() == 1);
}

void loop() {
  // ...
}
```

machineIsPositionErrorExceeded()

說明

檢查指定機器軸的定位誤差是否超過步進遺失補償的最大容許誤差。

語法

```
int machineIsPositionErrorExceeded(int machine_axis);
```

參數

- *[in] int machine_axis*

  指定 EtherCAT 從站上的機器軸號：

定義	值	說明
ECAT_MACHINE_X_AXIS	0	X 軸，其對應由 configMachineAxisMapping() 決定。
ECAT_MACHINE_Y_AXIS	1	Y 軸，其對應由 configMachineAxisMapping() 決定。
ECAT_MACHINE_Z_AXIS	2	Z 軸，其對應由 configMachineAxisMapping() 決定。

傳回值

回傳是否超過最大定位誤差：

- 1 表示誤差超過。
- 0 表示未超過。

若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為阻塞式，不可於 callback 函式中呼叫。

此函式僅在 G-code 控制器模式下有效。詳細請參考 `configDeviceMode()`。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void Callback(int event) {
  if (event == ECAT_EMERGENCY_STOPPED) {
    if (device.machineIsPositionErrorExceeded(ECAT_MACHINE_X_AXIS)) {
      // ...
    }
    if (device.machineIsPositionErrorExceeded(ECAT_MACHINE_Y_AXIS)) {
      // ...
    }
    if (device.machineIsPositionErrorExceeded(ECAT_MACHINE_Z_AXIS)) {
      // ...
    }
```

```
  }
}

void CyclicCallback() {
  device.update();
}

void setup() {
  master.begin();
  device.attach(0, master);
  device.attachInterrupt(Callback);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

machineServoOn()

說明

啟動 EtherCAT 從站機器上的所有馬達。由於此函式為非阻塞式，需要持續呼叫 `update()` 函式以執行狀態機，因此相關狀態，如 `machineIsServoOn()`，可能會有延遲才更新。

語法

```
int machineServoOn();
```

參數

無。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為阻塞式，不可於 callback 函式中呼叫。

此函式僅在 G-code 控制器模式下有效。詳細請參考 `configDeviceMode()`。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  slave.machineServoOn();
  while (slave.machineIsServoOn() == 0);
  delay(1000);
  // ..
  slave.machineServoOff();
  while (slave.machineIsServoOn() == 1);
}

void loop() {
```

```
  // ...
}
```

machineServoOff()

說明

關閉 EtherCAT 從站機器上的所有馬達。

由於此函式為非阻塞式，需要持續呼叫 `update()` 函式以執行狀態機，因此相關狀態，如 `machineIsServoOn()`，可能會有延遲才更新。

語法

```
int machineServoOff();
```

參數

無。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為阻塞式，不可於 callback 函式中呼叫。

此函式僅在 G-code 控制器模式下有效。詳細請參考 `configDeviceMode()`。

此函式在以下情況下無法運作：

- 機器正在執行原點復歸
- 機器正在運動中

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  slave.machineServoOn();
  while (slave.machineIsServoOn() == 0);
  delay(1000);
  // ..
```

```
  slave.machineServoOff();
  while (slave.machineIsServoOn() == 1);
}

void loop() {
  // ...
}
```

machineSetHomingSpeed()

說明

設定 EtherCAT 從站機器上指定軸的歸零速度。

語法

```
int machineSetHomingSpeed(int machine_axis, double mm_per_min);
```

參數

- *[in] int machine_axis*

  指定 EtherCAT 從站上的機器軸號：

定義	值	說明
ECAT_MACHINE_X_AXIS	0	X 軸，其對應由 configMachineAxisMapping() 決定。
ECAT_MACHINE_Y_AXIS	1	Y 軸，其對應由 configMachineAxisMapping() 決定。
ECAT_MACHINE_Z_AXIS	2	Z 軸，其對應由 configMachineAxisMapping() 決定。

- *[in] double mm_per_min*

  歸零運動的目標速度，單位：毫米/分鐘。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

本函式必須在成功執行 EthercatMaster::start() 並在 EthercatMaster::stop() 之前呼叫。此函式為阻塞式，不可於 callback 函式中呼叫。

此函式僅在 G-code 控制器模式下有效。詳細請參考 configDeviceMode()。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.machineSetHomingSpeed(ECAT_MACHINE_X_AXIS, 3000);
  slave.machineSetHomingSpeed(ECAT_MACHINE_Y_AXIS, 3000);
  slave.machineSetHomingSpeed(ECAT_MACHINE_Z_AXIS, 3000);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
```

```
  slave.machineServoOn();
  while (slave.machineIsServoOn() == 0);

  slave.machineStartHoming();
  delay(10);
  while (!slave.machineIsHomingAttained());
  // ..

  slave.machineServoOff();
  while (slave.machineIsServoOn() == 1);
}

void loop() {
  // ...
}
```

machineStartHoming()

說明

啟動 EtherCAT 從站機器的歸零程序。

由於此函式為非阻塞式，需持續呼叫 `update()` 函式以執行狀態機，因此相關狀態，如 `machineIsHomingAttained()`，可能需要一段時間才會反映變化。

語法

```
int machineStartHoming();
```

參數

無。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為阻塞式，不可於 callback 函式中呼叫。

此函式僅在 G-code 控制器模式下有效。詳細請參考 `configDeviceMode()`。

下列情況下此函式無法運作：

- 機器處於緊急停止狀態。
- 機器伺服尚未開啟（servo-off）。
- 機器正在執行歸零（homing）。
- 機器正在移動中（moving）。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  slave.machineServoOn();
  while (slave.machineIsServoOn() == 0);
```

```
  slave.machineStartHoming();
  delay(10);
  while (!slave.machineIsHomingAttained());
  // ..

  slave.machineServoOff();
  while (slave.machineIsServoOn() == 1);

}

void loop() {
  // ...
}
```

machineGcode()

說明

將 G-code 指令傳送至 EtherCAT 從站。

由於函數是非阻塞函數，並且需要不斷呼叫 `update()` 來執行狀態機，所以可能需要一些時間才能完成，因此相關狀態可能需要一些時間才能回應，例如 `machineIsMoving()`。

語法

```
int machineGcode(const char *fmt, ...);
```

參數

- *`[in] const char *fmt`*
  傳送至 EtherCAT 從站的 G-code 指令字串。這是一個指向以 null 結尾的字串指標，包含輸出的格式說明。格式字串遵循 C 語言的 printf 函式格式，允許插入變數及格式化選項。
- *`[in] ...`*
  根據格式字串中的格式指定符插入的變數數量不定引數。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為阻塞式，不可於 callback 函式中呼叫。

此函式僅在 G-code 控制器模式下有效。詳細請參考 `configDeviceMode()`。

以下情況下此函式無法運作：

- 機器處於緊急停止狀態。
- 機器為伺服關閉狀態（servo-off）。
- 機器正在歸零中（homing）。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
```

```
  slave.machineServoOn();
  while (slave.machineIsServoOn() == 0);

  while (1) {
    slave.machineGcode("G1 X100 Y50 Z25");
    delay(1000);
    slave.machineGcode("G1 X0 Y0 Z0");
    delay(1000);
    // ...
  }

  slave.machineServoOff();
  while (slave.machineIsServoOn() == 1);
}

void loop() {
  // ...
}
```

machineActualPosition()

說明

取得 EtherCAT 從站上指定機械軸的目前位置。

語法

```
double machineActualPosition(int machine_axis);
```

參數

- *[in] int machine_axis*

  指定 EtherCAT 從站上的機器軸號：

定義	值	說明
ECAT_MACHINE_X_AXIS	0	X 軸，其對應由 configMachineAxisMapping() 決定。
ECAT_MACHINE_Y_AXIS	1	Y 軸，其對應由 configMachineAxisMapping() 決定。
ECAT_MACHINE_Z_AXIS	2	Z 軸，其對應由 configMachineAxisMapping() 決定。

傳回值

傳回指定機械軸目前的位置，單位：毫米。

備註

本函式必須在成功執行 EthercatMaster::start() 並在 EthercatMaster::stop() 之前呼叫。此函式為阻塞式，不可於 callback 函式中呼叫。

此函式僅在 G-code 控制器模式下有效。詳細請參考 configDeviceMode()。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S slave;

void CyclicCallback() {
  slave.update();
}

void setup() {
  master.begin();
  slave.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  slave.machineServoOn();
  while (slave.machineIsServoOn() == 0);
}
```

```
void loop() {
  if (slave.machineIsMoving() == 0) {
    slave.machineGcode("G1 X0 Y0 Z0");
    slave.machineGcode("G1 X100 Y50 Z25");
    delay(10);
  }
  printf("Acutal Position => ");
  printf("X:%.2f, ", slave.machineActualPosition(ECAT_MACHINE_X_AXIS));
  printf("Y:%.2f, ", slave.machineActualPosition(ECAT_MACHINE_Y_AXIS));
  printf("Z:%.2f\n", slave.machineActualPosition(ECAT_MACHINE_Z_AXIS));
  delay(10);
  // ...
}
```

編碼器函式

適用於 EthercatDevice_DmpStepper_Generic 類別的編碼器函式。

函式：

- encoderWrite()
- encoderDirectionRead()
- encoderRead()

encoderWrite()

說明

寫入 EtherCAT 從站上指定編碼器的計數值。

語法

```
int encoderWrite(int encoder, int32_t value);
```

參數

- *[in] int encoder*

  指定的編碼器編號：

定義	值	說明
ECAT_ENCODER_1	0x01	EtherCAT 從站上的編碼器 1。
ECAT_ENCODER_2	0x02	EtherCAT 從站上的編碼器 2。
ECAT_ENCODER_3	0x03	EtherCAT 從站上的編碼器 3。
ECAT_ENCODER_X	0x11	映射到 X 軸的編碼器。
ECAT_ENCODER_Y	0x12	映射到 Y 軸的編碼器。
ECAT_ENCODER_Z	0x13	映射到 Z 軸的編碼器。

  編碼器到機械軸的對應由 `configMachineAxisMapping()` 和 `configMachinePositionFeedbackSource()` 決定。

- *[in] int32_t value*

  要寫入的計數值。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為非阻塞式，可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.encoderWrite(ECAT_ENCODER_1, 100);
  slave.encoderWrite(ECAT_ENCODER_2, 100);
  slave.encoderWrite(ECAT_ENCODER_3, 100);
```

```
  master.start();
}

void loop() {
  // ...
}
```

encoderDirectionRead()

說明

讀取 EtherCAT 從站上指定編碼器的當前旋轉方向。

語法

```
int encoderDirectionRead(int encoder);
```

參數

- *[in] int encoder*

  指定的編碼器編號：

定義	值	說明
ECAT_ENCODER_1	0x01	EtherCAT 從站上的編碼器 1。
ECAT_ENCODER_2	0x02	EtherCAT 從站上的編碼器 2。
ECAT_ENCODER_3	0x03	EtherCAT 從站上的編碼器 3。
ECAT_ENCODER_X	0x11	映射到 X 軸的編碼器。
ECAT_ENCODER_Y	0x12	映射到 Y 軸的編碼器。
ECAT_ENCODER_Z	0x13	映射到 Z 軸的編碼器。

  編碼器到機械軸的對應由 `configMachineAxisMapping()` 和 `configMachinePositionFeedbackSource()` 決定。

傳回值

傳回指定編碼器的當前方向：

- 傳回值為 0 表示正轉（Forward rotation）
- 傳回值為 1 表示反轉（Reverse rotation）

若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為阻塞式，不可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S slave;

void setup() {
  Serial.begin(115200);
  master.begin();
  slave.attach(0, master);
  master.start();
}
```

```
void loop() {
  Serial.print("Encoder 1 Direction: ");
  Serial.println(slave.encoderDirectionRead(ECAT_ENCODER_1));
  Serial.print("Encoder 2 Direction: ");
  Serial.println(slave.encoderDirectionRead(ECAT_ENCODER_2));
  Serial.print("Encoder 3 Direction: ");
  Serial.println(slave.encoderDirectionRead(ECAT_ENCODER_3));
  Serial.print("Encoder X Direction: ");
  Serial.println(slave.encoderDirectionRead(ECAT_ENCODER_X));
  Serial.print("Encoder Y Direction: ");
  Serial.println(slave.encoderDirectionRead(ECAT_ENCODER_Y));
  Serial.print("Encoder Z Direction: ");
  Serial.println(slave.encoderDirectionRead(ECAT_ENCODER_Z));
  delay(1000);
  // ...
}
```

encoderRead()

說明

讀取 EtherCAT 從站上指定編碼器的計數值。

語法

```
int32_t encoderRead(int encoder);
```

參數

- *[in] int encoder*

  指定的編碼器編號：

定義	值	說明
ECAT_ENCODER_1	0x01	EtherCAT 從站上的編碼器 1。
ECAT_ENCODER_2	0x02	EtherCAT 從站上的編碼器 2。
ECAT_ENCODER_3	0x03	EtherCAT 從站上的編碼器 3。
ECAT_ENCODER_X	0x11	映射到 X 軸的編碼器。
ECAT_ENCODER_Y	0x12	映射到 Y 軸的編碼器。
ECAT_ENCODER_Z	0x13	映射到 Z 軸的編碼器。

  編碼器到機械軸的對應由 `configMachineAxisMapping()` 和 `configMachinePositionFeedbackSource()` 決定。

傳回值

傳回指定編碼器的計數值，為 32 位元帶符號整數。

備註

本函式必須在成功執行 `EthercatMaster::start()` 並在 `EthercatMaster::stop()` 之前呼叫。此函式為阻塞式，不可於 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S slave;

void setup() {
  Serial.begin(115200);
  master.begin();
  slave.attach(0, master);
  master.start();
}

void loop() {
  Serial.print("Encoder 1: ");
  Serial.println(slave.encoderRead(ECAT_ENCODER_1));
```

```
  Serial.print("Encoder 2: ");
  Serial.println(slave.encoderRead(ECAT_ENCODER_2));
  Serial.print("Encoder 3: ");
  Serial.println(slave.encoderRead(ECAT_ENCODER_3));
  Serial.print("Encoder X: ");
  Serial.println(slave.encoderRead(ECAT_ENCODER_X));
  Serial.print("Encoder Y: ");
  Serial.println(slave.encoderRead(ECAT_ENCODER_Y));
  Serial.print("Encoder Z: ");
  Serial.println(slave.encoderRead(ECAT_ENCODER_Z));
  delay(1000);
  // ...
}
```

設定函式

適用於 EthercatDevice_DmpStepper_Generic 類別的組態函式。

函式：

- getDeviceMode()
- configDeviceMode()
- configCiA402MotorResolution()
- configCiA402MotorPositionFeedbackSource()
- configCiA402MotorPositionFeedbackScale()
- configCiA402MotorPositionFeedbackOffset()
- configCiA402MotorSelfStartingSpeed()
- configMachineAxisMapping()
- configMachineDefaultFeedrate()
- configMachineDefaultHomingSpeed()
- configMachineHomingDirection()
- configMachineHomingPriority()
- configMachineMaxVelocity()
- configMachineMaxAcceleration()
- configMachineSoftLimit()
- configMachinePPU()
- configMachineAxisDirection()
- configMachinePositionFeedbackSource()
- configMachinePositionFeedbackScale()
- configMachinePositionFeedbackOffset()
- configMachineStepLossCompensationMode()
- configMachineStepLossCompensationMaxError()
- configMachineSelfStartingSpeed()
- configMachineG54WorkOffset()
- configMachineG55WorkOffset()
- configMachineG56WorkOffset()
- configMachineG57WorkOffset()
- configMachineG58WorkOffset()
- configMachineG59WorkOffset()
- configEncoderMode()
- configEncoderDigitalFilter()
- configEncoderRange()
- configEncoderInputPolarity()
- configEncoderIndexReset()

getDeviceMode()

說明

取得 EtherCAT 從站的裝置模式。

語法

```
int getDeviceMode();
```

參數

無。

傳回值

傳回裝置模式。若回傳值小於 0，表示發生錯誤，並回傳對應的錯誤代碼。關於裝置模式的詳細說明，請參考 `configDeviceMode()`。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S slave;

void setup() {
  Serial.begin(115200);
  master.begin();
  slave.attach(0, master);

  Serial.print("Device Mode: ");
  Serial.println(slave.getDeviceMode());
}

void loop() {
  // ...
}
```

configDeviceMode()

說明

設定 EtherCAT 從站的裝置模式。該參數會寫入 EtherCAT 從站的 EEPROM，並在開機時載入，因此使用者不需在每次執行程式前重新設定。呼叫此函式後，EtherCAT 從站的裝置模式不會立即變更，需重新上電後變更才會生效。

此 EtherCAT 從站支援以下兩種裝置模式：

- ***CiA 402 Servo***
  此模式下，從站配備三軸 CiA 402 步進馬達，使用者可獨立控制每一軸馬達。
- ***G-code Controller***
  此模式下，從站作為 G-code 控制器，負責解析並執行 G-code 指令。

語法

```
int configDeviceMode(uint8_t mode);
```

參數

- *[in] uint8_t mode*

  此 EtherCAT 從站支援的裝置模式如下：

定義	值	說明
ECAT_CIA402_SERVO_MODE	0	*CiA 402 Servo* mode.
ECAT_GCODE_CONTROLLER_MODE	1	*G-code Controller* mode.

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S slave;

void setup() {
  master.begin();
  slave.attach(0, master);

  slave.configDeviceMode(ECAT_GCODE_CONTROLLER_MODE);
}
```

```
void loop() {
  // ...
}
```

configCiA402MotorResolution()

說明

在 CiA 402 伺服模式下，配置 EtherCAT 從站上指定馬達的馬達解析度。此參數寫入 EtherCAT 從站的 EEPROM 中，並在啟動時加載，因此使用者無需在每次運行程序前配置此參數。

語法

```
int configCiA402MotorResolution(int motor, uint32_t steps_per_rev);
```

參數

- *[in] motor*
  指定 EtherCAT 從站上的馬達編號：

值	說明
1	EtherCAT 從站上的馬達 1
2	EtherCAT 從站上的馬達 2
3	EtherCAT 從站上的馬達 3

- *[in] steps_per_rev*
  要配置的馬達分辨率，以每轉的步數來指定。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

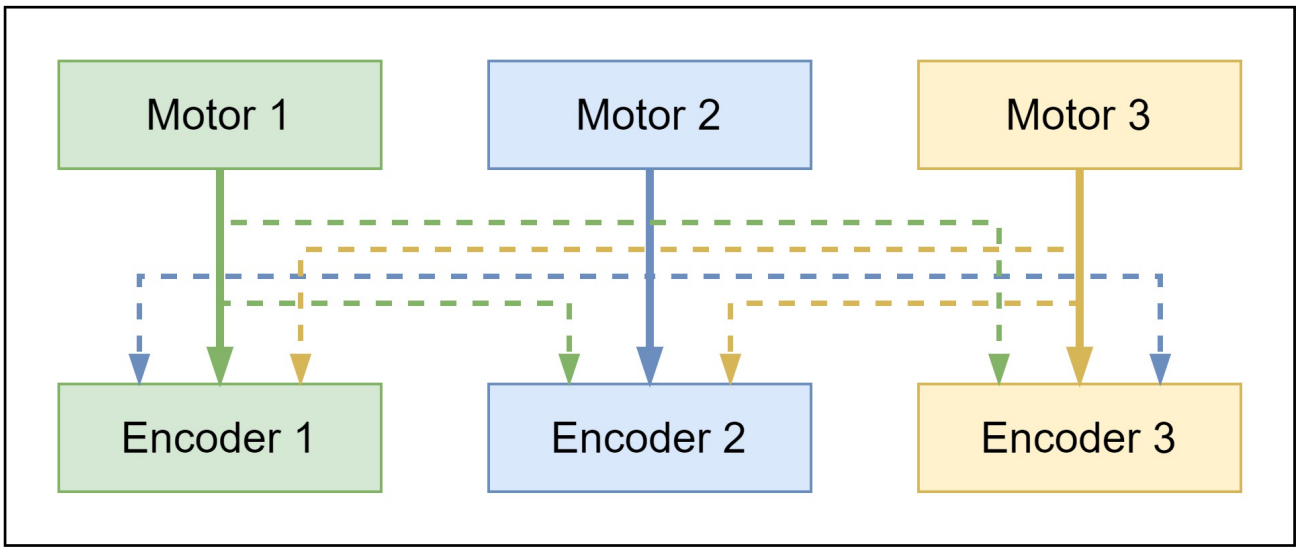
範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.configCiA402MotorResolution(1, 3200);
  slave.configCiA402MotorResolution(2, 3200);
  slave.configCiA402MotorResolution(3, 3200);
  // ...
}
void loop() {
  // ...
}
```

configCiA402MotorPositionFeedbackSource()

說明

設定 EtherCAT 從站中指定馬達的位置回授來源。



語法

