

使用手冊

QEC-PPC-M-XXXT

DM&P Vortex86EX2 處理器

具多功能整合能力的 EtherCAT MDevice。

9 吋 / 15 吋面板電腦，搭載四線電阻式觸控螢幕

(版本 1.2)

版本紀錄

日期	版本	說明
2025/8/26	Version1.0	首次發行。
2026/1/9	Version1.1	• 錯字修正： 第 118、126、127 與 139 頁。QEC MDevice Open-frame series => QEC-PPC-M series MDevice
2026/8/24	Version1.2	• 新增 2.3 Extension I/O 章節。 • 新增 4.12 Arduino Pins Usage 章節。

版權聲明

為持續改良產品，本手冊內容如有變更恕不另行通知。版權所有，翻印必究。製造商對本文件中可能存在的任何錯誤不負任何責任，亦不承諾更新或維持本手冊內容之即時性。

未經 ICOP Technology Inc. 事前書面同意，本手冊之任何部分均不得以任何形式或任何方式進行重製、複製、翻譯或傳播，無論全部或部分。

©Copyright 2026 ICOP Technology Inc.

Ver.1.2 2026 年 8 月

商標聲明

ICOP® 為 ICOP Corporation 之註冊商標。本文件中出現的其他品牌名稱或產品名稱，均為其各自所有人之財產與註冊商標。文中所提及之所有名稱僅供識別之用。

如需更詳細的資訊，或您對 ICOP 的其他產品有興趣，歡迎造訪我們的官方網站：

- 全球：www.icop.com.tw
- 美國：www.icoptech.com
- 日本：www.icop.co.jp
- 歐洲：www.icoptech.eu
- 中國：www.icop.com.cn

如需技術支援或下載驅動程式，請造訪我們的網站：

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

如需 EtherCAT 解決方案服務、技術支援或教學資源，以及 86Duino Coding IDE 500+ 的介紹、功能、語言、函式庫等資訊，請造訪 QEC 網站：

- QEC: https://www.qec.tw/

本手冊適用於 QEC 系列產品。

安全資訊

- 請詳閱以下安全注意事項。
- 搬運本機時請雙手持拿並小心操作。
- 電源輸入電壓為 +12 至 +50VDC (典型值 +24VDC)。
- 將設備連接至電源插座前，請先確認電源電壓是否正確。
- 為避免 QEC 設備發生觸電或火災危險，請保持乾燥，並遠離水氣與濕氣。
- 工作溫度：0°C 至 +60°C / -20°C 至 +70°C (選配)。
- 使用外接儲存裝置作為主要作業系統儲存媒體時，請務必在設備電源關閉的狀態下進行插拔。
- 在電源供應器未拔除前，切勿碰觸未絕緣的端子或線材。
- 請將 QEC 設備放置在盡可能靠近電源插座的位置以便操作，並避免手臂與周邊線材纏繞而造成拉扯。
- 若 QEC 設備將有一段時間不使用，請確實拔除電源，以免暫態過電壓造成損壞。

警告！



請勿嘗試開啟或拆解本產品機殼。如需維修，請聯絡您的經銷商，由合格技術人員進行處理。

目錄

Ch. 1 一般資訊 1
1.1 簡介 2
1.1.1 QEC EtherCAT MDevice 架構 3
1.1.2 硬體平台 4
1.1.3 雙系統同步 5
1.1.4 軟體支援 6
1.2 規格 7
1.2.1 QEC-PPC-M-090T 7
1.2.2 QEC-PPC-M-150T 9
1.3 尺寸 11
1.3.1 QEC-PPC-M-090T 11
1.3.2 QEC-PPC-M-150T 12
1.4 TFT-LCD 面板檢驗標準 13
1.5 訂購資訊 16
1.5.1 訂購料號 16
Ch. 2 硬體系統 17
2.1 一般技術資料 18
2.2 總覽 19
2.2.1 USB 20
2.2.2 EtherCAT 介面 21
2.2.3 電源狀態指示燈 23
2.2.4 電源連接器 24
2.2.5 USB Type C 25
2.2.6 Giga LAN 26
2.2.7 HD Audio 27
2.3 Extension I/O（選配） 28
2.3.1 Arduino 腳位定義 28
2.4 連接器配線 35
2.4.1 將線材接入連接器 35
2.4.2 將線材自連接器移除 35
Ch. 3 硬體安裝 36
3.1 安裝說明 37
3.1.1 一般準則 37
3.1.2 安裝尺寸 41
Ch. 4 快速上手 （本章提供多語版本） 43
4.1 包裝內容物 46
4.2 硬體配置 47
4.3 軟體 / 開發環境 48
4.4 連接至 PC 並設定開發環境 49
4.5 EtherCAT 通訊 51