```
int configCiA402MotorPositionFeedbackSource(int motor, int encoder);
```

參數

- *[in] motor*

  指定 EtherCAT 從站上的馬達編號：

值	說明
1	EtherCAT 從站上的馬達 1
2	EtherCAT 從站上的馬達 2
3	EtherCAT 從站上的馬達 3

- *[in] encoder*

  要設定的位置回授來源：

定義	值	說明
ECAT_ENCODER_1	0x01	*EtherCAT 從站上的編碼器 1*
ECAT_ENCODER_2	0x02	*EtherCAT 從站上的編碼器 2*
ECAT_ENCODER_3	0x03	*EtherCAT 從站上的編碼器 3*
Other Value	...	停用（Disabled）

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configCiA402MotorPositionFeedbackSource(1, ECAT_ENCODER_1);
  device.configCiA402MotorPositionFeedbackSource(2, ECAT_ENCODER_2);
  device.configCiA402MotorPositionFeedbackSource(3, ECAT_ENCODER_3);
  // ...
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

configCiA402MotorPositionFeedbackScale()

說明

設定 EtherCAT 從站中指定馬達的位置回授比例。

Position Feedback = (Encoder Raw Data × Scale) + Offset

位置回授 =（編碼器原始資料 × 比例）+ 偏移量（Offset）

比例（Scale）代表一個轉換因子，用於將數位編碼器的讀值轉換為實際的物理單位。本質上，它建立了編碼器計數與實際位移或旋轉之間的對應關係。此參數將寫入 EtherCAT 從站的 EEPROM，並於啟動時載入，因此使用者無需每次執行程式前都重新設定。

語法

```
int configCiA402MotorPositionFeedbackScale(int motor, double scale);
```

參數

- *[in] motor*

  指定 EtherCAT 從站上的馬達編號：

值	說明
1	EtherCAT 從站上的馬達 1
2	EtherCAT 從站上的馬達 2
3	EtherCAT 從站上的馬達 3

- *[in] scale*

  要設定的位置回授比例值。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configCiA402MotorPositionFeedbackScale(1, 2.0);
  device.configCiA402MotorPositionFeedbackScale(2, 2.0);
```

```
  device.configCiA402MotorPositionFeedbackScale(3, 2.0);
  // ...
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

configCiA402MotorPositionFeedbackOffset()

說明

設定 EtherCAT 從站中指定馬達的位置回授偏移量（Offset）。

Position Feedback = (Encoder Raw Data × Scale) + Offset

位置回授 =（編碼器原始資料 × 比例）+ 偏移量

偏移量（Offset）代表系統相對於參考點的初始位移或起始位置。本質上，它是一個校正因子，用來修正編碼器歸零讀數與實際物理歸零位置之間的差異。

此參數將寫入 EtherCAT 從站的 EEPROM，並於啟動時載入，因此使用者無需每次執行程式前都重新設定。

語法

```
int configCiA402MotorPositionFeedbackOffset(int motor, double offset);
```

參數

- *[in] motor*

  指定 EtherCAT 從站上的馬達編號：

值	說明
1	EtherCAT 從站上的馬達 1
2	EtherCAT 從站上的馬達 2
3	EtherCAT 從站上的馬達 3

- *[in] offset*

  要設定的位置回授偏移量。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configCiA402MotorPositionFeedbackOffset(1, 10.0);
```

```
  device.configCiA402MotorPositionFeedbackOffset(2, 10.0);
  device.configCiA402MotorPositionFeedbackOffset(3, 10.0);
  // ...
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

configCiA402MotorSelfStartingSpeed()

說明

設定 EtherCAT 從站中指定馬達的自啟動速度（Self-Starting Speed）。

自啟動速度是指步進馬達在未使用加速度斜率的情況下，能夠從靜止瞬間加速至穩定運轉所能承受的最高脈衝頻率。若超過此速度，馬達可能會失步或無法啟動。

語法

```
int configCiA402MotorSelfStartingSpeed(int motor, double rev_per_min);
```

參數

- *[in] motor*

  指定 EtherCAT 從站上的馬達編號：

值	說明
1	EtherCAT 從站上的馬達 1
2	EtherCAT 從站上的馬達 2
3	EtherCAT 從站上的馬達 3

- *[in] rev_per_min*

  要設定的自啟動速度，單位為 RPM (Revolutions Per Minute)。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

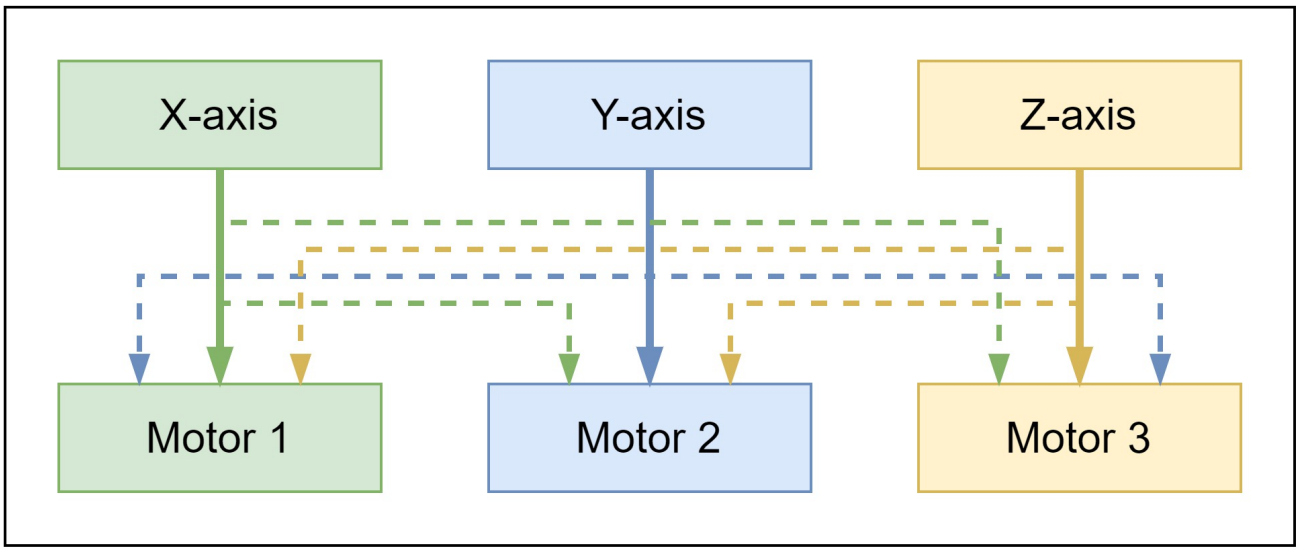
```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configCiA402MotorSelfStartingSpeed(1, 60.0);
  device.configCiA402MotorSelfStartingSpeed(2, 60.0);
  device.configCiA402MotorSelfStartingSpeed(3, 60.0);
  // ...
}

void loop() {
```

```
  // put your main code here, to run repeatedly:

}
```

configMachineAxisMapping()

說明

設定 EtherCAT 從站在 G-code 控制器模式下，機械軸與馬達之間的對應關係。



此參數會寫入 EtherCAT 從站的 EEPROM，並於開機時載入，因此使用者不需每次執行程式前皆進行設定。呼叫此函式後，對應關係不會立即生效。需重新啟動 EtherCAT 從站後設定才會生效。

語法

```
int configMachineAxisMapping(uint8_t mapping);
```

參數

- *`[in] mapping`*

  要設定的機械軸對應方式。EtherCAT 從站提供以下對應選項：

代碼	X-axis	Y-axis	Z-axis
0	Motor 1	Motor 2	Motor 3
1	Motor 1	Motor 3	Motor 2
2	Motor 2	Motor 1	Motor 3
3	Motor 2	Motor 3	Motor 1
4	Motor 3	Motor 1	Motor 2
5	Motor 3	Motor 2	Motor 1

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"
```

```
EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configMachineAxisMapping(0);
}

void loop() {
  // ...
}
```

configMachineDefaultFeedrate()

說明

設定 EtherCAT 從站在 G-code 控制器模式下的預設進給速率。若使用者透過 `machineGcode()` 傳送的 G-code 移動指令中尚未指定進給速率，則會使用此預設進給速率。此參數會寫入 EtherCAT 從站的 EEPROM，並於開機時載入，因此使用者不需每次執行程式前皆進行設定。

語法

```
int configMachineDefaultFeedrate(double mm_per_min);
```

參數

- *[in] double mm_per_min*
  要設定的預設進給速率，單位為毫米/分鐘（mm/min）。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configMachineDefaultFeedrate(3000);
  // ...
}

void loop() {
  // ...
}
```

configMachineDefaultHomingSpeed()

說明

設定 EtherCAT 從站於 G-code 控制器模式下指定機械軸的預設復歸（Homing）速度。若使用者尚未透過 `machineSetHomingSpeed()` 為每個軸單獨設定復歸速度，則會使用此預設值。此參數會寫入 EtherCAT 從站的 EEPROM，並於啟動時載入，因此使用者無需在每次執行程式前進行設定。

語法

```
int configMachineDefaultHomingSpeed(int machine_axis, double mm_per_min);
```

參數

- *`[in] int machine_axis`*
  指定 EtherCAT 從站上的機器軸號：

定義	值	說明
ECAT_MACHINE_X_AXIS	0	X 軸，其對應由 `configMachineAxisMapping()` 決定。
ECAT_MACHINE_Y_AXIS	1	Y 軸，其對應由 `configMachineAxisMapping()` 決定。
ECAT_MACHINE_Z_AXIS	2	Z 軸，其對應由 `configMachineAxisMapping()` 決定。

- *`[in] double mm_per_min`*
  欲設定的預設復歸速度，單位為毫米/分鐘（mm/min）。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S slave;

void setup() {
  master.begin();
  slave.attach(0, master);

  slave.configMachineDefaultHomingSpeed(ECAT_MACHINE_X_AXIS, 1000);
  slave.configMachineDefaultHomingSpeed(ECAT_MACHINE_Y_AXIS, 1000);
  slave.configMachineDefaultHomingSpeed(ECAT_MACHINE_Z_AXIS, 1000);
  // ...
}
```

```
void loop() {
  // ...
}
```

configMachineHomingDirection()

說明

設定 EtherCAT 從站於 G-code 控制器模式下，指定機械軸的復歸（Homing）方向。此參數會寫入 EtherCAT 從站的 EEPROM，並於啟動時載入，因此使用者無需在每次執行程式前進行設定。

語法

```
int configMachineHomingDirection(int machine_axis, bool positive);
```

參數

- *[in] int machine_axis*

  指定 EtherCAT 從站上的機器軸號：

定義	值	說明
ECAT_MACHINE_X_AXIS	0	X 軸，其對應由 configMachineAxisMapping() 決定。
ECAT_MACHINE_Y_AXIS	1	Y 軸，其對應由 configMachineAxisMapping() 決定。
ECAT_MACHINE_Z_AXIS	2	Z 軸，其對應由 configMachineAxisMapping() 決定。

- *[in] bool positive*

  布林值，用來設定復歸方向：

  - true：往正方向尋找原點開關。
  - false：往負方向尋找原點開關。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 EthercatMaster::begin() 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S slave;

void setup() {
  master.begin();
  slave.attach(0, master);

  slave.configMachineHomingDirection(ECAT_MACHINE_X_AXIS, false);
  slave.configMachineHomingDirection(ECAT_MACHINE_Y_AXIS, false);
  slave.configMachineHomingDirection(ECAT_MACHINE_Z_AXIS, false);
  // ...
}
```

```
void loop() {
  // ...
}
```

configMachineHomingPriority()

說明

設定 EtherCAT 從站於 G-code 控制器模式下，指定機械軸的復歸（Homing）優先順序。此參數會寫入 EtherCAT 從站的 EEPROM，並於啟動時載入，因此使用者無需在每次執行程式前進行設定。

語法

```
int configMachineHomingPriority(int machine_axis, uint8_t priority);
```

參數

- *[in] int machine_axis*

  指定 EtherCAT 從站上的機器軸號：

定義	值	說明
ECAT_MACHINE_X_AXIS	0	X 軸，其對應由 configMachineAxisMapping() 決定。
ECAT_MACHINE_Y_AXIS	1	Y 軸，其對應由 configMachineAxisMapping() 決定。
ECAT_MACHINE_Z_AXIS	2	Z 軸，其對應由 configMachineAxisMapping() 決定。

- *[in] uint8_t priority*

  設定復歸的優先順序。數值範圍為 0 至 255，數值越小表示優先順序越高。若數值相同，則無法保證其執行順序。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 EthercatMaster::begin() 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S slave;

void setup() {
  master.begin();
  slave.attach(0, master);

  slave.configMachineHomingPriority(ECAT_MACHINE_X_AXIS, 1);
  slave.configMachineHomingPriority(ECAT_MACHINE_Y_AXIS, 2);
  slave.configMachineHomingPriority(ECAT_MACHINE_Z_AXIS, 3);
  // ...
}
```

```
void loop() {
  // ...
}
```

configMachineMaxVelocity()

說明

設定 EtherCAT 從站於 G-code 控制器模式下，指定機械軸的最大速度。此參數主要目的是限制馬達的最大速度，以避免超出其規格限制。此參數會寫入 EtherCAT 從站的 EEPROM，並於啟動時載入，因此使用者無需在每次執行程式前進行設定。

語法

```
int configMachineMaxVelocity(int machine_axis, double mm_per_sec);
```

參數

- *[in] int machine_axis*

  指定 EtherCAT 從站上的機器軸號：

定義	值	說明
ECAT_MACHINE_X_AXIS	0	X 軸，其對應由 configMachineAxisMapping() 決定。
ECAT_MACHINE_Y_AXIS	1	Y 軸，其對應由 configMachineAxisMapping() 決定。
ECAT_MACHINE_Z_AXIS	2	Z 軸，其對應由 configMachineAxisMapping() 決定。

- *[in] double mm_per_sec*

  要設定的最大速度，單位為毫米/秒。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 EthercatMaster::begin() 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.configMachineMaxVelocity(ECAT_MACHINE_X_AXIS, 300);
  slave.configMachineMaxVelocity(ECAT_MACHINE_Y_AXIS, 300);
  slave.configMachineMaxVelocity(ECAT_MACHINE_Z_AXIS, 300);
}
void loop() {
  // ...
}
```

configMachineMaxAcceleration()

說明

設定 EtherCAT 從站於 G-code 控制器模式下的最大加速度。此參數會寫入 EtherCAT 從站的 EEPROM，並於啟動時載入，因此使用者無需在每次執行程式前進行設定。

語法

```
int configMachineMaxAcceleration(double mm_per_sec2);
```

參數

- *[in] double mm_per_sec2*
  要設定的最大加速度，單位為毫米/平方秒（mm/s²）。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S slave;

void setup() {
  master.begin();
  slave.attach(0, master);

  slave.configMachineMaxAcceleration(100000);
  // ...
}

void loop() {
  // ...
}
```

configMachineSoftLimit()

說明

設定 EtherCAT 從站於 G-code 控制器模式下，指定機器軸的軟體行程限制。這些參數會寫入 EtherCAT 從站的 EEPROM，並於啟動時載入，因此使用者無需在每次執行程式前進行設定。

當使用者透過 machineGcode() 發出移動命令時，EtherCAT 從站會內部計算每個軸的目標位置：

- 若某軸的目標位置 P 大於該軸的最大位置值 P_{max}，則該目標位置將被調整為 P_{max}。
- 若某軸的目標位置 P 小於該軸的最小位置值 P_{min} ，則該目標位置將被調整為 P_{min}。

語法

```
int configMachineSoftLimit(int machine_axis, double min, double max);
```

參數

- *[in] int machine_axis*

  指定 EtherCAT 從站上的機器軸號：

定義	值	說明
ECAT_MACHINE_X_AXIS	0	X 軸，其對應由 configMachineAxisMapping() 決定。
ECAT_MACHINE_Y_AXIS	1	Y 軸，其對應由 configMachineAxisMapping() 決定。
ECAT_MACHINE_Z_AXIS	2	Z 軸，其對應由 configMachineAxisMapping() 決定。

- *[in] double min*

  欲設定的軟體限制最小位置值 P_{min}，單位為毫米。

- *[in] double max*

  欲設定的軟體限制最大位置值 P_{max}，單位為毫米。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 EthercatMaster::begin() 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S slave;

void setup() {
  master.begin();
  slave.attach(0, master);

  slave.configMachineSoftLimit(ECAT_MACHINE_X_AXIS, 0, 100);
  slave.configMachineSoftLimit(ECAT_MACHINE_Y_AXIS, 0, 100);
```

```
  slave.configMachineSoftLimit(ECAT_MACHINE_Z_AXIS, 0, 100);
}

void loop() {
  // ...
}
```

configMachinePPU()

說明

設定 EtherCAT 從站於 G-code 控制器模式下，指定機器軸的 PPU (Pulse Per Unit) 。此參數用以定義機器移動與馬達之間的對應關係。此參數會寫入 EtherCAT 從站的 EEPROM，並於啟動時載入，因此使用者無需在每次執行程式前進行設定。

語法

```
int configMachinePPU(int machine_axis, double pulses_per_mm);
```

參數

- *[in] int machine_axis*

  指定 EtherCAT 從站上的機器軸號：

定義	值	說明
ECAT_MACHINE_X_AXIS	0	X 軸，其對應由 configMachineAxisMapping() 決定。
ECAT_MACHINE_Y_AXIS	1	Y 軸，其對應由 configMachineAxisMapping() 決定。
ECAT_MACHINE_Z_AXIS	2	Z 軸，其對應由 configMachineAxisMapping() 決定。

- *[in] double pulses_per_mm*

  欲設定的 PPU（每毫米所需的脈波數），單位為 pulses/mm。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 EthercatMaster::begin() 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S slave;

void setup() {
  master.begin();
  slave.attach(0, master);
  slave.configMachinePPU(ECAT_MACHINE_X_AXIS, 100);
  slave.configMachinePPU(ECAT_MACHINE_Y_AXIS, 100);
  slave.configMachinePPU(ECAT_MACHINE_Z_AXIS, 100);
}
void loop() {
  // ...
}
```

configMachineAxisDirection()

說明

設定 EtherCAT 從站於 G-code 控制器模式下，指定機器軸的馬達旋轉方向。此參數會寫入 EtherCAT 從站的 EEPROM，並於啟動時載入，因此使用者無需在每次執行程式前進行設定。

語法

```
int configMachineAxisDirection(int machine_axis, int invert);
```

參數

- *[in] int machine_axis*
  指定 EtherCAT 從站上的機器軸號：

定義	值	說明
ECAT_MACHINE_X_AXIS	0	X 軸，其對應由 configMachineAxisMapping() 決定。
ECAT_MACHINE_Y_AXIS	1	Y 軸，其對應由 configMachineAxisMapping() 決定。
ECAT_MACHINE_Z_AXIS	2	Z 軸，其對應由 configMachineAxisMapping() 決定。

- *[in] int invert*
  要設定的馬達方向。值為 0 表示正常方向，其他任何值則表示反轉方向。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

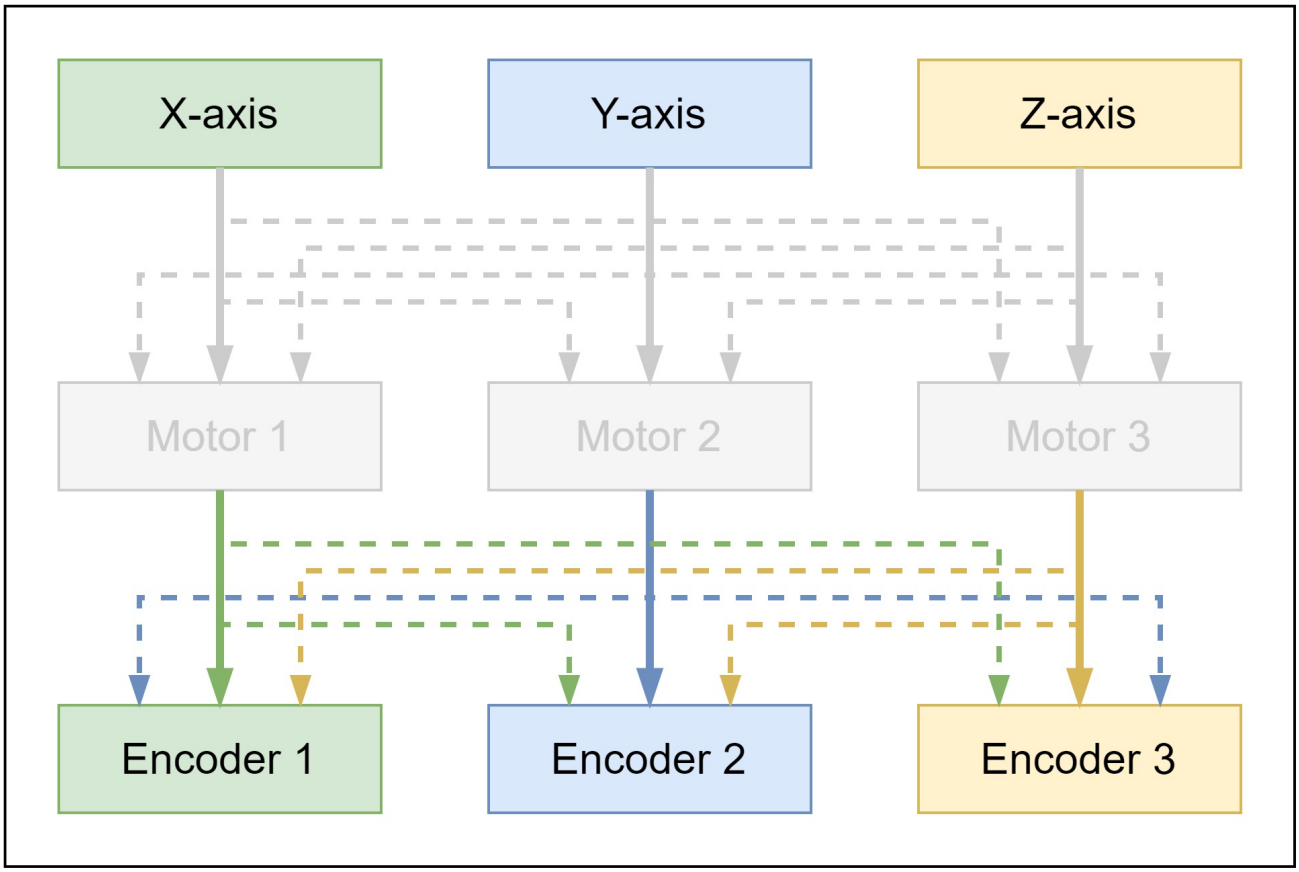
範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S slave;

void setup() {
  master.begin();
  slave.attach(0, master);

  slave.configMachineAxisDirection(ECAT_MACHINE_X_AXIS, 0);
  slave.configMachineAxisDirection(ECAT_MACHINE_Y_AXIS, 0);
  slave.configMachineAxisDirection(ECAT_MACHINE_Z_AXIS, 0);
}
void loop() {
  // ...
}
```

configMachinePositionFeedbackSource()

說明

設定 EtherCAT 從站上指定機器軸的回授來源（Position Feedback Source）。



語法

```
int configMachinePositionFeedbackSource(int machine_axis, int encoder);
```

參數

- *[in] int machine_axis*

  指定 EtherCAT 從站上的機器軸號：

定義	值	說明
ECAT_MACHINE_X_AXIS	0	X 軸，其對應由 configMachineAxisMapping() 決定。
ECAT_MACHINE_Y_AXIS	1	Y 軸，其對應由 configMachineAxisMapping() 決定。
ECAT_MACHINE_Z_AXIS	2	Z 軸，其對應由 configMachineAxisMapping() 決定。

- *[in] encoder*

  要設定的位置回授來源：

定義	值	說明
ECAT_ENCODER_1	0x01	*EtherCAT 從站上的編碼器 1*
ECAT_ENCODER_2	0x02	*EtherCAT 從站上的編碼器 2*

ECAT_ENCODER_3	0x03	*EtherCAT 從站上的編碼器 3*
Other Value	...	停用（Disabled）

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configMachinePositionFeedbackSource(ECAT_MACHINE_X_AXIS,
ECAT_ENCODER_1);
  device.configMachinePositionFeedbackSource(ECAT_MACHINE_Y_AXIS,
ECAT_ENCODER_2);
  device.configMachinePositionFeedbackSource(ECAT_MACHINE_Z_AXIS,
ECAT_ENCODER_3);
  // ...
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

configMachinePositionFeedbackScale()

說明

設定 EtherCAT 從站上指定機器軸的位置回授比例。位置回授公式如下：

Position Feedback = (Encoder Raw Data × Scale) + Offset

Position Feedback = (編碼器原始數值 × Scale) + Offset

其中，Scale 表示一個縮放係數，用於將編碼器的數位讀數轉換為實際的物理單位。此比例建立了編碼器計數值與實際位移或旋轉之間的對應關係。

該參數將寫入 EtherCAT 從站的 EEPROM 中，並於啟動時加載，因此使用者無需每次執行程式前重新設定。

語法

```
int configMachinePositionFeedbackScale(int machine_axis, double scale);
```

參數

- *[in] int machine_axis*

  指定 EtherCAT 從站上的機器軸號：

定義	值	說明
ECAT_MACHINE_X_AXIS	0	X 軸，其對應由 configMachineAxisMapping() 決定。
ECAT_MACHINE_Y_AXIS	1	Y 軸，其對應由 configMachineAxisMapping() 決定。
ECAT_MACHINE_Z_AXIS	2	Z 軸，其對應由 configMachineAxisMapping() 決定。

- *[in] scale*

  要配置的位置回授比例。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 EthercatMaster::begin() 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configMachinePositionFeedbackScale(ECAT_MACHINE_X_AXIS, 2.0);
```

```
  device.configMachinePositionFeedbackScale(ECAT_MACHINE_Y_AXIS, 2.0);
  device.configMachinePositionFeedbackScale(ECAT_MACHINE_Z_AXIS, 2.0);
  // ...
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

configMachinePositionFeedbackOffset()

說明

設定 EtherCAT 從站上指定機器軸的位置回授偏移量。位置回授公式如下：

Position Feedback = (Encoder Raw Data × Scale) + Offset

Position Feedback = (編碼器原始數值 × Scale) + Offset

其中，Offset 表示相對於參考點的初始位移或起始位置。它本質上是一個校正因子，用於補償編碼器零點讀數與實際物理零點之間的誤差。

該參數將寫入 EtherCAT 從站的 EEPROM 中，並於啟動時加載，因此使用者無需每次執行程式前重新設定。

語法

```
int configMachinePositionFeedbackOffset(int machine_axis, double offset);
```

參數

- *[in] machine_axis*

  指定 EtherCAT 從站上的機器軸號：

定義	值	說明
ECAT_MACHINE_X_AXIS	0	X 軸，其對應由 configMachineAxisMapping() 決定。
ECAT_MACHINE_Y_AXIS	1	Y 軸，其對應由 configMachineAxisMapping() 決定。
ECAT_MACHINE_Z_AXIS	2	Z 軸，其對應由 configMachineAxisMapping() 決定。

- *[in] offset*

  要配置的位置回饋偏移。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 EthercatMaster::begin() 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configMachinePositionFeedbackOffset(ECAT_MACHINE_X_AXIS, 10.0);
```

```
  device.configMachinePositionFeedbackOffset(ECAT_MACHINE_Y_AXIS, 10.0);
  device.configMachinePositionFeedbackOffset(ECAT_MACHINE_Z_AXIS, 10.0);
  // ...
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

configMachineStepLossCompensationMode()

說明

在 G-code 控制器模式下，設定指定 EtherCAT 從站機台軸的步進遺失補償模式。此參數會寫入 EtherCAT 從站的 EEPROM，並在啟動時載入，因此使用者無需在每次執行程式前重新設定此參數。

語法

```
int configMachineStepLossCompensationMode(int machine_axis, uint8_t mode);
```

參數

- *[in] machine_axis*

  指定 EtherCAT 從站上的機器軸號：

定義	值	說明
ECAT_MACHINE_X_AXIS	0	X 軸，其對應由 configMachineAxisMapping() 決定。
ECAT_MACHINE_Y_AXIS	1	Y 軸，其對應由 configMachineAxisMapping() 決定。
ECAT_MACHINE_Z_AXIS	2	Z 軸，其對應由 configMachineAxisMapping() 決定。

- *[in] mode*

  T 要設定的步進遺失補償模式：
  - 0：停用步進遺失補償。
  - 1：啟用步進遺失補償。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 EthercatMaster::begin() 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configMachineStepLossCompensationMode(ECAT_MACHINE_X_AXIS, 1);
  device.configMachineStepLossCompensationMode(ECAT_MACHINE_Y_AXIS, 1);
  device.configMachineStepLossCompensationMode(ECAT_MACHINE_Z_AXIS, 1);
  // ...
}
```

```
void loop() {
  // put your main code here, to run repeatedly:

}
```

configMachineStepLossCompensationMaxError()

說明

在 G-code 控制器模式下，設定指定 EtherCAT 從站機台軸的步進遺失補償所允許的最大位置誤差。此參數會寫入 EtherCAT 從站的 EEPROM，並在啟動時自動載入，因此使用者無需在每次執行程式前重新設定此參數。

語法

```
int configMachineStepLossCompensationMaxError(int machine_axis, double max_error);
```

參數

- *[in] machine_axis*
  指定 EtherCAT 從站上的機器軸號：

定義	值	說明
ECAT_MACHINE_X_AXIS	0	X 軸，其對應由 configMachineAxisMapping() 決定。
ECAT_MACHINE_Y_AXIS	1	Y 軸，其對應由 configMachineAxisMapping() 決定。
ECAT_MACHINE_Z_AXIS	2	Z 軸，其對應由 configMachineAxisMapping() 決定。

- *[in] max_error*
  要設定的步進遺失補償的最大位置誤差值。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 EthercatMaster::begin() 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configMachineStepLossCompensationMaxError(ECAT_MACHINE_X_AXIS, 20.0);
  device.configMachineStepLossCompensationMaxError(ECAT_MACHINE_Y_AXIS, 20.0);
  device.configMachineStepLossCompensationMaxError(ECAT_MACHINE_Z_AXIS, 20.0);
```

```
  // ...
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

configMachineSelfStartingSpeed()

說明

設定指定 EtherCAT 從站機台軸的自起動速度（Self-Starting Speed）。

自起動速度表示步進馬達在沒有使用任何加速斜率的情況下，能夠從靜止瞬間加速到穩定運轉的最高步進脈波頻率。如果超過此速度，馬達可能會失步或無法啟動。

語法

```
int configMachineSelfStartingSpeed(int machine_axis, double rev_per_min);
```

參數

- *[in] machine_axis*

  指定 EtherCAT 從站上的機器軸號：

定義	值	說明
ECAT_MACHINE_X_AXIS	0	X 軸，其對應由 configMachineAxisMapping() 決定。
ECAT_MACHINE_Y_AXIS	1	Y 軸，其對應由 configMachineAxisMapping() 決定。
ECAT_MACHINE_Z_AXIS	2	Z 軸，其對應由 configMachineAxisMapping() 決定。

- *[in] rev_per_min*

  要設定的自起動速度，單位為 RPM (Revolutions Per Minute)。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configMachineSelfStartingSpeed(ECAT_MACHINE_X_AXIS, 60.0);
  device.configMachineSelfStartingSpeed(ECAT_MACHINE_Y_AXIS, 60.0);
  device.configMachineSelfStartingSpeed(ECAT_MACHINE_Z_AXIS, 60.0);
  // ...
}

void loop() {
```

```
  // put your main code here, to run repeatedly:

}
```

configMachineG54WorkOffset()

說明

在 G-code 控制器模式下，設定 EtherCAT 從站中 G54 工件座標系統相對於機器座標系統的偏移量。

語法

```
int configMachineG54WorkOffset(double x_offset, double y_offset, double z_offset);
```

參數

- *[in] x_offset*
  要設定的 X 軸工件座標偏移量，單位為毫米 (mm)。
- *[in] y_offset*
  要設定的 Y 軸工件座標偏移量，單位為毫米 (mm)。
- *[in] z_offset*
  要設定的 Z 軸工件座標偏移量，單位為毫米 (mm)。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configMachineG54WorkOffset(10.0, 10.0, 10.0);
  // ...
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

configMachineG55WorkOffset()

說明

在 G-code 控制器模式下，設定 EtherCAT 從站中 G55 工件座標系統相對於機器座標系統的偏移量。

語法

```
int configMachineG55WorkOffset(double x_offset, double y_offset, double z_offset);
```

參數

- *[in] x_offset*
  要設定的 X 軸工件座標偏移量，單位為毫米 (mm)。
- *[in] y_offset*
  要設定的 Y 軸工件座標偏移量，單位為毫米 (mm)。
- *[in] z_offset*
  要設定的 Z 軸工件座標偏移量，單位為毫米 (mm)。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configMachineG55WorkOffset(10.0, 10.0, 10.0);
  // ...
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

configMachineG56WorkOffset()

說明

在 G-code 控制器模式下，設定 EtherCAT 從站中 G56 工件座標系統相對於機器座標系統的偏移量。

語法

```
int configMachineG56WorkOffset(double x_offset, double y_offset, double z_offset);
```

參數

- *[in] x_offset*
  要設定的 X 軸工件座標偏移量，單位為毫米 (mm)。
- *[in] y_offset*
  要設定的 Y 軸工件座標偏移量，單位為毫米 (mm)。
- *[in] z_offset*
  要設定的 Z 軸工件座標偏移量，單位為毫米 (mm)。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configMachineG56WorkOffset(10.0, 10.0, 10.0);
  // ...
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

configMachineG57WorkOffset()

說明

在 G-code 控制器模式下，設定 EtherCAT 從站中 G57 工件座標系統相對於機器座標系統的偏移量。

語法

```
int configMachineG57WorkOffset(double x_offset, double y_offset, double z_offset);
```

參數

- *[in] x_offset*
  要設定的 X 軸工件座標偏移量，單位為毫米 (mm)。
- *[in] y_offset*
  要設定的 Y 軸工件座標偏移量，單位為毫米 (mm)。
- *[in] z_offset*
  要設定的 Z 軸工件座標偏移量，單位為毫米 (mm)。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configMachineG57WorkOffset(10.0, 10.0, 10.0);
  // ...
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

configMachineG58WorkOffset()

說明

在 G-code 控制器模式下，設定 EtherCAT 從站中 G58 工件座標系統相對於機器座標系統的偏移量。

語法

```
int configMachineG58WorkOffset(double x_offset, double y_offset, double z_offset);
```

參數

- *[in] x_offset*
  要設定的 X 軸工件座標偏移量，單位為毫米 (mm)。
- *[in] y_offset*
  要設定的 Y 軸工件座標偏移量，單位為毫米 (mm)。
- *[in] z_offset*
  要設定的 Z 軸工件座標偏移量，單位為毫米 (mm)。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configMachineG58WorkOffset(10.0, 10.0, 10.0);
  // ...
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

configMachineG59WorkOffset()

說明

在 G-code 控制器模式下，設定 EtherCAT 從站中 G59 工件座標系統相對於機器座標系統的偏移量。

語法