4.5.1 EtherCAT State Machine (ESM) 控制 51
4.5.2 SubDevice 資訊 58
4.5.3 Process Data Objects (PDO) 函式 63
4.5.4 CANopen over EtherCAT (CoE) 功能 68
4.5.5 Cyclic Callback 函式 75
4.5.6 Distributed Clock (DC) 組態函式 81
4.5.7 86EVA：EtherCAT 組態工具 86
4.5.8 將 ENI 匯入 QEC MDevice 100
4.6 Bootloader Menu 使用說明 107
4.6.1 開啟 Bootloader Menu 108
4.6.2 General 頁面 111
4.6.3 EtherCAT 頁面 115
4.6.4 Security 頁面 118
4.6.5 Exit 頁面 124
4.7 USB 裝置使用說明 125
4.7.1 範例 1：將 .txt 儲存至 USB 隨身碟 126
4.7.2 範例 2：將 .txt 儲存至 eMMC 127
4.8 Giga LAN 設定 128
4.8.1 Ethernet 通訊 129
4.8.2 Modbus TCP 通訊 135
4.9 HMI 設計 137
4.9.1 函式庫說明 137
4.9.2 使用圖形化 HMI 編輯器：86HMI Editor 138
4.10 音訊使用說明 141
4.10.1 範例 1：簡易音訊播放器 141
4.11 疑難排解 143
4.11.1 無法順利上傳程式碼 143
4.12 Arduino 腳位使用說明 146
4.12.1 GPIO 149
4.12.2 ADC 150
4.12.3 TX/RX 151
4.12.4 RS-485 152
4.12.5 CAN 156
4.12.6 SPI 157
4.12.7 I2C 159
4.12.8 MCM 160
Ch. 5 軟體功能 162
5.1 軟體說明 163
5.2 EtherCAT 函式列表 164
5.2.1 EtherCAT MDevice 165
5.2.2 EtherCAT SubDevice 167
5.2.3 QEC 系列 SubDevice 174
5.3 延伸資源 176
附錄 177

A1. 關於 86Duino IDE 的 ENI 組態 178
保固 181

Ch. 1

一般資訊

1.1 簡介

ICOP 的 QEC-PPC-M 系列（QEC-PPC-M-090T / QEC-PPC-M-150T）是整合面板與觸控 LCD 系統的 EtherCAT MDevice，專為即時、可靠且同步的工業 HMI 控制而設計。各機種皆整合高解析度的觸控 TFT LCD（9 吋或 15 吋），在精巧的機構中提供堅固的機殼與強大的控制核心。

以 86Duino IDE 高效開發

開發環境採用 86Duino IDE，這是一套工業級的類 Arduino 平台，支援 EtherCAT API、圖形化程式設計工具與高階 C/C++ 程式語言，可加速開發、降低人才招募難度並縮短上市時間。除 EtherCAT 之外，QEC-PPC-M 系列亦支援 Modbus TCP、Ethernet TCP/IP 與 CAN bus，提供完整的工業自動化解決方案。

運動控制與 I/O 控制的即時精準度

QEC MDevice 支援 PDO、CoE、FoE、Distributed Clock (DC) 等必要的 EtherCAT 功能，可彈性整合伺服驅動器、數位 I/O 等第三方 EtherCAT 裝置。最小 cycle time 為 125 µs，jitter 小於 1 µs（透過 86Duino IDE），非常適合高同步需求的運動控制與 I/O 控制應用。

延伸閱讀：EtherCAT MDevice Benchmark

嵌入式規格下的穩固儲存、可靠 I/O 與多元連接能力

各機種皆內建 2GB SLC eMMC，確保作業系統穩定運作，並提供充裕空間存放執行檔、HMI 圖形與應用資料，檔案可透過 86Duino IDE 部署。IDE 亦整合 LVGL 圖形函式庫，開發者可直接在 QEC 裝置上打造現代化且具互動性的觸控 HMI。

除 EtherCAT 控制之外，QEC-PPC-M 系列亦可監控系統溫度、電壓與電流，為碳足跡分析與系統壽命推估提供實用數據。 堅固的面板設計（尺寸依機種而異）可彈性整合並支援工業應用的客製化需求。標準工作溫度為 0°C 至 +60°C，並提供 -20°C 至 +70°C 的寬溫選配。

每台 QEC-PPC-M 皆具備雙 EtherCAT 埠（供 redundancy 使用）、一組 Giga LAN 埠、三組 USB 埠、一組 USB 除錯埠（供上傳 / 除錯）與 HD Audio，以上皆可透過標準函式庫存取。

1.1.1 QEC EtherCAT MDevice 架構

EtherCAT MDevice 軟體主要分為兩個部分，分別運行於 Vortex86EX2 CPU 的兩個系統上，各自負責下列工作：

- **EtherCAT MDevice Library**
  - 提供 C/C++ 應用程式介面：
    - 初始化介面。
    - 組態設定介面。
    - Process Data (PDO) 存取介面。
    - CAN application protocol over EtherCAT (CoE) 存取介面。
    - File access over EtherCAT (FoE) 存取介面。
    - SubDevice Information Interface (SII) 存取介面。
    - Distributed Clocks (DC) 存取介面。
- **EtherCAT MDevice Firmware**
  - 執行 EtherCAT MDevice Core。
  - 控制 Primary/Secondary Ethernet Driver，負責發送 EtherCAT frame