```
int configMachineG58WorkOffset(double x_offset, double y_offset, double z_offset);
```

參數

- *[in] x_offset*
  要設定的 X 軸工件座標偏移量，單位為毫米 (mm)。
- *[in] y_offset*
  要設定的 Y 軸工件座標偏移量，單位為毫米 (mm)。
- *[in] z_offset*
  要設定的 Z 軸工件座標偏移量，單位為毫米 (mm)。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configMachineG59WorkOffset(10.0, 10.0, 10.0);
  // ...
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

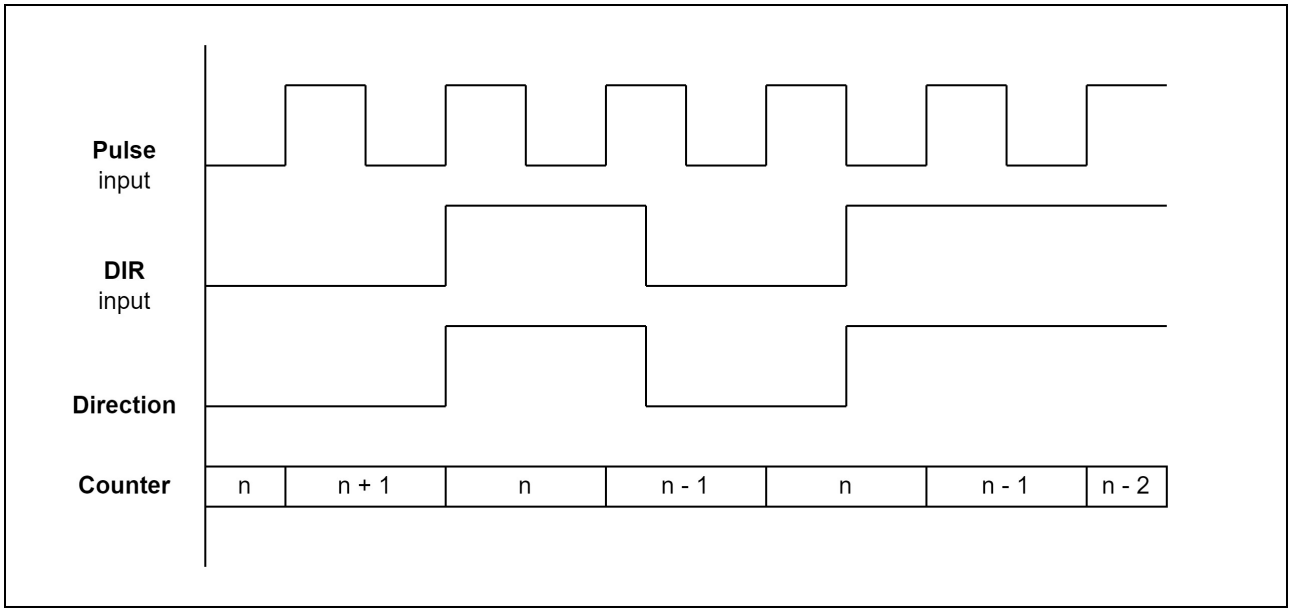
configEncoderMode()

說明

設定 EtherCAT 從站上指定編碼器的編碼器模式。此參數會寫入 EtherCAT 從站的 EEPROM，並在啟動時載入，因此使用者無需在每次執行程式前重新設定此參數。

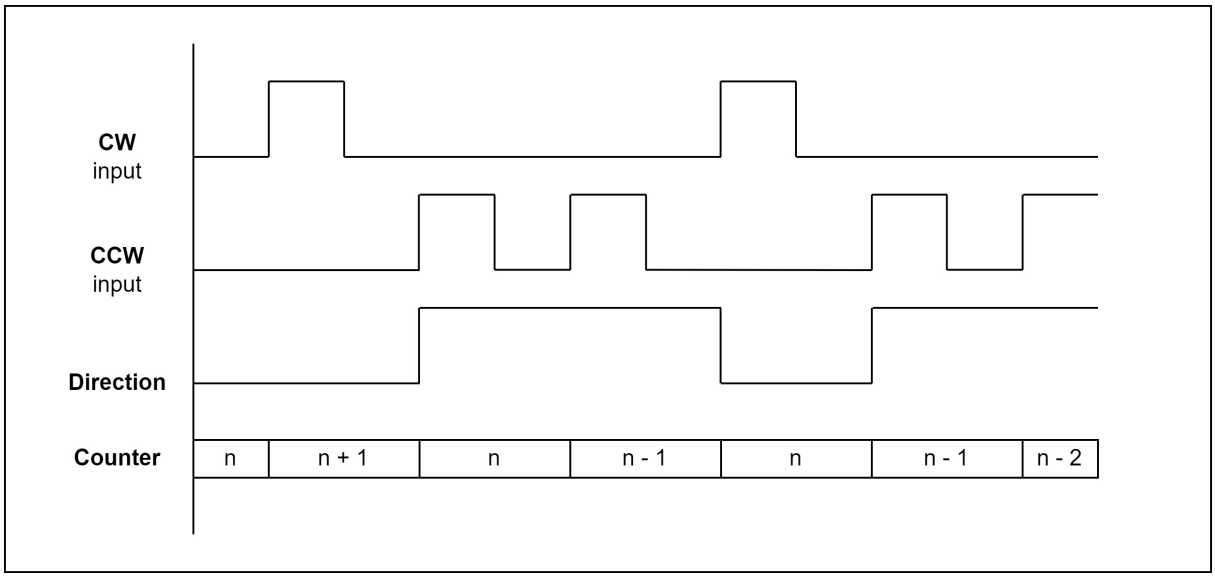
此 EtherCAT 從站共支援六種編碼器模式，如下所示：

- Pulse/DIR (*1-Pulse* mode)
  此模式亦稱為 1-Pulse 模式，使用兩個輸入腳位接收來自編碼器的訊號：Pulse 輸入與 DIR 輸入。Pulse 輸入用於計算脈波數量，DIR 輸入則表示計數方向。在此模式下，計數器僅在 Pulse 輸入的上升緣進行計數。



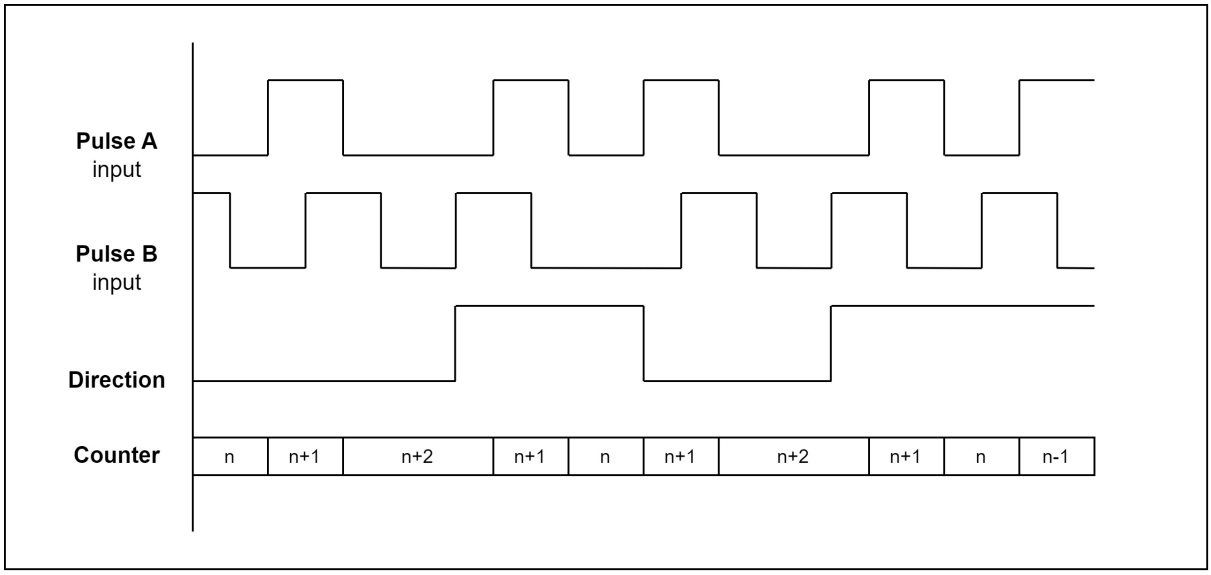
- CW/CCW (*2-Pulse* mode)

  此模式亦稱為 2-Pulse 模式，使用兩個輸入腳位接收來自編碼器的訊號：CW 輸入與 CCW 輸入。CW 輸入的脈波代表正向計數，CCW 輸入的脈波代表反向計數。在此模式下，計數器僅在 CW 與 CCW 輸入的上升緣進行計數。



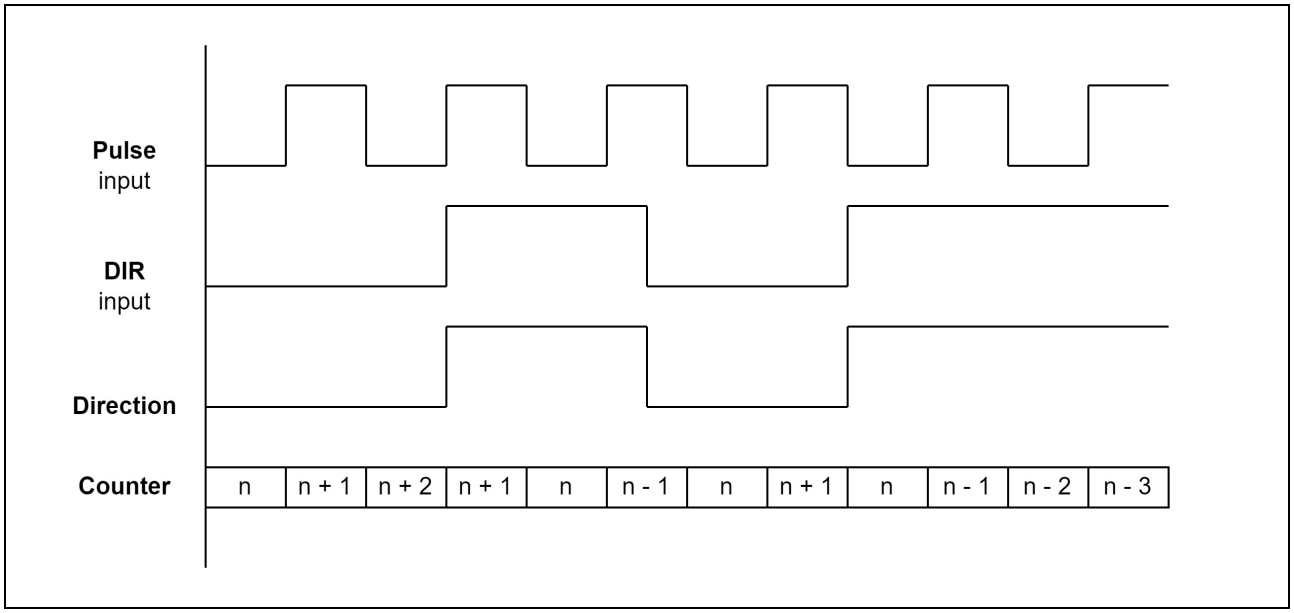
- Pulse A/B (*QEI* mode)

  此模式亦稱為 QEI（正交編碼器介面）模式，使用兩個輸入腳位：Pulse A 與 Pulse B。Pulse A 與 Pulse B 的相位關係決定計數方向：若 A 領先 B，則為正向；若 A 落後 B，則為反向。在此模式下，計數器僅在 Pulse A 的上升與下降緣進行計數。



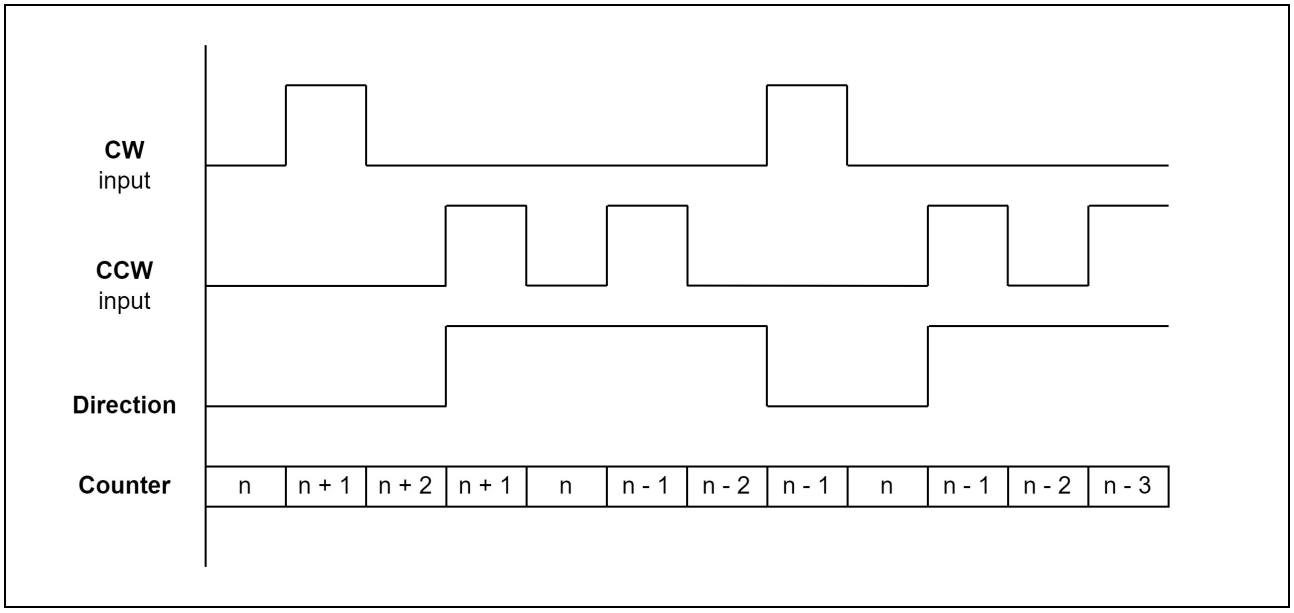
- Pulse/DIR x2

  與 Pulse/DIR 模式類似，但計數器會在 Pulse 輸入的上升與下降緣皆進行計數。



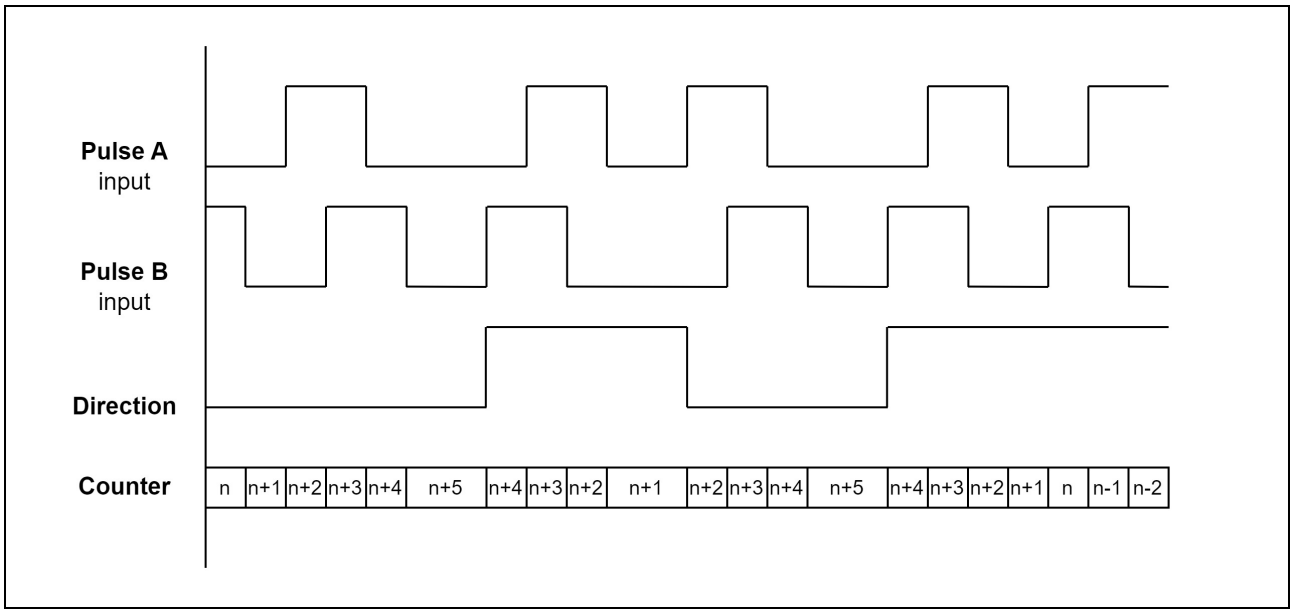
- CW/CCW x2

  與 CW/CCW 模式類似，但計數器會在 CW 與 CCW 輸入的上升與下降緣皆進行計數。



- Pulse A/B x2

  與 Pulse A/B 模式類似，但計數器會在 Pulse A 與 Pulse B 的上升與下降緣皆進行計數。



語法

```
int configEncoderMode(int encoder, uint8_t mode);
```

參數

- *[in] int encoder*

  指定的編碼器編號：

定義	值	說明
ECAT_ENCODER_1	0x01	EtherCAT 從站上的編碼器 1。
ECAT_ENCODER_2	0x02	EtherCAT 從站上的編碼器 2。
ECAT_ENCODER_3	0x03	EtherCAT 從站上的編碼器 3。
ECAT_ENCODER_X	0x11	映射到 X 軸的編碼器。
ECAT_ENCODER_Y	0x12	映射到 Y 軸的編碼器。
ECAT_ENCODER_Z	0x13	映射到 Z 軸的編碼器。

  編碼器到機械軸的對應由 `configMachineAxisMapping()` 和 `configMachinePositionFeedbackSource()` 決定。

- *[in] uint8_t mode*

  要設定的編碼器模式。使用者可從下表所列的六種編碼器模式中選擇。若輸入值未在表中定義，則預設為 ECAT_ENCODER_MODE_AB_PHASE_x2。

定義	值	說明
ECAT_ENCODER_MODE_STEP_DIR	0	請參考 Pulse/DIR 模式。
ECAT_ENCODER_MODE_CWCCW	1	請參考 CW/CCW 模式。
ECAT_ENCODER_MODE_AB_PHASE	2	請參考 Pulse A/B 模式。
ECAT_ENCODER_MODE_STEP_DIR_x2	5	請參考 Pulse/DIR x2 模式。
ECAT_ENCODER_MODE_CWCCW_x2	6	請參考 CW/CCW x2 模式。
ECAT_ENCODER_MODE_AB_PHASE_x2	7	請參考 Pulse A/B x2 模式。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

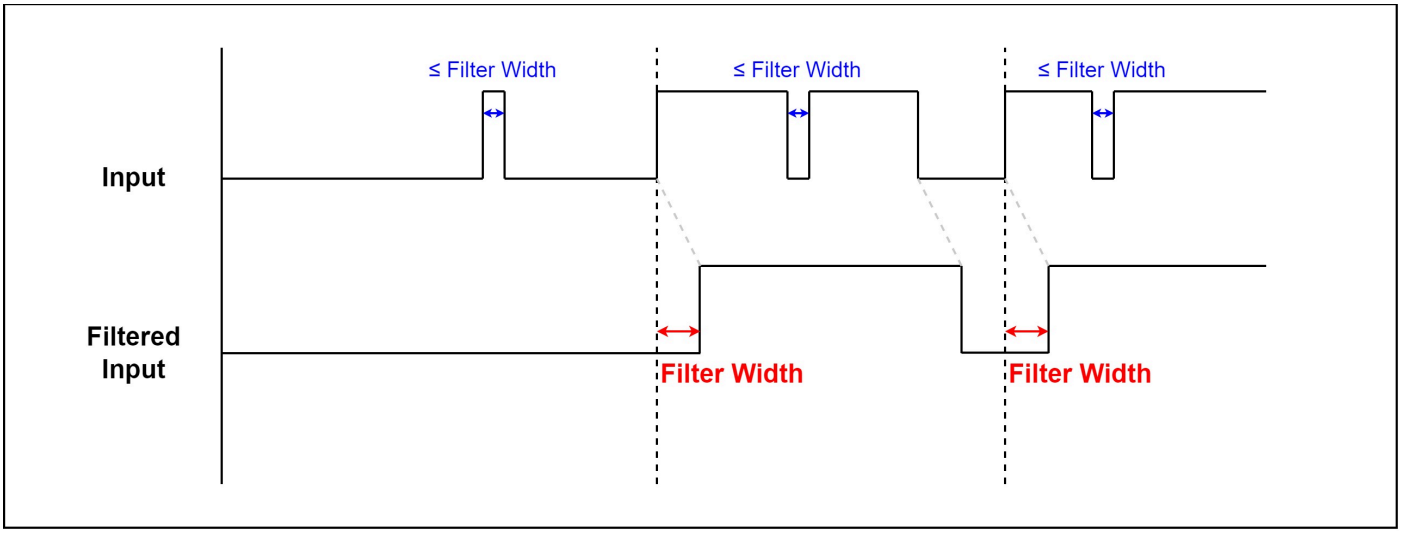
範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configEncoderMode(ECAT_ENCODER_1, ECAT_ENCODER_MODE_AB_PHASE_x2);
  device.configEncoderMode(ECAT_ENCODER_2, ECAT_ENCODER_MODE_AB_PHASE_x2);
  device.configEncoderMode(ECAT_ENCODER_3, ECAT_ENCODER_MODE_AB_PHASE_x2);
  // ...
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

configEncoderDigitalFilter()

說明

設定 EtherCAT 從站中指定編碼器的數位濾波器。如下圖所示，當脈波寬度小於濾波器的頻寬時，該脈波將被濾除；而未被濾除的脈波，會延遲一段與濾波器頻寬相等的時間。此參數會寫入 EtherCAT 從站的 EEPROM，並於啟動時自動載入，因此使用者無需每次執行程式前重新設定此參數。



語法

```
int configEncoderDigitalFilter(int encoder, uint32_t width);
```

參數

- *[in] int encoder*

  指定的編碼器編號：

定義	值	說明
ECAT_ENCODER_1	0x01	EtherCAT 從站上的編碼器 1。
ECAT_ENCODER_2	0x02	EtherCAT 從站上的編碼器 2。
ECAT_ENCODER_3	0x03	EtherCAT 從站上的編碼器 3。
ECAT_ENCODER_X	0x11	映射到 X 軸的編碼器。
ECAT_ENCODER_Y	0x12	映射到 Y 軸的編碼器。
ECAT_ENCODER_Z	0x13	映射到 Z 軸的編碼器。

  編碼器到機械軸的對應由 `configMachineAxisMapping()` 和 `configMachinePositionFeedbackSource()` 決定。

- *[in] uint32_t width*

  要設定的編碼器數位濾波器頻寬，單位為 10 奈秒 (ns)。例如，設定值為 100 時，表示濾波器寬度為 1000 奈秒。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

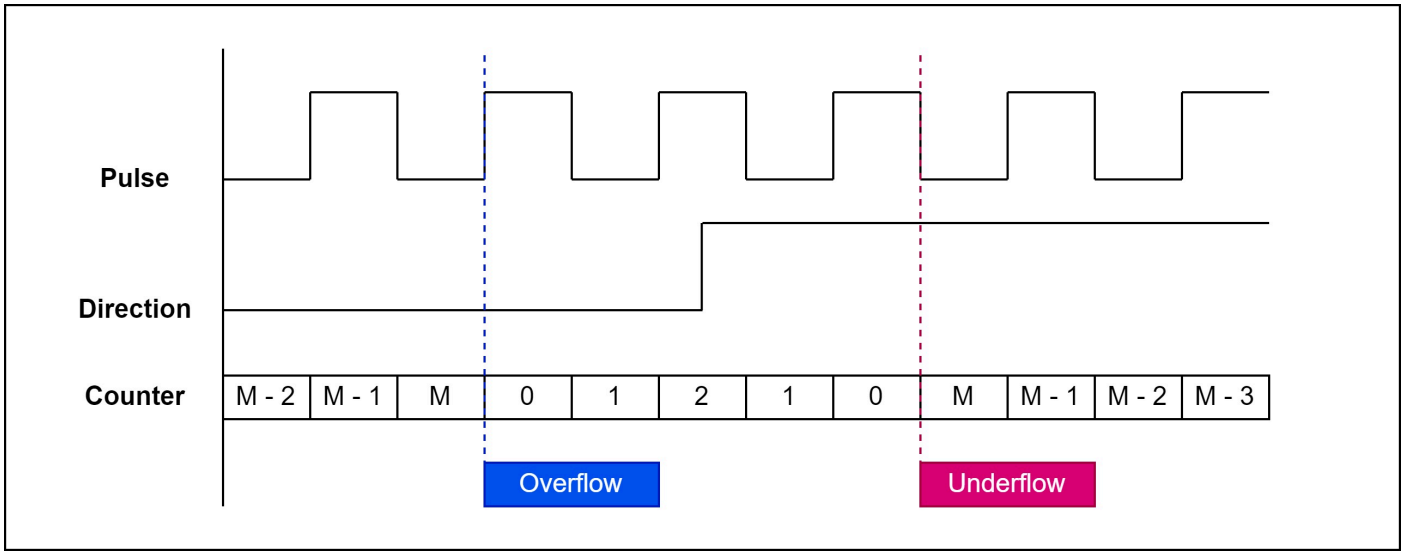
範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configEncoderDigitalFilter(ECAT_ENCODER_1, 100);
  device.configEncoderDigitalFilter(ECAT_ENCODER_2, 100);
  device.configEncoderDigitalFilter(ECAT_ENCODER_3, 100);
  // ...
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

configEncoderRange()

說明

設定 EtherCAT 從站中指定編碼器的計數器最大值。如圖所示，當計數器遞增至最大值 (M) 時，會產生溢位 (*Overflow*)，計數器重設為 0，並觸發 Overflow 事件；相反地，當計數器遞減至 0 時，會產生下溢 (*Underflow*) ，計數器重設為最大值，並觸發 Underflow 事件。
此參數會寫入 EtherCAT 從站的 EEPROM，並於啟動時自動載入，因此使用者無需每次執行程式前重新設定此參數。



語法

```
int configEncoderRange(int encoder, uint32_t value);
```

參數

- *[in] int encoder*
  指定的編碼器編號：

定義	值	說明
ECAT_ENCODER_1	0x01	EtherCAT 從站上的編碼器 1。
ECAT_ENCODER_2	0x02	EtherCAT 從站上的編碼器 2。
ECAT_ENCODER_3	0x03	EtherCAT 從站上的編碼器 3。
ECAT_ENCODER_X	0x11	映射到 X 軸的編碼器。
ECAT_ENCODER_Y	0x12	映射到 Y 軸的編碼器。
ECAT_ENCODER_Z	0x13	映射到 Z 軸的編碼器。

  編碼器到機械軸的對應由 `configMachineAxisMapping()` 和 `configMachinePositionFeedbackSource()` 決定。
- *[in] uint32_t value*
  要設定的編碼器計數器最大值。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configEncoderRange(ECAT_ENCODER_1, 8000);
  device.configEncoderRange(ECAT_ENCODER_2, 8000);
  device.configEncoderRange(ECAT_ENCODER_3, 8000);
  // ...
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

configEncoderInputPolarity()

說明

設定 EtherCAT 從站中指定編碼器輸入腳位的極性。此參數會寫入 EtherCAT 從站的 EEPROM，並於啟動時自動載入，因此使用者無需每次執行程式前重新設定此參數。

語法

```
int configEncoderInputPolarity(int encoder, bool pin_a, bool pin_b, bool pin_z);
```

參數

- *[in] int encoder*

  指定的編碼器編號：

定義	值	說明
ECAT_ENCODER_1	0x01	EtherCAT 從站上的編碼器 1。
ECAT_ENCODER_2	0x02	EtherCAT 從站上的編碼器 2。
ECAT_ENCODER_3	0x03	EtherCAT 從站上的編碼器 3。
ECAT_ENCODER_X	0x11	映射到 X 軸的編碼器。
ECAT_ENCODER_Y	0x12	映射到 Y 軸的編碼器。
ECAT_ENCODER_Z	0x13	映射到 Z 軸的編碼器。

  編碼器到機械軸的對應由 `configMachineAxisMapping()` 和 `configMachinePositionFeedbackSource()` 決定。

- *[in] bool pin_a*

  要設定的編碼器輸入腳位 A 的極性：

  - true：高電位表示邏輯 1，低電位表示邏輯 0。
  - false：高電位表示邏輯 0，低電位表示邏輯 1。

- *[in] bool pin_b*

  要設定的編碼器輸入腳位 B 的極性，解釋同 pin_a。

- *[in] bool pin_z*

  要設定的編碼器輸入腳位 Z 的極性，解釋同 pin_a。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

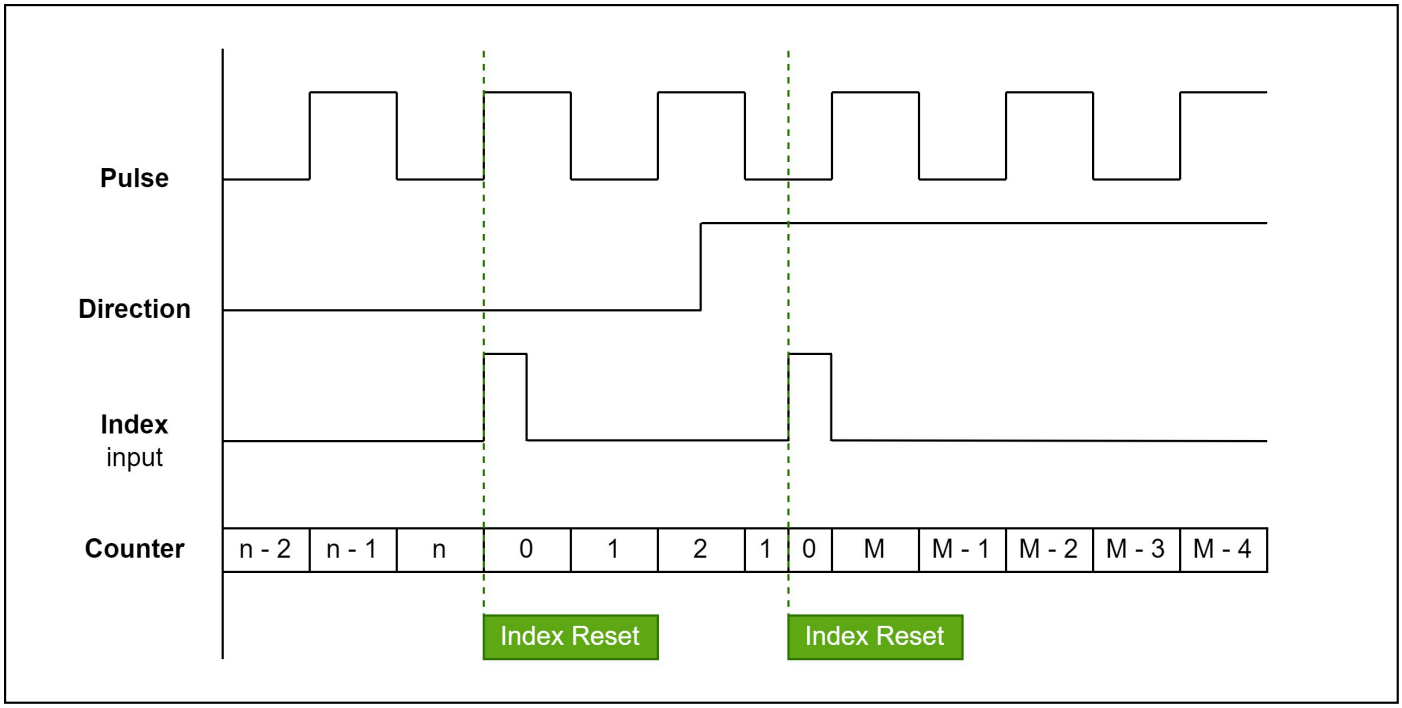
```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;
```

```
void setup() {
  master.begin();
  device.attach(0, master);

  device.configEncoderInputPolarity(ECAT_ENCODER_1, true, true, true);
  device.configEncoderInputPolarity(ECAT_ENCODER_2, true, true, true);
  device.configEncoderInputPolarity(ECAT_ENCODER_3, true, true, true);
  // ...
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

configEncoderIndexReset()

說明

啟用或停用 EtherCAT 從站中指定編碼器的索引訊號（Z 訊號）歸零計數功能。如圖所示，當此功能啟用時，當偵測到索引訊號（Z 訊號）時，計數器將被重設為零 (*Index Reset*)，並觸發 Index Reset 事件。此參數會寫入 EtherCAT 從站的 EEPROM，並於啟動時自動載入，因此使用者無需每次執行程式前重新設定此參數。



語法

```
int configEncoderIndexReset(int encoder, bool enable);
```

參數

- *[in] int encoder*

  指定的編碼器編號：

定義	值	說明
ECAT_ENCODER_1	0x01	EtherCAT 從站上的編碼器 1。
ECAT_ENCODER_2	0x02	EtherCAT 從站上的編碼器 2。
ECAT_ENCODER_3	0x03	EtherCAT 從站上的編碼器 3。
ECAT_ENCODER_X	0x11	映射到 X 軸的編碼器。
ECAT_ENCODER_Y	0x12	映射到 Y 軸的編碼器。
ECAT_ENCODER_Z	0x13	映射到 Z 軸的編碼器。

  編碼器到機械軸的對應由 `configMachineAxisMapping()` 和 `configMachinePositionFeedbackSource()` 決定。

- *[in] bool enable*

  是否啟用索引訊號歸零功能的布林值：

  - true：啟用索引訊號歸零功能。

- false：停用索引訊號歸零功能。

傳回值

回傳錯誤代碼。若回傳值為 0，表示此函式執行成功。

備註

此函式必須在成功執行 `EthercatMaster::begin()` 之後呼叫。此函式為阻塞式，不能在 callback 函式中呼叫。

範例

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configEncoderIndexReset(ECAT_ENCODER_1, false);
  device.configEncoderIndexReset(ECAT_ENCODER_2, false);
  device.configEncoderIndexReset(ECAT_ENCODER_3, false);
  // ...
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

章節. 3

範例

3.1 從站裝置資訊範例

從站裝置資訊範例。

3.1.1 範例 1：使用 EthercatMaster 類別

透過 EthercatMaster 類別顯示從站裝置資訊。

以下是範例程式碼：

```
#include "Ethercat.h"

EthercatMaster master;

void setup() {
  Serial.begin(115200);
  while (!Serial);

  master.begin(); // Initialize EtherCAT Master in Pre-OP state

  Serial.println("Starting EtherCAT Master...");

  // Print Out All Slave Information
  for (int i = 0; i < master.getSlaveCount(); i++) {
    Serial.print("Slave ");
    Serial.print(i);
    Serial.print(" VID: ");
    Serial.print(master.getVendorID(i), HEX);
    Serial.print(", PID: ");
    Serial.print(master.getProductCode(i), HEX);
    Serial.print(", Rev: ");
    Serial.print(master.getRevisionNumber(i), HEX);
    Serial.print(", Ser: ");
    Serial.print(master.getSerialNumber(i), HEX);
    Serial.print(", Alias: ");
    Serial.print(master.getAliasAddress(i));
    Serial.println();
  }
}
```

```
void loop() {
  // put your main code here, to run repeatedly:

}
```

3.1.2 範例 2：使用 EthercatDevice_Generic 類別

透過 EthercatDevice_Generic 類別顯示從站裝置資訊。

以下是範例程式碼：

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

char name[256];

void setup() {
  Serial.begin(115200);
  while (!Serial);

  master.begin(); // Initialize EtherCAT Master in Pre-Operational state

  for (int i = 0; i < master.getSlaveCount(); i++) {
    slave.attach(i, master); // Attach the slave to the master
    Serial.print("Slave ");
    Serial.println(i);

    Serial.print("               Name: ");
    Serial.println(slave.getDeviceName(name, 256));

    Serial.print("          Vendor ID: 0x");
    Serial.println(slave.getVendorID(), HEX);

    Serial.print("       Product Code: 0x");
    Serial.println(slave.getProductCode(), HEX);

    Serial.print("    Revision Number: 0x");
    Serial.println(slave.getRevisionNumber(), HEX);

    Serial.print("      Serial Number: 0x");
    Serial.println(slave.getSerialNumber(), HEX);

    Serial.print("      Alias Address: ");
    Serial.println(slave.getAliasAddress());
```

```
    Serial.print("    Mailbox Protocol: 0x");
    Serial.println(slave.getMailboxProtocol(), HEX);

    Serial.print("        CoE Details: 0x");
    Serial.println(slave.getCoEDetails(), HEX);

    Serial.print("        FoE Details: 0x");
    Serial.println(slave.getFoEDetails(), HEX);

    Serial.print("        EoE Details: 0x");
    Serial.println(slave.getEoEDetails(), HEX);

    Serial.print("       SoE Channels: ");
    Serial.println(slave.getSoEChannels());

    Serial.print("       DC Supported: ");
    Serial.println(slave.isSupportDC());
  }
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

3.1.3 範例 3：使用 EthercatDevice_CiA402 類別

透過 EthercatDevice_CiA402 類別顯示子裝置資訊。

以下是範例程式碼：

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 slave;

char name[256];

void setup() {
  Serial.begin(115200);
  while (!Serial);

  master.begin();  // Initialize EtherCAT Master in Pre-Operational state

  for (int i = 0; i < master.getSlaveCount(); i++) {
    // Attach the slave to the master
    if (slave.attach(i, master) < 0) {
      continue; // Skip this slave if attachment fails
    }

    Serial.print("Slave ");
    Serial.println(i);

    Serial.print("               Name: ");
    Serial.println(slave.getDeviceName(name, 256));

    Serial.print("          Vendor ID: 0x");
    Serial.println(slave.getVendorID(), HEX);

    Serial.print("       Product Code: 0x");
    Serial.println(slave.getProductCode(), HEX);

    Serial.print("    Revision Number: 0x");
    Serial.println(slave.getRevisionNumber(), HEX);

    Serial.print("      Serial Number: 0x");
    Serial.println(slave.getSerialNumber(), HEX);
```

```
    Serial.print("      Alias Address: ");
    Serial.println(slave.getAliasAddress());

    Serial.print("    Mailbox Protocol: 0x");
    Serial.println(slave.getMailboxProtocol(), HEX);

    Serial.print("        CoE Details: 0x");
    Serial.println(slave.getCoEDetails(), HEX);

    Serial.print("        FoE Details: 0x");
    Serial.println(slave.getFoEDetails(), HEX);

    Serial.print("        EoE Details: 0x");
    Serial.println(slave.getEoEDetails(), HEX);

    Serial.print("       SoE Channels: ");
    Serial.println(slave.getSoEChannels());

    Serial.print("       DC Supported: ");
    Serial.println(slave.isSupportDC());
  }
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

3.2 SDO 上傳/下載範例

SDO 讀取/寫入範例。

3.2.1 範例 1：使用 sdoUpload8() 進行 SDO 讀取

sdoUpload16()、sdoUpload32()、和 sdoUpload64() 的使用方式與 sdoUpload8() 類似，差別僅在於回傳值的型別不同。

以下是範例程式碼：

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);

  Serial.print("1C12h.0 => ");
  Serial.println(device.sdoUpload8(0x1C12, 0x00));

  Serial.print("1C13h.0 => ");
  Serial.println(device.sdoUpload8(0x1C13, 0x00));
  // ...
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

3.2.2 範例 2：使用 sdoUpload() 進行 SDO 讀取

使用 `sdoUpload()` 函式來執行 SDO 讀取操作。

以下是範例程式碼：

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  uint8_t value;

  master.begin();
  device.attach(0, master);

  if (device.sdoUpload(0x1C12, 0x00, &value, sizeof(value)) >= 0) {
    Serial.print("1C12h.0 => ");
    Serial.println(value);
  }

  if (device.sdoUpload(0x1C13, 0x00, &value, sizeof(value)) >= 0) {
    Serial.print("1C13h.0 => ");
    Serial.println(value);
  }
  //...
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

3.2.3 範例 3：使用 sdoUpload() 進行 SDO 讀取並處理中止碼

使用 `sdoUpload()` 進行 SDO 讀取並處理中止碼。
本範例透過 `sdoUpload()` 發起一個 SDO Upload 命令，嘗試讀取一個不存在的物件值，預期會回傳中止碼 0x06020000。有關中止碼的更多資訊，請參閱 SDO Abort Code 章節。

以下是範例程式碼：

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  uint32_t abortcode;
  uint8_t value;

  master.begin();
  device.attach(0, master);

  if (device.sdoUpload(0xFFFF, 0xFF, &value, sizeof(value), &abortcode) ==
ECAT_ERR_DEVICE_COE_ERROR) {
    Serial.print("Abort Code: 0x");
    Serial.println(abortcode, HEX);
  }
  //...
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

3.2.4 範例 4：使用 sdoDownload8() 進行 SDO 寫入

使用 sdoDownload8() 進行 SDO 寫入。
sdoDownload16()、sdoDownload32()、和 sdoDownload64() 的用法與 sdoDownload8() 類似，差異僅在於輸入參數的資料型別不同。

以下是範例程式碼：

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.sdoDownload8(0x1C12, 0x00, 0);
  device.sdoDownload8(0x1C13, 0x00, 0);
  //...
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