這些程式設計為運行於 FreeDOS 作業系統，並使用 DJGPP 環境所提供的 GCC 編譯器進行編譯。

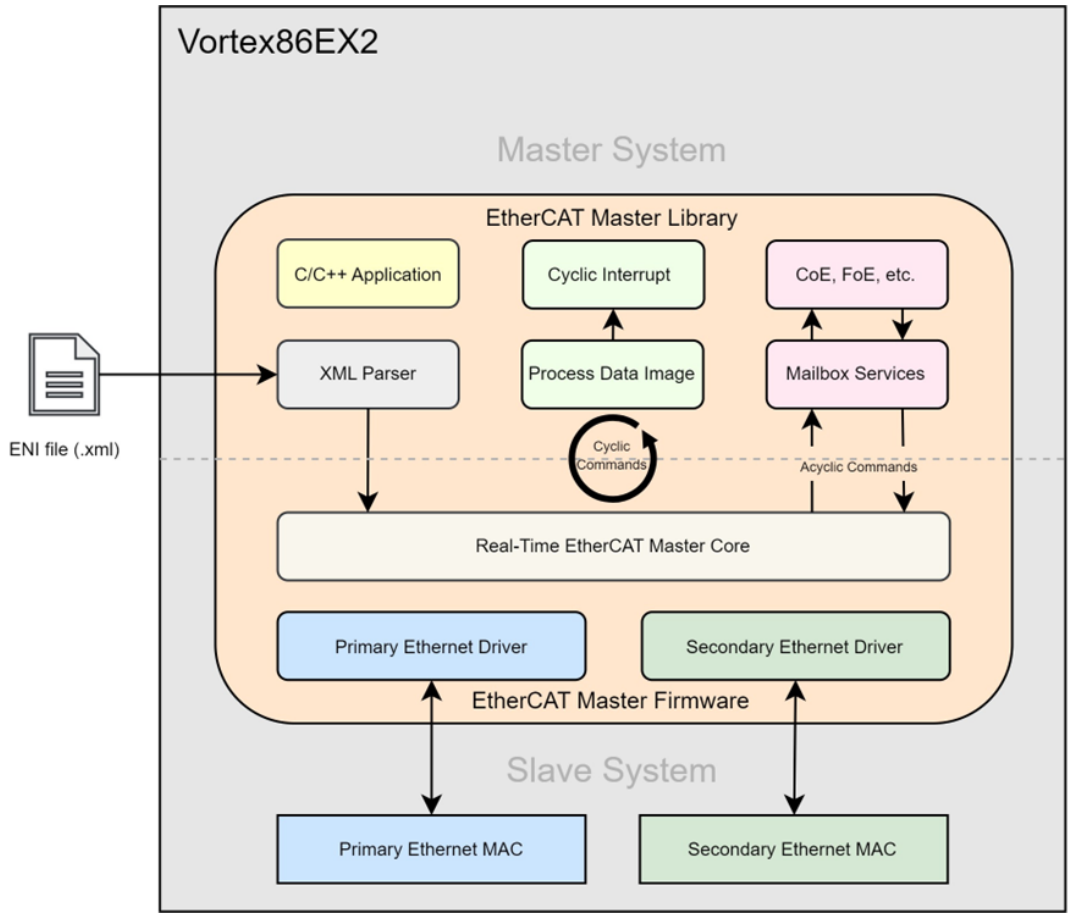
1.1.2 硬體平台

EtherCAT MDevice 軟體僅能運行於 DM&P 所生產的 Vortex86EX2 CPU，該處理器採用雙系統架構，分為 Master System 與 Slave System，各自運行獨立的作業系統，兩系統之間透過 Dual-Port RAM 與事件中斷進行通訊。

兩者各自負責的工作如下：

- **Master System**
  - 使用者的 EtherCAT 應用程式。
  - 使用者的 HMI 應用程式。
  - 使用者的 Ethernet 應用程式。
  - 以及其他應用。
- **Slave System**
  - 僅負責運行 EtherCAT MDevice Firmware。

由於大多數應用程式都運行於 Master System，運行於 Slave System 的 EtherCAT MDevice Firmware 不會受到其他應用程式干擾，因此得以專注執行 EtherCAT MDevice Core，確保 EtherCAT 的同步性與即時性。