3.2.5 範例 5：使用 sdoDownload() 進行 SDO 寫入

使用 sdoDownload() 函式來執行 SDO 寫入操作。

以下是範例程式碼：

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  uint8_t value;

  master.begin();
  device.attach(0, master);

  value = 0;
  device.sdoDownload(0x1C12, 0x00, &value, sizeof(value));
  value = 0;
  device.sdoDownload(0x1C13, 0x00, &value, sizeof(value));
  //...
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

3.2.6 範例 6：使用 sdoDownload() 進行 SDO 寫入並處理中止碼

使用 `sdoDownload()` 進行 SDO 寫入操作並處理中止碼。
使用帶有中止程式碼的 `sdoDownload()` 進行 SDO 寫入。啟動 SDO 寫入指令，將值寫入不存在的對象，預期中止代碼為 0x06020000。有關中止代碼的更多信息，請參閱 SDO Abort Code 章節。

以下是範例程式碼：

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  uint32_t abortcode;
  uint8_t value = 0x01;

  master.begin();
  device.attach(0, master);

  if (device.sdoDownload(0xFFFF, 0xFF, &value, sizeof(value), &abortcode) ==
ECAT_ERR_DEVICE_COE_ERROR) {
    char buf[20];
    sprintf(buf, "Abort Code: 0x%08lX", abortcode);
    Serial.println(buf);
  }
  //...
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

3.2.7 範例 7：列印出 PDO 映射設定

列印 PDO (Process Data Object) 映射的相關設定。

以下是範例程式碼：

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  uint8_t assign_nr, mapping_nr;
  uint16_t mapping;
  uint32_t entry;

  master.begin();
  device.attach(0, master);

  // RxPDO Mapping List
  assign_nr = device.sdoUpload8(0x1C12, 0x00);
  for (int m = 0; m < assign_nr; m++) {
    mapping = device.sdoUpload16(0x1C12, m + 1);
    Serial.print("  RxPDO");
    Serial.print(m + 1);
    Serial.print(" (");
    Serial.print(mapping, HEX);
    Serial.println("h)");

    mapping_nr = device.sdoUpload8(mapping, 0x00);
    for (int n = 0; n < mapping_nr; n++) {
      entry = device.sdoUpload32(mapping, n + 1);
      Serial.print("    ");
      char entryBuf[11];  // "XXXXXXXXh" + null
      sprintf(entryBuf, "%08lXh", entry);
      Serial.println(entryBuf);
    }
  }

  // TxPDO Mapping List
```

```
  assign_nr = device.sdoUpload8(0x1C13, 0x00);
  for (int m = 0; m < assign_nr; m++) {
    mapping = device.sdoUpload16(0x1C13, m + 1);
    Serial.print("  TxPDO");
    Serial.print(m + 1);
    Serial.print(" (");
    Serial.print(mapping, HEX);
    Serial.println("h)");

    mapping_nr = device.sdoUpload8(mapping, 0x00);
    for (int n = 0; n < mapping_nr; n++) {
      entry = device.sdoUpload32(mapping, n + 1);
      Serial.print("    ");
      char entryBuf[11];
      sprintf(entryBuf, "%08lXh", entry);
      Serial.println(entryBuf);
    }
  }
  //...
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

3.2.8 範例 8：更改 PDO 映射設定

更改 PDO 映射設定。
將以下物件對應至支援 PDO 映射的 CiA 402 裝置的 PDO 中：

- ***Output PDO (RxPDO)***
  - Object 6040_h: Controlword
  - Object $607A_h$: Target position
  - Object $60FF_h$: Target velocity
  - Object 6071_h: Target torque
  - Object 6060_h: Modes of operation
- ***Input PDO (TxPDO)***
  - Object 6041_h: Statusword
  - Object 6064_h: Position actual value
  - Object $606C_h$: Velocity actual value
  - Object 6077_h: Torque actual value
  - Object 6061_h: Modes of operation display

以下是範例程式碼：

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  master.begin();
  device.attach(0, master);
  // 1C12
  device.sdoDownload8(0x1C12, 0x00, 0);
  device.sdoDownload8(0x1601, 0x00, 0);
  device.sdoDownload32(0x1601, 0x01, 0x60400010);
  device.sdoDownload32(0x1601, 0x02, 0x607A0020);
  device.sdoDownload32(0x1601, 0x03, 0x60FF0020);
  device.sdoDownload32(0x1601, 0x04, 0x60710010);
  device.sdoDownload32(0x1601, 0x05, 0x60600008);
  device.sdoDownload8(0x1601, 0x00, 5);
  device.sdoDownload16(0x1C12, 0x1601, 1);
  device.sdoDownload8(0x1C12, 0x00, 1);
  // 1C13
  device.sdoDownload8(0x1C13, 0x00, 0);
```

```
  device.sdoDownload8(0x1A01, 0x00, 0);
  device.sdoDownload32(0x1A01, 0x01, 0x60410010);
  device.sdoDownload32(0x1A01, 0x02, 0x60640020);
  device.sdoDownload32(0x1A01, 0x03, 0x606C0020);
  device.sdoDownload32(0x1A01, 0x04, 0x60770010);
  device.sdoDownload32(0x1A01, 0x05, 0x60610008);
  device.sdoDownload8(0x1A01, 0x00, 5);
  device.sdoDownload16(0x1C13, 0x1A01, 1);
  device.sdoDownload8(0x1C13, 0x00, 1);
  //...
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

3.3 PDO 讀/寫範例

PDO 讀/寫範例。

3.3.1 範例 1：使用 pdoBitRead() 讀取輸入 PDO 的 1 位元資料

使用 pdoBitRead() 從輸入 PDO 中讀取單一位元資料。
此範例使用一個具備 16 通道數位輸入的 EtherCAT 從站裝置，其 Input PDO 為 2-Byte，每一個位元對應一個數位輸入通道。通道 0 和 9 的狀態將以 1 Hz 的頻率列印。

B15	B14	B13	B12	B11	B10	**B9**	B8	B7	B6	B5	B4	B3	B2	B1	**B0**

以下是範例程式碼：

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.start();
}

void loop() {
  // Read and display the state of PDO bits
  Serial.print("Bit0 => ");
  Serial.print(device.pdoBitRead(0)); // Read PDO bit 0
  Serial.print(", Bit9 => ");
  Serial.println(device.pdoBitRead(9)); // Read PDO bit 9

  delay(1000);
}
```

3.3.2 範例 2：使用 pdoRead8() 讀取輸入 PDO 的 1 Byte 資料

使用 pdoRead8() 從輸入 PDO 讀取資料。
與範例 1 相同。

以下是範例程式碼：

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.start();
}

void loop() {
  // Read specific bits using pdoRead8 and print their values
  Serial.print("Bit0 => ");
  Serial.print((device.pdoRead8(0) >> 0) & 1);
  Serial.print(", Bit9 => ");
  Serial.println((device.pdoRead8(1) >> 1) & 1);

  delay(1000);
}
```

3.3.3 範例 3：使用 pdoRead() 從輸入 PDO 讀取資料

使用 pdoRead() 從輸入 PDO 讀取資料。
與範例 1 相同。

以下是範例程式碼：

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

uint16_t value;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.start();
}

void loop() {
  device.pdoRead(0, &value, sizeof(value));

  Serial.print("Bit0 => ");
  Serial.print((value >> 0) & 1);
  Serial.print(", Bit9 => ");
  Serial.println((value >> 9) & 1);

  delay(1000);
}
```

3.3.4 範例 4：使用 pdoBitWrite() 寫入輸出 PDO 的 1 位元資料

使用 `pdoBitWrite()` 將單一位元資料寫入到輸出 PDO。
此範例使用一個具備 16 通道數位輸出的 EtherCAT 從站裝置，其 Output PDO 為 2-Byte，每一個位元對應一個數位輸出通道。通道 0 和 9 的狀態將以 1 Hz 的頻率切換。

B15	B14	B13	B12	B11	B10	**B9**	B8	B7	B6	B5	B4	B3	B2	B1	**B0**

以下是範例程式碼：

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  master.begin();
  device.attach(0, master);
  master.start();
}

void loop() {
  device.pdoBitWrite(0, 1);
  device.pdoBitWrite(9, 1);
  delay(1000);

  device.pdoBitWrite(0, 0);
  device.pdoBitWrite(9, 0);
  delay(1000);
}
```

3.3.5 範例 5：使用 pdoWrite8() 寫入輸出 PDO 的 1 Byte 資料

使用 pdoWrite8() 將一個位元組資料寫入到輸出 PDO。
與範例 4 相同。

以下是範例程式碼：

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  master.begin();
  device.attach(0, master);
  master.start();
}

void loop() {
  device.pdoWrite8(0, 0x01);
  device.pdoWrite8(1, 0x02);
  delay(1000);

  device.pdoWrite8(0, 0x00);
  device.pdoWrite8(1, 0x00);
  delay(1000);
}
```

3.3.6 範例 6：使用 pdoWrite() 寫入輸出 PDO 資料

使用 pdoWrite() 從輸出 PDO 寫入資料。
與範例 4 相同。

以下是範例程式碼：

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

uint16_t value;

void setup() {
  master.begin();
  device.attach(0, master);
  master.start();
}

void loop() {
  value = 0x0201;
  device.pdoWrite(0, &value, sizeof(value));
  delay(1000);

  value = 0x0000;
  device.pdoWrite(0, &value, sizeof(value));
  delay(1000);
}
```

3.4 Callback 範例

Callback 範例。

3.4.1 範例 1：Cyclic callback

一個具有 16 通道數位輸出的 EtherCAT 從站，其 Output PDO 佔用 2 個位元組，每一個位元對應一個數位輸出通道。此範例將以 1Hz 的頻率切換第 0 與第 9 通道。

B15	B14	B13	B12	B11	B10	**B9**	B8	B7	B6	B5	B4	B3	B2	B1	**B0**

以下是範例程式碼：

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

int toggle = 0;
int cycle_count = 0;

void MyCyclicCallback()
{
  if (++cycle_count < 1000)
    return;

  cycle_count = 0;

  toggle = !toggle;
  device.pdoBitWrite(0, toggle);
  device.pdoBitWrite(9, toggle);
}

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.attachCyclicCallback(MyCyclicCallback);
  master.start(1000000);  // 1ms cycle
```

```
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

3.4.2 範例 2：啟用 FPU 的 Cyclic callback

啟用 FPU 的 Cyclic callback。

若 callback 函式中需要進行浮點數運算，請使用支援 FPU (浮點運算單元) 的 callback 函式版本。詳細資訊請參考下列函式說明：

- attachCyclicCallback()
- attachErrorCallback()
- attachEventCallback()

以下是範例程式碼：

```
#include "Ethercat.h"

#define GRAVITY (9.80665)

EthercatMaster master;
EthercatDevice_Generic device;

int toggle = 0;
int cycle_count = 0;
double S = 0.0, T = 0.0;

double s = 0.0, t = 0.0;


void MyCyclicCallback()
{
  S = (GRAVITY * T * T) / 2.0;
  T += 0.001;

  if (++cycle_count < 1000)
    return;
  cycle_count = 0;

  toggle = !toggle;
  device.pdoBitWrite(0, toggle);
  device.pdoBitWrite(9, toggle);
}

void setup() {
  Serial.begin(115200);
```

```
  master.begin();
  device.attach(0, master);
  master.attachCyclicCallback(MyCyclicCallback, true);
  master.start(1000000);
}

void loop() {
  s = (GRAVITY * t * t) / 2.0;
  t += 1.0;

  Serial.print("S = ");
  Serial.println(s, 6);

  delay(1000);
}
```

3.4.3 範例 3：Error callback

每秒列印一次每種錯誤類型的計數。

以下是範例程式碼：

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

// Error counters
int wkc_single_fault_cnt = 0;
int wkc_multiple_faults_cnt = 0;
int single_lost_frame_cnt = 0;
int multiple_lost_frames_cnt = 0;
int cable_broken_cnt = 0;
int wait_ack_timeout_cnt = 0;

// Error callback function
void myErroCallback(uint32_t errorcode) {
  switch (errorcode) {
    case ECAT_ERR_WKC_SINGLE_FAULT:
      wkc_single_fault_cnt++;
      break;
    case ECAT_ERR_WKC_MULTIPLE_FAULTS:
      wkc_multiple_faults_cnt++;
      break;
    case ECAT_ERR_SINGLE_LOST_FRAME:
      single_lost_frame_cnt++;
      break;
    case ECAT_ERR_MULTIPLE_LOST_FRAMES:
      multiple_lost_frames_cnt++;
      break;
    case ECAT_ERR_CABLE_BROKEN:
      cable_broken_cnt++;
      break;
    case ECAT_ERR_WAIT_ACK_TIMEOUT:
      wait_ack_timeout_cnt++;
      break;
  }
}
```

```
void setup() {
  Serial.begin(115200);

  master.attachErrorCallback(myErroCallback);
  master.begin();
  master.start();
}

void loop() {

  Serial.print("ECAT_ERR_WKC_SINGLE_FAULT     = ");
  Serial.println(wkc_single_fault_cnt);
  Serial.print("ECAT_ERR_WKC_MULTIPLE_FAULTS  = ");
  Serial.println(wkc_multiple_faults_cnt);
  Serial.print("ECAT_ERR_SINGLE_LOST_FRAME    = ");
  Serial.println(single_lost_frame_cnt);
  Serial.print("ECAT_ERR_MULTIPLE_LOST_FRAMES = ");
  Serial.println(multiple_lost_frames_cnt);
  Serial.print("ECAT_ERR_CABLE_BROKEN        = ");
  Serial.println(cable_broken_cnt);
  Serial.print("ECAT_ERR_WAIT_ACK_TIMEOUT     = ");
  Serial.println(wait_ack_timeout_cnt);

  delay(1000);
}
```

3.4.4 範例 4：Event callback

每秒列印一次每種事件類型的計數。

以下是範例程式碼：

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

// Event counters
int state_changed_cnt = 0;
int cable_reconnected_cnt = 0;

// Event callback function
void myEventCallback(uint32_t eventcode) {
  switch (eventcode) {
    case ECAT_EVT_STATE_CHANGED:
      state_changed_cnt++;
      break;
    case ECAT_EVT_CABLE_RECONNECTED:
      cable_reconnected_cnt++;
      break;
  }
}

void setup() {
  Serial.begin(115200);

  master.attachEventCallback(myEventCallback);
  master.begin();
  master.start();
}

void loop() {
  Serial.print("ECAT_EVT_STATE_CHANGED     = ");
  Serial.println(state_changed_cnt);
  Serial.print("ECAT_EVT_CABLE_RECONNECTED = ");
  Serial.println(cable_reconnected_cnt);
  delay(1000);
}
```

3.5 DC 範例

DC 範例。

3.5.1 範例 1：啟用 DC 同步

啟用 DC 同步。
本範例展示如何使用 EthercatDevice_Generic 類別，在一個支援 CiA 402 的 EtherCAT 從站裝置上實現位置控制。控制模式設定為 Cyclic Synchronous Position (CSP)，並啟用 DC 同步以達成精確的時間同步控制。

預設 PDO 對應如下：

- **Output PDO (RxPDO)**

Byte 0	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**
Controlword		Target Position			

- **Input PDO (TxPDO)**

Byte 0	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**
Statusword		Position Actual Value			

以下是範例程式碼：

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

uint32_t position = 0;

// Cyclic callback function, executed every 1ms
void MyCyclicCallback() {
  // Check if the device is in "Operation Enable" state (0x27)
  if ((device.pdoRead8(0) & 0x6F) != 0x27)
    return;

  // Increment target position by 1000 and write it to PDO
  device.pdoWrite32(2, position += 1000);
}
```

```
void setup() {
  Serial.begin(115200);
  while (!Serial);  // Wait for Serial to be ready

  master.begin();

  device.attach(0, master);            // Attach slave device (slave ID 0)
  device.setDc(1000000);               // Enable Distributed Clock (1ms cycle)
  device.sdoDownload8(0x6060, 0x00, 8); // Set mode of operation to CSP (mode
8)

  // Register the cyclic callback function
  master.attachCyclicCallback(MyCyclicCallback);
  master.start(1000000, ECAT_SYNC);    // Start EtherCAT master with 1ms sync
cycle

  // Read current position and set as initial target position
  position = device.pdoRead32(2);
  device.pdoWrite32(2, position);

  // Transition through CiA 402 state machine to reach "Operation Enable"
  device.pdoWrite8(0, 0x80); // Shutdown
  delay(1000);
  device.pdoWrite8(0, 0x06); // Switch On
  delay(1000);
  device.pdoWrite8(0, 0x07); // Enable Voltage
  delay(1000);
  device.pdoWrite8(0, 0x0F); // Operation Enable
  delay(1000);
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

附錄

A.1 錯誤代碼

對於大多數函式而言，若回傳值小於零，則表示發生錯誤，該值即為錯誤代碼。若出現錯誤代碼，可參考下方內容查詢錯誤原因及對應的修正措施。

定義	程式碼
ECAT_SUCCESS	0
ECAT_ERR_MODULE_INIT_FAIL	-100
ECAT_ERR_MODULE_GET_VERSION_FAIL	-101
ECAT_ERR_MODULE_VERSION_MISMATCH	-102
ECAT_ERR_MODULE_GENERIC_TRANSFER_INIT_FAIL	-103
ECAT_ERR_MASTER_DOWNLOAD_SETTINGS_FAIL	-200
ECAT_ERR_MASTER_SET_DEVICE_SETTINGS_FAIL	-201
ECAT_ERR_MASTER_GET_GROUP_INFO_FAIL	-202
ECAT_ERR_MASTER_GET_MASTER_INFO_FAIL	-203
ECAT_ERR_MASTER_GET_DEVICE_INFO_FAIL	-204
ECAT_ERR_MASTER_SET_GROUP_SETTINGS_FAIL	-205
ECAT_ERR_MASTER_MAPPING_INIT_FAIL	-206
ECAT_ERR_MASTER_INTERRUPT_INIT_FAIL	-207
ECAT_ERR_MASTER_ACTIVE_FAIL	-208
ECAT_ERR_MASTER_ENI_INITCMDS_FAIL	-209
ECAT_ERR_MASTER_NO_DEVICE	-210
ECAT_ERR_MASTER_ACYCLIC_INIT_FAIL	-300
ECAT_ERR_MASTER_ACYCLIC_REQUEST_FAIL	-301
ECAT_ERR_MASTER_ACYCLIC_BUSY	-302
ECAT_ERR_MASTER_ACYCLIC_TIMEOUT	-303
ECAT_ERR_MASTER_ACYCLIC_ERROR	-304
ECAT_ERR_MASTER_ACYCLIC_WRONG_STATUS	-405
ECAT_ERR_MASTER_GENERIC_SEND_FAIL	-400
ECAT_ERR_MASTER_GENERIC_RECV_FAIL	-401
ECAT_ERR_MASTER_NOT_BEGIN	-1000
ECAT_ERR_MASTER_WRONG_BUFFER_SIZE	-1001
ECAT_ERR_MASTER_REDUNDANCY_NO_DC	-1002
ECAT_ERR_MASTER_MEMORY_ALLOCATION_FAIL	-1003
ECAT_ERR_MASTER_OSLAYER_INIT_FAIL	-1004
ECAT_ERR_MASTER_NIC_INIT_FAIL	-1005
ECAT_ERR_MASTER_BASE_INIT_FAIL	-1006
ECAT_ERR_MASTER_CIA402_INIT_FAIL	-1007
ECAT_ERR_MASTER_SETUP_PDO_FAIL	-1008