1.1.3 雙系統同步

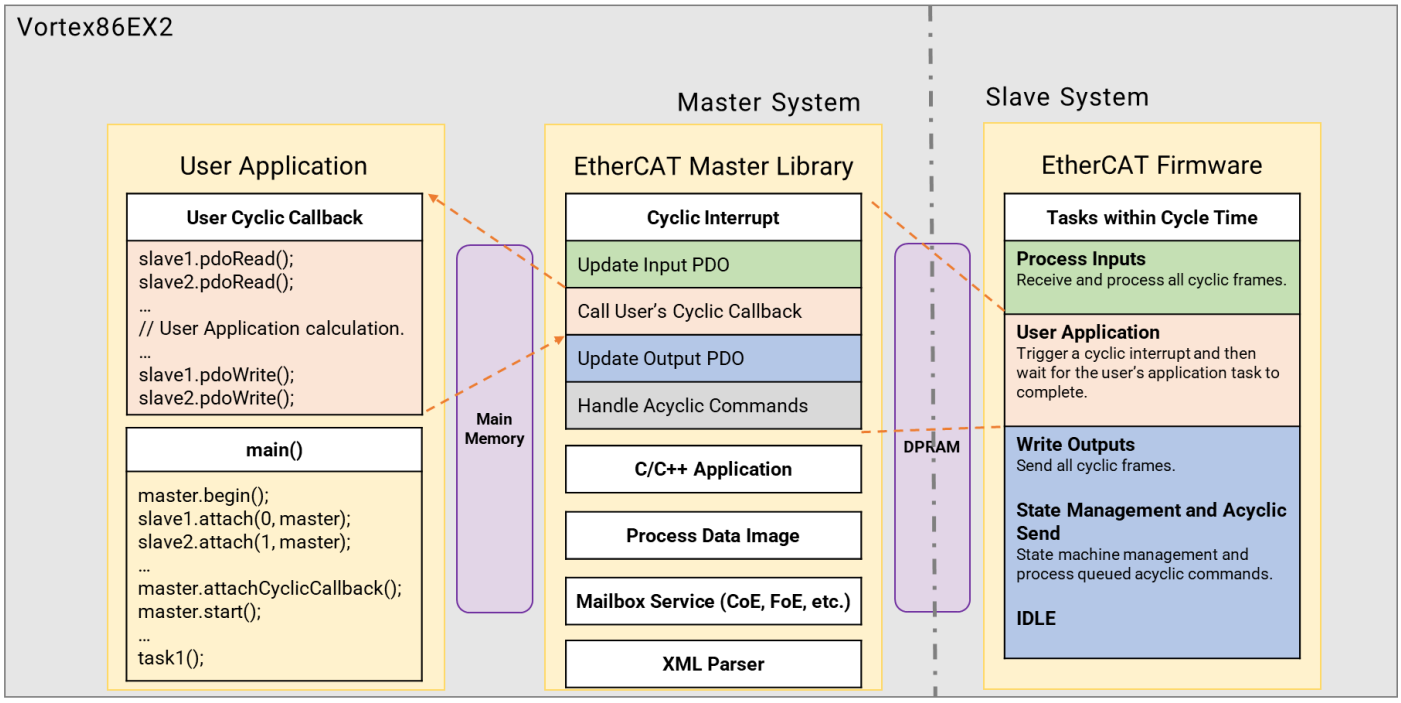
本節主要說明雙系統的 PDO 資料同步機制。如下圖所示，User Application 與 EtherCAT MDevice Library 區塊運行於 Master System，而 Real-Time EtherCAT MDevice Core 則運行於 Slave System。

當 EtherCAT MDevice Core 進入 **Process Inputs** 階段時，會從 Ethernet Driver 接收所有週期性 frame，並將 Input PDO 資料複製到 DPRAM。

進入 **User Application** 階段後，EtherCAT MDevice Core 會對 Master System 觸發 Cyclic Interrupt。Master System 收到 Cyclic Interrupt 後，隨即執行 EtherCAT MDevice Library 的中斷處理程序：將 Input PDO 資料由 DPRAM 搬移至主記憶體、呼叫使用者註冊的 Cyclic Callback、在 Cyclic Callback 結束後將 Output PDO 資料由主記憶體搬移至 DPRAM、處理非週期性命令，最後結束中斷處理程序。此時 EtherCAT MDevice Core 的 **User Application** 階段與中斷處理程序會同時完成。

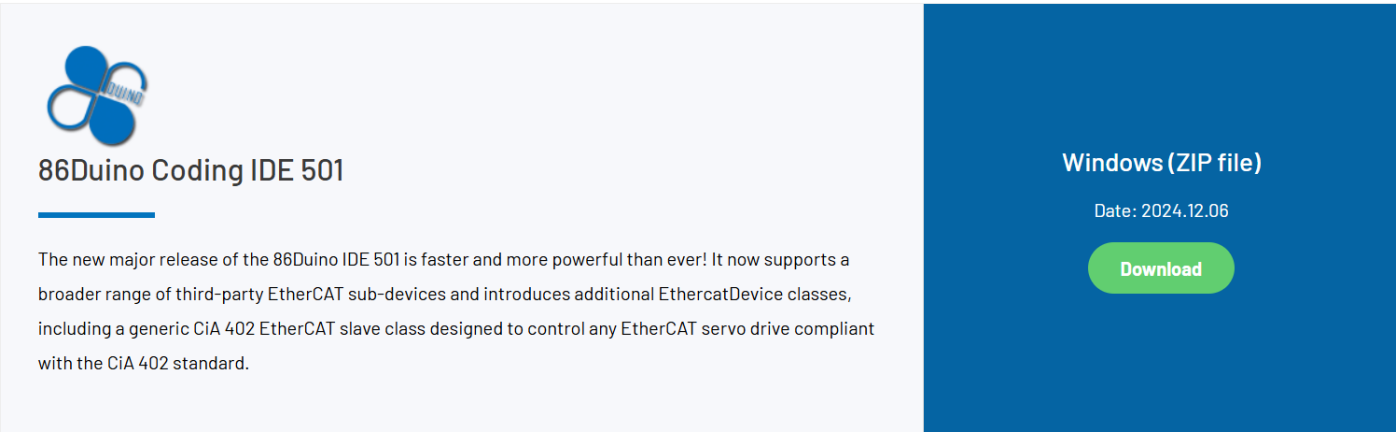
當 EtherCAT MDevice Core 進入 **Write Outputs** 階段時，會將 Output PDO 資料由 DPRAM 複製到 Ethernet Driver 的 DMA 並送出 frame。

上述工作會依照所述流程週期性地循環執行，確保雙系統 PDO 資料的同步。



1.1.4 軟體支援

86Duino 整合開發環境（IDE）讓您能輕鬆撰寫程式碼並上傳至 86Duino 開發板與 QEC MDevice，可運行於 Windows、Mac OS X 與 Linux。此開發環境以 Java 撰寫，並以 Arduino IDE、Processing、DJGPP 等開源軟體為基礎，可由 https://www.qec.tw/software/ 下載。



QEC MDevice 的軟體 86Duino IDE 另提供一套組態工具：**86EVA**，這是一款圖形化介面工具，可供使用者編輯 EtherCAT 網路的參數，功能如下：

- 掃描 EtherCAT SubDevice。
- 匯入 ENI 檔案。
- 設定 EtherCAT MDevice。
- 設定 EtherCAT SubDevice。

其他詳細功能請參閱 86EVA 使用手冊。



1.2 規格

1.2.1 QEC-PPC-M-090T

CPU 板規格

CPU	DM&P Vortex86EX2 處理器，Master 533MHz / Slave 400MHz
記憶體	板載 512MB / 1GB DDRIII
儲存裝置	32MB SPI Flash / 2GB SLC eMMC
LAN	1Gbps Ethernet RJ45 x1 10/100Mbps Ethernet RJ45 x2（供 EtherCAT 使用）
PoE（選配）	可選配於 10/100Mbps Ethernet x 1，支援 IEEE 802.3AT (Mode B)
I/O 連接器	DC 電源輸入 / 輸出連接器 x1 USB (Type-C) x1（僅供上傳 / 除錯） USB 2.0 Host x3 HD Audio x1 RJ45 x3
通訊協定	EtherCAT、Modbus、Ethernet 等
Ethernet 標準	IEEE 802.3
控制 Cycle Time	125 μs（最小值）
軟體支援	86Duino Coding IDE 501+ （此開發環境以 Java 撰寫，並以 Arduino IDE、Processing、DJGPP 等開源軟體為基礎）

*MCM 訊號等同於 Arduino 的 PWM 訊號。

機構與環境

電源連接器	4-pin 電源輸入 / 輸出與 2-pin FGND
電源需求	+12 至 +50VDC 電源輸入（典型值 +24VDC）
消耗功率	9.6 瓦
工作溫度	0 至 +60°C / -20 至 +70°C（選配）
儲存溫度	-30 ~ +85°C
工作濕度	0% ~ 90% 相對濕度，無凝結
尺寸	242 x 156.5 x 44 mm
重量	1360 g
前面板防護等級	IP65
認證	CE, FCC, VCCI
內部監控	溫度、電壓、電流、啟動時間

LCD 規格

顯示類型	9 吋 WSVGA TFT-LCD
背光模組	LED
顯示解析度	1024 (W) x 600 (H)
對比度	500: 1
顯示色彩	262,144
有效顯示區域	196.61 (W) x 114.15 (H) mm
可視角	垂直 120º，水平 140º
亮度 (cd/m^2)	300 nits
背光壽命	20,000 小時

觸控螢幕

類型	類比電阻式
解析度	連續式
穿透率	80%
控制器	PS/2 介面
耐用度	100 萬次

1.2.2 QEC-PPC-M-150T

CPU 板規格

CPU	DM&P Vortex86EX2 處理器，Master 533MHz / Slave 400MHz
記憶體	板載 512MB / 1GB DDRIII
儲存裝置	32MB SPI Flash / 2GB SLC eMMC
LAN	1Gbps Ethernet RJ45 x1 10/100Mbps Ethernet RJ45 x2（供 EtherCAT 使用）
PoE（選配）	可選配於 10/100Mbps Ethernet x 1，支援 IEEE 802.3AT (Mode B)
I/O 連接器	DC 電源輸入 / 輸出連接器 x1 USB (Type-C) x1（僅供上傳 / 除錯） USB 2.0 Host x3 HD Audio x1 RJ45 x3
通訊協定	EtherCAT、Modbus、Ethernet 等
Ethernet 標準	IEEE 802.3
控制 Cycle Time	125 μs（最小值）
軟體支援	86Duino Coding IDE 501+ （此開發環境以 Java 撰寫，並以 Arduino IDE、Processing、DJGPP 等開源軟體為基礎）

*MCM 訊號等同於 Arduino 的 PWM 訊號。

機構與環境

電源連接器	4-pin 電源輸入 / 輸出與 2-pin FGND
電源需求	+12 至 +50VDC 電源輸入（典型值 +24VDC）
消耗功率	13 瓦
工作溫度	0 至 +60°C / -20 至 +70°C（選配）
儲存溫度	-30 ~ +85°C
工作濕度	0% ~ 90% 相對濕度，無凝結
尺寸	354.81 x 280.84 x 55 mm
重量	4000 g
前面板防護等級	IP65
認證	CE, FCC, VCCI