ECAT_ERR_MASTER_SCAN_NETWORK_FAIL	-1009
ECAT_ERR_MASTER_START_MASTER_FAIL	-1010
ECAT_ERR_MASTER_CYCLETIME_TOO_SMALL	-1011
ECAT_ERR_MASTER_DUMP_OUTPUT_PDO_FAIL	-1012
ECAT_ERR_MASTER_CONFIG_DEVICE_FAIL	-1013
ECAT_ERR_MASTER_CONFIG_MAPPING_FAIL	-1014
ECAT_ERR_MASTER_WAIT_BUS_SYNC_TIMEOUT	-1015
ECAT_ERR_MASTER_WAIT_MASTER_SYNC_TIMEOUT	-1016
ECAT_ERR_MASTER_CYCLIC_START_FAIL	-1017
ECAT_ERR_MASTER_WRONG_BUFFER_POINTER	-1018
ECAT_ERR_MASTER_ENI_INIT_FAIL	-1050
ECAT_ERR_MASTER_ENI_MISMATCH	-1051
ECAT_ERR_MASTER_STOPPED	-1100
ECAT_ERR_MASTER_STARTED	-1101
ECAT_ERR_MASTER_NOT_IN_PREOP	-1102
ECAT_ERR_MASTER_NOT_IN_SAFEOP	-1103
ECAT_ERR_MASTER_NOT_IN_OP	-1104
ECAT_ERR_MASTER_II_TRANSITION_FAIL	-1200
ECAT_ERR_MASTER_IP_TRANSITION_FAIL	-1201
ECAT_ERR_MASTER_PS_TRANSITION_FAIL	-1202
ECAT_ERR_MASTER_SO_TRANSITION_FAIL	-1203
ECAT_ERR_DEVICE_NOT_EXIST	-2000
ECAT_ERR_DEVICE_NOT_ATTACH	-2001
ECAT_ERR_DEVICE_NO_MAILBOX	-2002
ECAT_ERR_DEVICE_NO_DC	-2003
ECAT_ERR_DEVICE_WRONG_INPUT	-2004
ECAT_ERR_DEVICE_MEMORY_ALLOCATION_FAIL	-2005
ECAT_ERR_DEVICE_VENDOR_ID_MISMATCH	-2006
ECAT_ERR_DEVICE_PRODUCT_CODE_MISMATCH	-2007
ECAT_ERR_DEVICE_NO_SUCH_FUNCTION	-2008
ECAT_ERR_DEVICE_FUNCTION_NOT_INIT	-2009
ECAT_ERR_DEVICE_BUSY	-2010
ECAT_ERR_DEVICE_TIMEOUT	-2011
ECAT_ERR_DEVICE_NO_DATA	-2012
ECAT_ERR_DEVICE_SII_READ_FAIL	-2100
ECAT_ERR_DEVICE_SII_WRITE_FAIL	-2101
ECAT_ERR_DEVICE_PDO_NOT_EXIST	-2200
ECAT_ERR_DEVICE_PDO_OUT_OF_RANGE	-2201
ECAT_ERR_DEVICE_FOE_NOT_SUPPORT	-2300
ECAT_ERR_DEVICE_FOE_REQUEST_FAIL	-2310
ECAT_ERR_DEVICE_FOE_TIMEOUT	-2311

ECAT_ERR_DEVICE_FOE_ERROR	-2312
ECAT_ERR_DEVICE_FOE_BUFFER_TOO_SMALL	-2313
ECAT_ERR_DEVICE_FOE_READ_FAIL	-2314
ECAT_ERR_DEVICE_FOE_WRITE_FAIL	-2315
ECAT_ERR_DEVICE_COE_SDO_NOT_SUPPORT	-2400
ECAT_ERR_DEVICE_COE_SDO_INFO_NOT_SUPPORT	-2401
ECAT_ERR_DEVICE_COE_BUSY	-2410
ECAT_ERR_DEVICE_COE_REQUEST_FAIL	-2411
ECAT_ERR_DEVICE_COE_TIMEOUT	-2412
ECAT_ERR_DEVICE_COE_ERROR	-2413
ECAT_ERR_DEVICE_CIA402_NOT_EXIST	-2500
ECAT_ERR_DEVICE_CIA402_ADD_FAIL	-2501
ECAT_ERR_DEVICE_CIA402_TYPE_MISMATCH	-2502
ECAT_ERR_DEVICE_CIA402_NO_MODE_SUPPORT	-2503
ECAT_ERR_DEVICE_CIA402_WRONG_MODE	-2504
ECAT_ERR_DEVICE_CIA402_MODE_NOT_SUPPORT	-2505
ECAT_ERR_DEVICE_CIA402_CHANGE_WRONG_STATE	-2506
ECAT_ERR_DEVICE_CIA402_WRITE_OBJECT_FAIL	-2507
ECAT_ERR_DEVICE_CIA402_NO_SUCH_TOUCH_PROBE	-2580
ECAT_ERR_DEVICE_CIA402_NO_SUCH_TOUCH_PROBE_SOURCE	-2581
ECAT_ERR_DEVICE_EOE_NOT_SUPPORT	-2600
ECAT_ERR_DEVICE_EOE_NO_SUCH_PORT	-2601
ECAT_ERR_DEVICE_EOE_TOO_MUCH_CONTENT	-2602
ECAT_ERR_DEVICE_EOE_BUSY	-2610
ECAT_ERR_DEVICE_EOE_REQUEST_FAIL	-2611
ECAT_ERR_DEVICE_EOE_TIMEOUT	-2612
ECAT_ERR_GROUP_WRONG_INPUT	-3000
ECAT_ERR_GROUP_NOT_ATTACH	-3001