LCD 規格

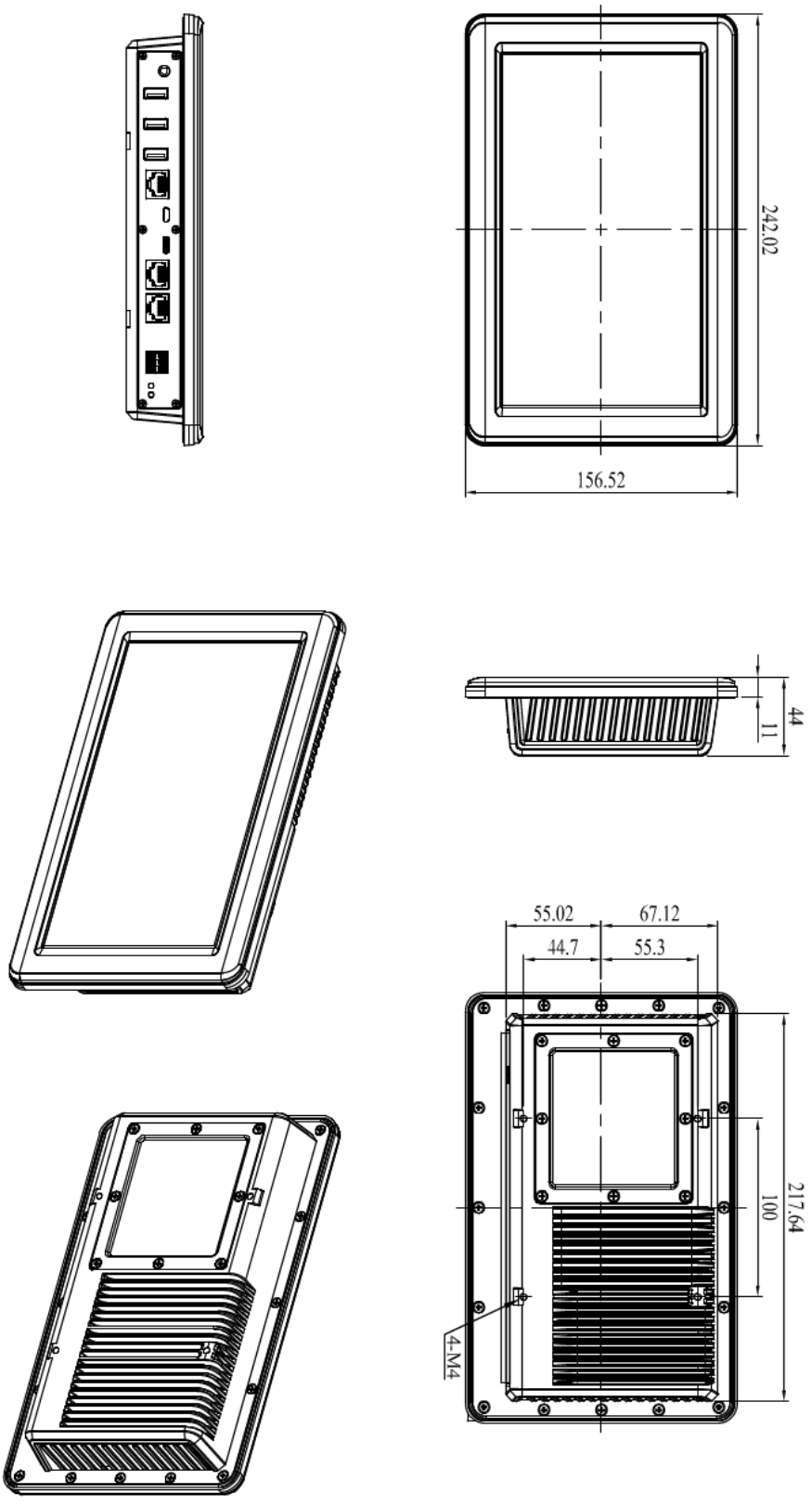
顯示類型	15 吋 XGA TFT LCD
背光模組	LED
像素	1024 (H) x 768 (V)
像素間距 (mm)	0.297 x 0.297 mm
對比度	600: 1
亮度 (cd/m^2)	350 nits
支援色彩	262,144
可視角	垂直 160°，水平 160°
背光壽命	30,000 小時

觸控螢幕

類型	類比電阻式
解析度	連續式
穿透率	80%
控制器	PS/2 介面
耐用度	100 萬次

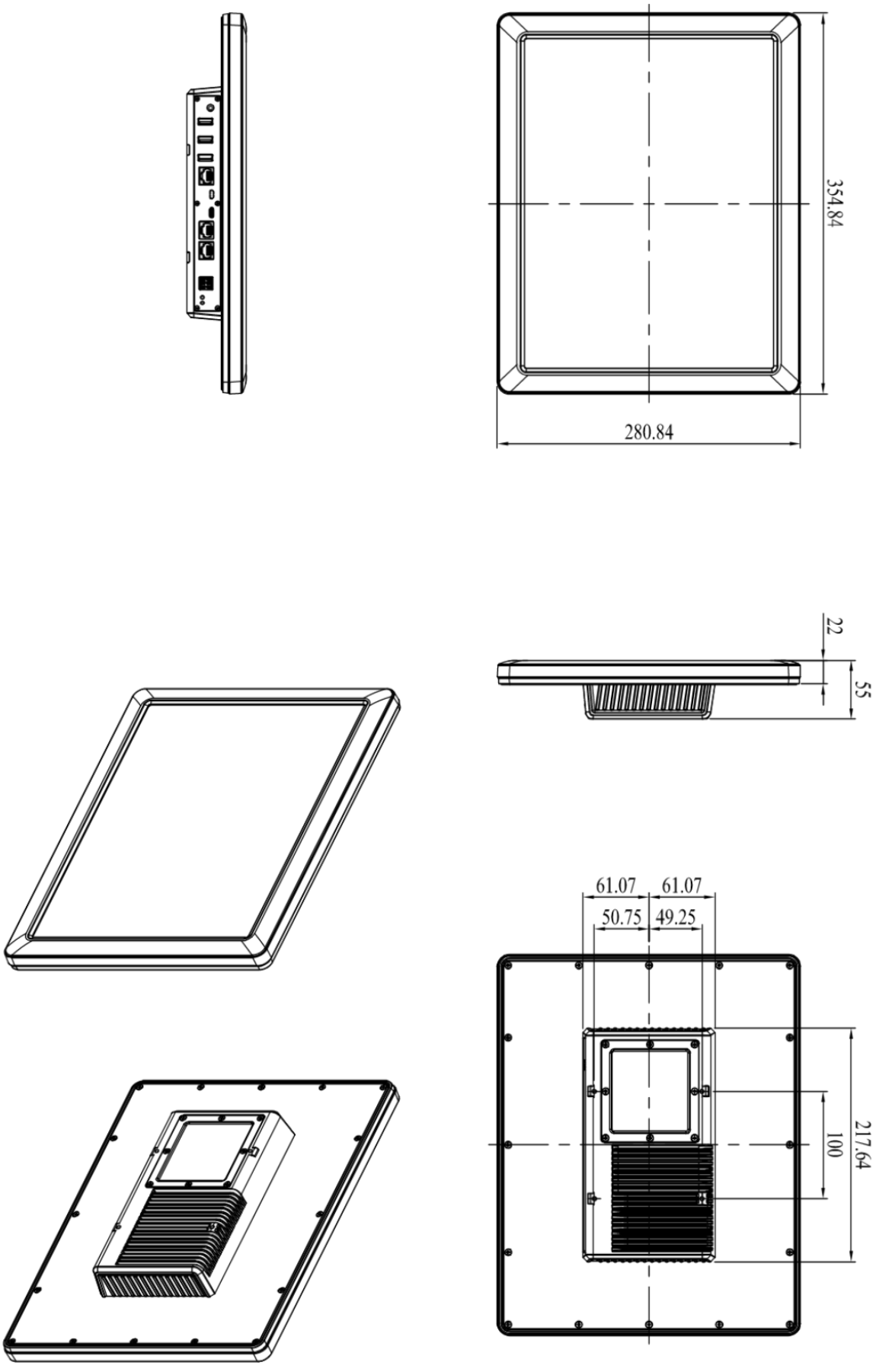
1.3 尺寸

1.3.1 QEC-PPC-M-090T



（單位：mm）

1.3.2 QEC-PPC-M-150T



（單位：mm）

1.4 TFT-LCD 面板檢驗標準

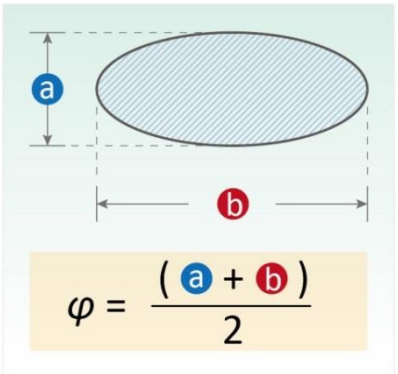
<table>
<tr><th colspan="3">缺陷類型</th><th colspan="6">允收標準</th><th>備註</th></tr>
<tr><td rowspan="10">外觀缺陷</td><td rowspan="10">內部</td><td rowspan="3">點狀缺陷</td><td colspan="4">φ＜0.15mm</td><td colspan="2">忽略不計</td><td rowspan="3">Note1</td></tr>
<tr><td colspan="4">0.15mm≦φ≦0.5mm</td><td colspan="2">N≦4</td></tr>
<tr><td colspan="4">0.5mm＜φ</td><td colspan="2">N=0</td></tr>
<tr><td rowspan="2">纖維狀缺陷</td><td colspan="4">0.03mm<W≦0.1mm, L≦5mm</td><td colspan="2">N≦3</td><td rowspan="2">Note1</td></tr>
<tr><td colspan="4">1.0mm＜W, 1.5mm＜L</td><td colspan="2">N=0</td></tr>
<tr><td rowspan="3">偏光板氣泡</td><td colspan="4">φ＜0.15mm</td><td colspan="2">忽略不計</td><td rowspan="3">Note1</td></tr>
<tr><td colspan="4">0.15mm≦φ≦0.5mm</td><td colspan="2">N≦2</td></tr>
<tr><td colspan="4">0.5mm＜φ</td><td colspan="2">N=0</td></tr>
<tr><td>Mura（亮度不均）</td><td colspan="6">透過 6% ND 濾片檢視時，輕微 mura 為允收範圍。</td><td></td></tr>
<tr><td rowspan="8">電性缺陷</td><td rowspan="2" colspan="2">亮點</td><td colspan="3">A 級</td><td colspan="3">B 級</td><td></td></tr>
<tr><td>C 區</td><td>O 區</td><td>合計</td><td>C 區</td><td>O 區</td><td>合計</td><td>Note3</td></tr>
<tr><td colspan="2"></td><td>N≦0</td><td>N≦2</td><td>N≦2</td><td>N≦2</td><td>N≦3</td><td>N≦5</td><td>Note2</td></tr>
<tr><td colspan="2">暗點</td><td>N≦2</td><td>N≦3</td><td>N≦3</td><td>N≦3</td><td>N≦5</td><td>N≦8</td><td></td></tr>
<tr><td colspan="2">總點數</td><td colspan="3">N≦4</td><td>N≦5</td><td>N≦6</td><td>N≦8</td><td>Note2</td></tr>
<tr><td colspan="2">兩相鄰點</td><td>N≦0</td><td>N≦1 對</td><td>N≦1 對</td><td>N≦1 對</td><td>N≦1 對</td><td>N≦1 對</td><td>Note4</td></tr>
<tr><td colspan="2">三點以上相鄰點</td><td colspan="6">不允收</td><td></td></tr>
<tr><td colspan="2">線狀缺陷</td><td colspan="6">不允收</td><td></td></tr>
</table>