A.2　錯誤說明與修正措施

0: ECAT_SUCCESS

說明

函式呼叫成功。

修正建議

無需任何操作。

-100: ECAT_ERR_MODULE_INIT_FAIL

說明

EtherCAT 韌體初始化錯誤。

修正建議

請聯絡製造商。

-101: ECAT_ERR_MODULE_GET_VERSION_FAIL

說明

取得 EtherCAT 韌體版本的指令發生錯誤。

修正建議

請聯絡製造商。

-102: ECAT_ERR_MODULE_VERSION_MISMATCH

說明

EtherCAT 韌體版本與 EtherCAT 函式庫版本不相符。

修正建議

請更新 EtherCAT 韌體。若問題仍然存在，請聯絡製造商。

-103: ECAT_ERR_MODULE_GENERIC_TRANSFER_INIT_FAIL

說明

雙系統之間的通用傳輸介面初始化失敗。

修正建議

請聯絡製造商。

-200: ECAT_ERR_MASTER_DOWNLOAD_SETTINGS_FAIL

說明

設定 EtherCAT 主站組態的指令發生錯誤。

修正建議

請聯絡製造商。

-201: ECAT_ERR_MASTER_SET_DEVICE_SETTINGS_FAIL

說明

設定 EtherCAT 從站組態的指令發生錯誤。

修正建議

請聯絡製造商。

-202: ECAT_ERR_MASTER_GET_GROUP_INFO_FAIL

說明

取得 EtherCAT 群組組態的指令發生錯誤。

修正建議

請聯絡製造商。

-203: ECAT_ERR_MASTER_GET_MASTER_INFO_FAIL

說明

取得 EtherCAT 主站組態的指令發生錯誤。

修正建議

請聯絡製造商。

-204: ECAT_ERR_MASTER_GET_DEVICE_INFO_FAIL

說明

取得 EtherCAT 從站組態的指令發生錯誤。

修正建議

請聯絡製造商。

-205: ECAT_ERR_MASTER_SET_GROUP_SETTINGS_FAIL

說明

設定 EtherCAT 群組組態的指令發生錯誤。

修正建議

請聯絡製造商。

-206: ECAT_ERR_MASTER_MAPPING_INIT_FAIL

說明

PDO 映射（Mapping）記憶體分配失敗。

修正建議

請聯絡製造商。

-207: ECAT_ERR_MASTER_INTERRUPT_INIT_FAIL

說明

中斷功能初始化失敗。

修正建議

請聯絡製造商。

-208: ECAT_ERR_MASTER_ACTIVE_FAIL

說明

啟動 EtherCAT 主站的指令執行失敗。

修正建議

請聯絡製造商。

-209: ECAT_ERR_MASTER_ENI_INITCMDS_FAIL

說明

執行 ENI 檔案中的 SDO Download 指令失敗。

修正建議

請確認 ENI 檔案內容的正確性，並確保所有從站均正常運作。

-210: ECAT_ERR_MASTER_NO_DEVICE

說明

EtherCAT 網路中未偵測到任何從站。

修正建議

請至少連接一個 EtherCAT 從站。

-300: ECAT_ERR_MASTER_ACYCLIC_INIT_FAIL

說明

雙系統之間的非週期性（Acyclic）傳輸介面初始化失敗。

修正建議

請聯絡製造商。

-301: ECAT_ERR_MASTER_ACYCLIC_REQUEST_FAIL

說明

非週期性（Acyclic）傳輸請求失敗。

修正建議

請稍後再試。

-302: ECAT_ERR_MASTER_ACYCLIC_BUSY

說明

非週期性（Acyclic）傳輸正忙碌中。

修正建議

請稍後再試。

-303: ECAT_ERR_MASTER_ACYCLIC_TIMEOUT

說明

非週期性（Acyclic）傳輸逾時。

修正建議

請稍後再試，或確認從站是否正常運作。

-304: ECAT_ERR_MASTER_ACYCLIC_ERROR

說明

非週期性（Acyclic）傳輸過程中發生錯誤。

修正建議

請檢查錯誤碼。例如在 CoE 通訊中，可檢查 Abort Code。

-405: ECAT_ERR_MASTER_ACYCLIC_WRONG_STATUS

說明

非週期性（Acyclic）傳輸取得了無效的狀態。

修正建議

請聯絡製造商。

-400: ECAT_ERR_MASTER_GENERIC_SEND_FAIL

說明

通用傳輸（Generic Transmission）過程中資料傳送失敗。

修正建議

請聯絡製造商。

-401: ECAT_ERR_MASTER_GENERIC_RECV_FAIL

說明

通用傳輸（Generic Transmission）過程中資料接收失敗。

修正建議

請聯絡製造商。

-1000: ECAT_ERR_MASTER_NOT_BEGIN

說明

EtherCAT 主站尚未初始化。

修正建議

請呼叫 `EthercatMaster::begin()`。

-1001: ECAT_ERR_MASTER_WRONG_BUFFER_SIZE

說明

輸入的緩衝區大小不正確。

修正建議

請輸入正確大小的緩衝區。

-1002: ECAT_ERR_MASTER_REDUNDANCY_NO_DC

說明

線路冗餘（Cable Redundancy）模式不支援 DC（分散式時鐘）。

修正建議

請勿呼叫此函式，或避免使用線路冗餘模式。

-1003: ECAT_ERR_MASTER_MEMORY_ALLOCATION_FAIL

說明

記憶體分配失敗。

修正建議

請聯絡製造商。

-1004: ECAT_ERR_MASTER_OSLAYER_INIT_FAIL

說明

作業系統層（OS Layer）初始化失敗。

修正建議

請聯絡製造商。

-1005: ECAT_ERR_MASTER_NIC_INIT_FAIL

說明

網路控制器初始化失敗。

修正建議

請確認網路控制器是否存在，或聯絡製造商。

-1006: ECAT_ERR_MASTER_BASE_INIT_FAIL

說明

EtherCAT 主站指令介面的初始化失敗。

修正建議

請聯絡製造商。

-1007: ECAT_ERR_MASTER_CIA402_INIT_FAIL

說明

EtherCAT 主站 CiA 402 架構初始化失敗。

修正建議

請聯絡製造商。

-1008: ECAT_ERR_MASTER_SETUP_PDO_FAIL

說明

透過 SDO Download 設定從站的 PDO 映射時失敗。

修正建議

請確認所有從站均正常運作。若問題仍然存在，請聯絡製造商。

-1009: ECAT_ERR_MASTER_SCAN_NETWORK_FAIL

說明

掃描 EtherCAT 網路失敗。

修正建議

請正確連接網路線，確保 EtherCAT 從站已通電，或確認網路控制器是否存在。

-1010: ECAT_ERR_MASTER_START_MASTER_FAIL

說明

啟動 EtherCAT 主站失敗。

修正建議

請確認所有從站的 PDO 映射設定正確。若問題仍然存在，請聯絡製造商。

-1011: ECAT_ERR_MASTER_CYCLETIME_TOO_SMALL

說明

輸入的週期時間過短。

修正建議

請嘗試增加週期時間。

-1012: ECAT_ERR_MASTER_DUMP_OUTPUT_PDO_FAIL

說明

透過 SDO Upload 更新輸出流程資料（Output Process Data）失敗。

修正建議

請確認所有從站均正常運作，並檢查 PDO 映射的設定是否正確。

-1013: ECAT_ERR_MASTER_CONFIG_DEVICE_FAIL

說明

EtherCAT 從站初始化失敗。

修正建議

請聯絡製造商。

-1014: ECAT_ERR_MASTER_CONFIG_MAPPING_FAIL

說明

建立 PDO 映射失敗。

修正建議

請聯絡製造商。

-1015: ECAT_ERR_MASTER_WAIT_BUS_SYNC_TIMEOUT

說明

所有從站的總線同步等待逾時。

修正建議

請聯絡製造商。

-1016: ECAT_ERR_MASTER_WAIT_MASTER_SYNC_TIMEOUT

說明

主站與從站之間的同步等待逾時。

修正建議

請聯絡製造商。

-1017: ECAT_ERR_MASTER_CYCLIC_START_FAIL

說明

啟動週期性傳輸失敗。

修正建議

請聯絡製造商。

-1018: ECAT_ERR_MASTER_WRONG_BUFFER_POINTER

說明

資料緩衝區指標不正確。

修正建議

請輸入正確的資料緩衝區指標。

-1050: ECAT_ERR_MASTER_ENI_INIT_FAIL

說明

ENI 初始化失敗。

修正建議

請確認指定的 ENI 檔案是否存在。若問題仍然存在，請聯絡製造商。

-1051: ECAT_ERR_MASTER_ENI_MISMATCH

說明

ENI 檔案中的從站資訊與實際掃描到的 EtherCAT 網路從站不一致。

修正建議

請調整 EtherCAT 網路中從站的順序，或重新設定從站的識別資訊。

-1100: ECAT_ERR_MASTER_STOPPED

說明

EtherCAT 主站尚未啟動。

修正建議

請呼叫 `EthercatMaster::start()`，或避免呼叫此函式。

-1101: ECAT_ERR_MASTER_STARTED

說明

EtherCAT 主站已經啟動。

修正建議

請呼叫 `EthercatMaster::stop()`，或避免呼叫此函式。

-1102: ECAT_ERR_MASTER_NOT_IN_PREOP

說明

EtherCAT 主站不在 PRE-OP 狀態。

修正建議

請確認所有從站皆處於 PRE-OP 狀態後再呼叫此函式。

-1103: ECAT_ERR_MASTER_NOT_IN_SAFEOP

說明

EtherCAT 主站不在 SAFE-OP 狀態。

修正建議

請確認所有從站皆處於 SAFE-OP 狀態後再呼叫此函式。

-1104: ECAT_ERR_MASTER_NOT_IN_OP

說明

EtherCAT 主站不在 OP 狀態。

修正建議

請確認所有從站皆處於 OP 狀態後再呼叫此函式。

-1200: ECAT_ERR_MASTER_II_TRANSITION_FAIL

說明

在 EtherCAT 主站初始化期間，將所有從站切換至 INIT 狀態失敗。

修正建議

請重新啟動所有從站電源，然後再次執行 EtherCAT 主站程式。

-1201: ECAT_ERR_MASTER_IP_TRANSITION_FAIL

說明

EtherCAT 狀態從 INIT 切換至 PRE-OP 失敗。

修正建議

請確認所有從站運作正常，或檢查網路連線品質。

-1202: ECAT_ERR_MASTER_PS_TRANSITION_FAIL

說明

EtherCAT 狀態從 PRE-OP 切換至 SAFE-OP 失敗。

修正建議

請檢查所有從站的 PDO 映射設定是否正確。若啟用了 DC 同步，請嘗試調整與 DC 同步相關的參數。

-1203: ECAT_ERR_MASTER_SO_TRANSITION_FAIL

說明

EtherCAT 狀態從 SAFE-OP 切換至 OP 失敗。

修正建議

請嘗試調整與 DC 同步相關的參數後重新測試。

-2000: ECAT_ERR_DEVICE_NOT_EXIST

說明

指定的從站不存在。

修正建議

請避免存取該指定的從站。

-2001: ECAT_ERR_DEVICE_NOT_ATTACH

說明

該從站的物件尚未初始化。

修正建議

請呼叫 `attach()` 以初始化該從站物件。

-2002: ECAT_ERR_DEVICE_NO_MAILBOX

說明

該從站不支援 Mailbox 功能。

修正建議

請勿對該從站呼叫與 Mailbox 相關的函式。

-2003: ECAT_ERR_DEVICE_NO_DC

說明

該從站不支援 DC 同步功能。

修正建議

請勿對該從站呼叫與 DC 同步相關的函式。

-2004: ECAT_ERR_DEVICE_WRONG_INPUT

說明

輸入參數錯誤。

修正建議

請輸入正確的參數。

-2005: ECAT_ERR_DEVICE_MEMORY_ALLOCATION_FAIL

說明

為從站物件分配記憶體失敗。

修正建議

請聯絡製造商。

-2006: ECAT_ERR_DEVICE_VENDOR_ID_MISMATCH

說明

從站的 Vendor ID 與從站物件的 Vendor ID 不一致。

修正建議

請使用正確的從站物件。

-2007: ECAT_ERR_DEVICE_PRODUCT_CODE_MISMATCH

說明

從站的產品代碼（Product Code）與從站物件的不一致。

修正建議

請使用正確的從站物件。

-2008: ECAT_ERR_DEVICE_NO_SUCH_FUNCTION

說明

該從站不支援此功能。

修正建議

請勿呼叫此函式。

-2009: ECAT_ERR_DEVICE_FUNCTION_NOT_INIT

說明

該從站的特定功能尚未初始化。

修正建議

請先初始化該功能。

-2010: ECAT_ERR_DEVICE_BUSY

說明

從站正忙碌中。

修正建議

請稍後再試。

-2011: ECAT_ERR_DEVICE_TIMEOUT

說明

該從站發生逾時錯誤。

修正建議

請稍後再試，或確認該從站是否正常運作。

-2012: ECAT_ERR_DEVICE_NO_DATA

說明

該從站沒有可用資料。

修正建議

請稍後再試。

-2100: ECAT_ERR_DEVICE_SII_READ_FAIL

說明

讀取從站的 SII EEPROM 失敗。

修正建議

請確保在該從站處於 EtherCAT 的 INIT 或 PRE-OP 狀態時呼叫此函式。

-2101: ECAT_ERR_DEVICE_SII_WRITE_FAIL

說明

寫入從站的 SII EEPROM 失敗。

修正建議

請確保在從站處於 EtherCAT 的 INIT 或 PRE-OP 狀態時呼叫此函式。

-2200: ECAT_ERR_DEVICE_PDO_NOT_EXIST

說明

該從站沒有輸出流程資料（Output Process Data）。

修正建議

請勿呼叫此函式。

-2201: ECAT_ERR_DEVICE_PDO_OUT_OF_RANGE

說明

嘗試存取超出該從站流程資料範圍的資料。

修正建議

請存取從站正確範圍內的流程資料。

-2300: ECAT_ERR_DEVICE_FOE_NOT_SUPPORT

說明

該從站不支援 FoE 功能。

修正建議

請勿對該從站呼叫與 FoE 相關的函式。

-2310: ECAT_ERR_DEVICE_FOE_REQUEST_FAIL

說明

請求 FoE 通訊失敗。

修正建議

請稍後再試。

-2311: ECAT_ERR_DEVICE_FOE_TIMEOUT

說明

FoE 通訊逾時。

修正建議

請確認該從站支援 FoE 功能並且運作正常。

-2312: ECAT_ERR_DEVICE_FOE_ERROR

說明

FoE 通訊發生錯誤。

修正建議

請確認該從站支援 FoE 功能並且運作正常。

-2313: ECAT_ERR_DEVICE_FOE_BUFFER_TOO_SMALL

說明

FoE 通訊所使用的輸入緩衝區大小過小。

修正建議

請輸入合適大小的緩衝區。

-2314: ECAT_ERR_DEVICE_FOE_READ_FAIL

說明

透過 FoE 通訊讀取檔案失敗。

修正建議

請確認該從站支援透過 FoE 通訊讀取檔案的功能。

-2315: ECAT_ERR_DEVICE_FOE_WRITE_FAIL

說明

透過 FoE 通訊寫入檔案失敗。

修正建議

請確認該從站支援透過 FoE 通訊寫入檔案的功能。

-2400: ECAT_ERR_DEVICE_COE_SDO_NOT_SUPPORT

說明

該從站不支援 CoE 通訊中的 SDO 指令。

修正建議

請勿呼叫與 CoE 通訊中的 SDO 指令相關的函式。

-2401: ECAT_ERR_DEVICE_COE_SDO_INFO_NOT_SUPPORT

說明

該從站不支援 CoE 通訊中的 SDO 資訊指令。

修正建議

請勿呼叫與 CoE 通訊中 SDO 資訊指令相關的函式。

-2410: ECAT_ERR_DEVICE_COE_BUSY

說明

CoE 通訊正忙碌中。

修正建議

請稍後再試。

-2411: ECAT_ERR_DEVICE_COE_REQUEST_FAIL

說明

請求 CoE 通訊失敗。

修正建議

請稍後再試。

-2412: ECAT_ERR_DEVICE_COE_TIMEOUT

說明

CoE 通訊逾時。

修正建議

請稍後再試，或確認該從站是否正常運作。

-2413: ECAT_ERR_DEVICE_COE_ERROR

說明

CoE 通訊發生錯誤。

修正建議

請檢查 CoE 通訊的中止碼（Abort Code）。

-2500: ECAT_ERR_DEVICE_CIA402_NOT_EXIST

說明

該從站不包含與 CiA 402 有關的物件。

修正建議

請勿對該從站呼叫與 CiA 402 相關的函式。

-2501: ECAT_ERR_DEVICE_CIA402_ADD_FAIL

說明

無法將 CiA 402 從站物件插入至 EtherCAT 主站的 CiA 402 架構中。

修正建議

請確認 CiA 402 從站物件是否已成功插入至 EtherCAT 主站的 CiA 402 架構中。

-2502: ECAT_ERR_DEVICE_CIA402_TYPE_MISMATCH

說明

該從站的 Device Type（索引 1000h）物件內容並非 CiA 402 類型。

修正建議

請勿對該從站呼叫與 CiA 402 相關的函式。

-2503: ECAT_ERR_DEVICE_CIA402_NO_MODE_SUPPORT

說明

該 CiA 402 從站不支援任何 CiA 402 所定義的運作模式。

修正建議

請勿對該從站呼叫與 CiA 402 相關的函式，並確認該從站確實支援 CiA 402。

-2504: ECAT_ERR_DEVICE_CIA402_WRONG_MODE

說明

目前該從站的 CiA 402 運作模式下無法呼叫此函式。

修正建議

請先將該從站的 CiA 402 運作模式切換至正確模式後，再呼叫此函式。

-2505: ECAT_ERR_DEVICE_CIA402_MODE_NOT_SUPPORT

說明

該從站不支援指定的 CiA 402 運作模式。

修正建議

請勿將該從站的 CiA 402 運作模式切換至不支援的模式。

-2506: ECAT_ERR_DEVICE_CIA402_CHANGE_WRONG_STATE

說明

目前的 CiA 402 狀態不允許切換至指定的 CiA 402 狀態。

修正建議

請檢查當前的 CiA 402 狀態。若處於 FAULT 狀態，請先切換至 SWITCH_ON_DISABLED 狀態，再切換至目標狀態。

-2507: ECAT_ERR_DEVICE_CIA402_WRITE_OBJECT_FAIL

說明

在週期性回呼函式（cyclic callback functions）中，禁止透過 SDO Upload 或 SDO Download 存取 CiA 402 物件。

修正建議

請避免在週期性回呼函式中使用 SDO Upload/Download 存取 CiA 402 物件。如有需要，請將該 CiA 402 物件對應至流程資料（Process Data）。

-2580: ECAT_ERR_DEVICE_CIA402_NO_SUCH_TOUCH_PROBE

說明

輸入的 Touch Probe 編號不正確。

修正建議

請輸入正確的 Touch Probe 編號，範圍為 0 到 1。

-2581: ECAT_ERR_DEVICE_CIA402_NO_SUCH_TOUCH_PROBE_SOURCE

說明

輸入的 Touch Probe 訊號來源不正確。

修正建議

請輸入正確的訊號來源，範圍為 0 到 2。

-2600: ECAT_ERR_DEVICE_EOE_NOT_SUPPORT

說明

該從站不支援 EoE 功能。

修正建議

請勿對該從站呼叫與 EoE 相關的函式。

-2601: ECAT_ERR_DEVICE_EOE_NO_SUCH_PORT

說明

EoE 埠號不正確。

修正建議

請輸入正確的 EoE 埠號。

-2602: ECAT_ERR_DEVICE_EOE_TOO_MUCH_CONTENT

說明

輸入的內容過多。

修正建議

請提供適當大小的內容。

-2610: ECAT_ERR_DEVICE_EOE_BUSY

說明

EoE 通訊正忙碌中。

修正建議

請稍後再試。

-2611: ECAT_ERR_DEVICE_EOE_REQUEST_FAIL

說明

請求 EoE 通訊失敗。

修正建議

請稍後再試。

-2612: ECAT_ERR_DEVICE_EOE_TIMEOUT

說明

EoE 通訊逾時。

修正建議

請確認該從站支援 EoE 且運作正常。

-3000: ECAT_ERR_GROUP_WRONG_INPUT

說明

輸入參數不正確。

修正建議

請輸入正確的參數。

-3001: ECAT_ERR_GROUP_NOT_ATTACH

說明

該群組的物件尚未初始化。

修正建議

請初始化該群組物件。

A.3 Error Callback Code

Error Callback 代碼清單：

名稱	說明	代碼值
ECAT_ERR_WKC_SINGLE_FAULT	工作計數器單一錯誤	2000001
ECAT_ERR_WKC_MULTIPLE_FAULTS	工作計數器多重錯誤	2000002
ECAT_ERR_SINGLE_LOST_FRAME	單一封包遺失	2000003
ECAT_ERR_MULTIPLE_LOST_FRAMES	多個封包遺失	2000004
ECAT_ERR_LOST_SLAVE	從站遺失	2000005
ECAT_ERR_STATE_MISMATCH	狀態不一致	2000006
ECAT_ERR_CABLE_BROKEN	網路線斷線	2000007
ECAT_ERR_WAIT_ACK_TIMEOUT	等待 ACK 回應逾時	2001000

A.4 Event Callback Code

Event Callback 代碼清單：

名稱	說明	代碼值
ECAT_EVT_STATE_CHANGED	狀態變更事件	1000001
ECAT_EVT_CABLE_RECONNECTED	網路線重新連接事件	1000002

A.5 SDO Abort Code

ETG.1000.6 中定義的 CoE SDO 中止代碼：

值	說明
0x05030000	Toggle 位元未變化
0x05040000	SDO 通訊協定逾時
0x05040001	Client/Server 命令指定無效或未知
0x05040005	記憶體不足
0x06010000	不支援對此物件的存取
0x06010001	嘗試讀取僅能寫入的物件
0x06010002	嘗試寫入僅能讀取的物件
0x06010003	子索引不可寫入，寫入存取時 SI0 必須為 0
0x06010004	不支援對可變長度物件（如 ENUM 型別）進行 SDO 完整存取
0x06010005	物件長度超過信箱大小
0x06010006	物件已對應至 RxPDO，SDO 下載被阻擋
0x06020000	物件在物件字典中不存在
0x06040041	無法將該物件對應進 PDO
0x06040042	要對應的物件數量與長度將超過 PDO 限制
0x06040043	一般參數不相容原因
0x06040047	裝置內部發生一般性不相容
0x06060000	因硬體錯誤導致存取失敗
0x06070010	資料型別不匹配，服務參數長度不符
0x06070012	資料型別不匹配，服務參數長度過長
0x06070013	資料型別不匹配，服務參數長度過短
0x06090011	子索引不存在
0x06090030	參數值超出範圍（僅限寫入存取）
0x06090031	寫入的參數值過高
0x06090032	寫入的參數值過低
0x06090036	最大值小於最小值
0x08000000	一般錯誤
0x08000020	資料無法傳輸或儲存至應用程式 NOTE：此為無法確定具體原因時的一般性中止碼。建議使用更具體的中止碼（如 0x08000021, 0x08000022）
0x08000021	因本地控制導致資料無法傳輸或儲存至應用程式 NOTE：「本地控制」指的是應用程式層級的原因，不代表 ESM 控制

0x08000022	因裝置目前的狀態導致資料無法傳輸或儲存至應用程式。 NOTE：「裝置狀態」指的是 ESM（EtherCAT State Machine）狀態。
0x08000023	物件字典動態產生失敗或物件字典不存在。

ETG.1020 中定義的擴充 CoE SDO 中止代碼：

值	說明
0x06010003	子索引不可寫入，進行寫入操作時 SI0 必須為 0
0x06010004	不支援對可變長度的物件（例如 ENUM 型別）進行 SDO 完整存取
0x06010005	物件長度超過信箱大小
0x06010006	物件已對應至 RxPDO，SDO 下載被阻擋 此選用的中止碼僅在 SafeOp 與 Op 狀態下使用
0x06090033	設定的模組清單與偵測到的模組清單不一致 當寫入物件 0xF03x，但其內容與物件 0xF05x 不符時，應使用此中止碼

A.6 資料型別

ETG.1000.6 中定義的基本資料型態：

Index (hex)	物件類型	名稱
0001	DEFTYPE	BOOLEAN
0002	DEFTYPE	INTEGER8
0003	DEFTYPE	INTEGER16
0004	DEFTYPE	INTEGER32
0005	DEFTYPE	UNSIGNED8
0006	DEFTYPE	UNSIGNED16
0007	DEFTYPE	UNSIGNED32
0008	DEFTYPE	REAL32
0009	DEFTYPE	VISIBLE_STRING
000A	DEFTYPE	OCTET_STRING
000B	DEFTYPE	UNICODE_STRING
000C	DEFTYPE	TIME_OF_DAY
000D	DEFTYPE	TIME_DIFFERENCE
000F	DEFTYPE	DOMAIN
0010	DEFTYPE	INTEGER24
0011	DEFTYPE	REAL64
0012	DEFTYPE	INTEGER40
0013	DEFTYPE	INTEGER48
0014	DEFTYPE	INTEGER56
0015	DEFTYPE	INTEGER64
0016	DEFTYPE	UNSIGNED24
0018	DEFTYPE	UNSIGNED40
0019	DEFTYPE	UNSIGNED48
001A	DEFTYPE	UNSIGNED56
001B	DEFTYPE	UNSIGNED64
001D	DEFTYPE	GUID
001E	DEFTYPE	BYTE
002D	DEFTYPE	BITARR8
002E	DEFTYPE	BITARR16
002F	DEFTYPE	BITARR32

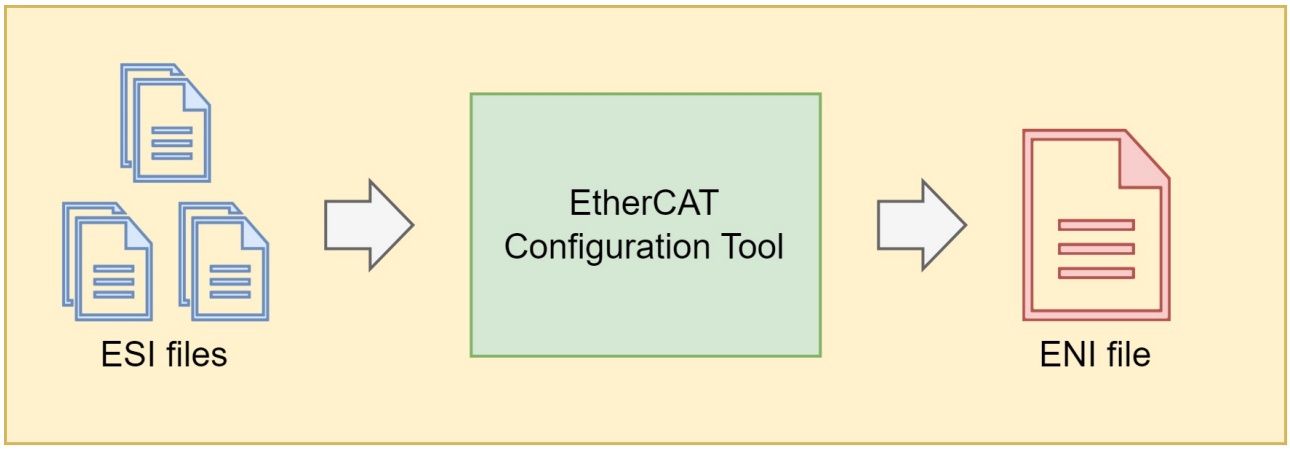
ETG.1000.6 中定義的擴展資料類型：

Index (hex)	物件類型	名稱
0021	DEFSTRUCT	PDO_MAPPING
0023	DEFSTRUCT	IDENTITY
0025	DEFSTRUCT	COMMAND_PAR
0029	DEFSTRUCT	SYNC_PAR
0030	DEFTYPE	BIT1
0031	DEFTYPE	BIT2
0032	DEFTYPE	BIT3
0033	DEFTYPE	BIT4
0034	DEFTYPE	BIT5
0035	DEFTYPE	BIT6
0036	DEFTYPE	BIT7
0037	DEFTYPE	BIT8
0040-005F	DEFSTRUCT	Manufacturer Specific Complex Data Types
0060-007F	DEFTYPE	Device Profile 0 Specific Standard Data Types
0080-009F	DEFSTRUCT	Device Profile 0 Specific Complex Data Types
00A0-00BF	DEFTYPE	Device Profile 1 Specific Standard Data Types
00C0-00DF	DEFSTRUCT	Device Profile 1 Specific Complex Data Types
00E0-00FF	DEFTYPE	Device Profile 2 Specific Standard Data Types
0100-011F	DEFSTRUCT	Device Profile 2 Specific Complex Data Types
0120-013F	DEFTYPE	Device Profile 3 Specific Standard Data Types
0140-015F	DEFSTRUCT	Device Profile 3 Specific Complex Data Types
0160-017F	DEFTYPE	Device Profile 4 Specific Standard Data Types
0180-019F	DEFSTRUCT	Device Profile 4 Specific Complex Data Types
01A0-01BF	DEFTYPE	Device Profile 5 Specific Standard Data Types
01C0-01DF	DEFSTRUCT	Device Profile 5 Specific Complex Data Types
01E0-01FF	DEFTYPE	Device Profile 6 Specific Standard Data Types
0200-021F	DEFSTRUCT	Device Profile 6 Specific Complex Data Types
0220-023F	DEFTYPE	Device Profile 7 Specific Standard Data Types
0240-025F	DEFSTRUCT	Device Profile 7 Specific Complex Data Types

A.7 EtherCAT Network Information

EtherCAT Network Information (ENI) 檔案包含設定 EtherCAT 網路所需的所有參數。這是一個以 XML 為基礎的檔案，內含主站的一般資訊，以及與該主站相連之所有從站的配置資料。

EtherCAT 設定工具會讀取 ESI 檔案，或透過線上掃描來偵測所有從站。接著，使用者可以設定相關的 EtherCAT 參數，例如 PDO 對應 (PDO Mapping)、啟用分散式時鐘 (DC)，並將其匯出為 ENI 檔案。



EtherCAT 技術組織 (ETG) 規範 EtherCAT 主站軟體在「網路設定」部分至少必須支援下列其中一項功能：**線上掃描 (Online Scanning) 或讀取 ENI (Reading ENI)**。然而，本函式庫同時支援這兩項功能。針對讀取 ENI 的情境，目前此函式庫僅從 ENI 檔案中擷取部分資訊來進行網路設定。

提取的資訊包含以下內容：

EtherCATConfig : Config : SubDevice : Info

- **元素**
  - VendorId
  - ProductCode
- **屬性**
  - Identification : Value
- **用途**
  用於檢查 EtherCAT 網路上的從站裝置是否與 ENI 檔案中所指定的從站裝置相符。檢查規則如下：
  - 檢查 ENI 檔案中的從站裝置數量是否與實際網路上的從站裝置數量一致。
  - 對於 ENI 檔案中具有 Identification: Value 屬性的從站裝置，檢查網路上是否有從站裝置具有相符的 Alias Address (別名地址) 與 Identification: Value、Vendor ID 及 Product Code。如果存在，則表示匹配成功。
  - 對於無法匹配的具有 Identification: Value 屬性的從站裝置，或是未設定該屬性的從站裝置，將根據其在序列中的順序比對 Vendor ID 與 Product Code 是否一致。

EtherCATConfig : Config : SubDevice : Mailbox

- **元素**
  - Send : MailboxSendInfoType : Start
  - Recv : MailboxRecvInfoType : Start
- **用途**
  用於設定 EtherCAT 裝置的 Mailbox 實體起始地址。

EtherCATConfig : Config : SubDevice : Mailbox : CoE

- **元素**
  - InitCmds : InitCmd : Index
  - InitCmds : InitCmd : SubIndex
  - InitCmds : InitCmd : Data
  - InitCmds : InitCmd : Timeout
- **屬性**
  - InitCmds : InitCmd : CompleteAccess
- **用途**
  在將 EtherCAT 狀態機切換至 Pre-Operational 狀態後，於 `EthercatMaster::begin()` 中執行從站裝置的 CoE 初始化指令。

EtherCATConfig : Config : SubDevice : ProcessData

- **元素**
  - Recv : BitLength
  - Send : BitLength
- **用途**

  提供從站裝置輸出與輸入處理資料的位元長度，以供韌體進行相關設定。

EtherCATConfig : Config : SubDevice : ProcessData : Sm

- **元素**
  - SyncManagerSettings : StartAddress
  - SyncManagerSettings : ControlByte
  - SyncManagerSettings : Enable
- **用途**

  用於設定 EtherCAT 裝置處理資料的 Sync Manager 註冊位址。

EtherCATConfig : Config : SubDevice: DC

- **元素**
  - CycleTime0
  - CycleTime1
  - ShiftTime
- **用途**

  用於設定 EtherCAT 裝置的分散式時鐘 (DC) 參數。

保固

本產品自購買日起一年內保證處於良好工作狀態。若在保固期間內產品無法正常運作，我們將依情況提供免費維修或更換服務，惟以下條款除外。本保固不適用於因誤用、改裝、意外或災害而導致損壞的產品。本公司對於因使用或誤用本產品所造成的任何損害、利潤損失、儲存資料損失，或任何其他附帶性或衍生性損害不負任何責任。本公司亦不對任何第三方提出的任何相關索賠負責。產品退貨前，須先向本公司申請退貨授權（RMA）。您可透過來電或傳真方式聯絡本公司以取得退貨授權碼。退回商品時，須附上明確的問題描述。