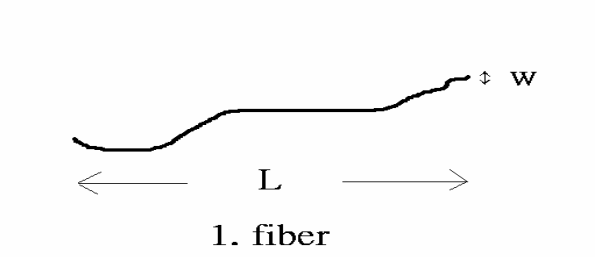
(1) 一個 pixel 由 R、G、B 三個 sub-pixel 組成。(Sub-pixel = Dot)

(2) 在 6% ND 濾片下，輕微亮點為允收範圍。

(3) 若需 G0 等級（總點數 N = 0），請洽詢您所屬區域的業務代表。

[註 1] W：寬度 [mm]；L：長度 [mm]；N：數量；φ：平均直徑。

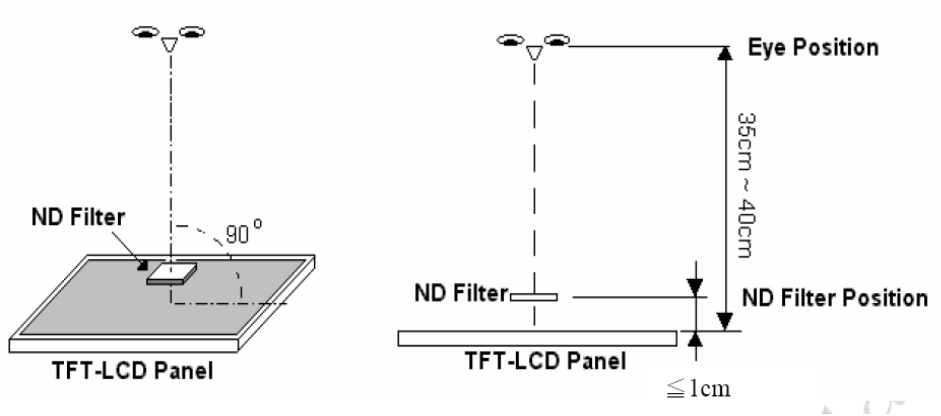




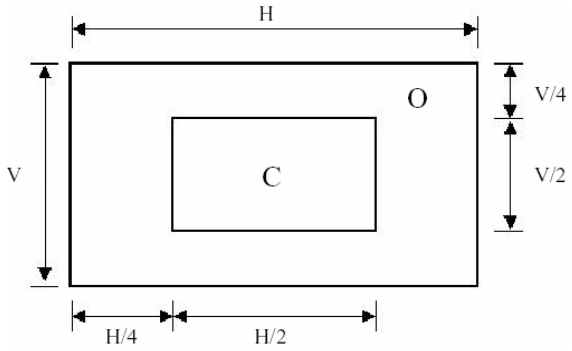
1. fiber

(a) 白點 / 黑點　(b) 偏光板氣泡

[註 2] 亮點的定義係透過 6% 穿透率的 ND 濾片判定，如下所示。

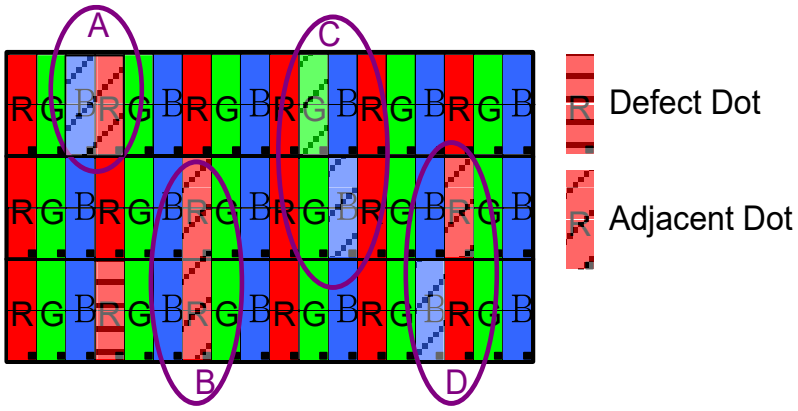


[註 3] 顯示區域



C 區：顯示區域中央　　**O 區**：顯示區域外圍

[註 4] 缺陷點與相鄰點的判定方式如下。下列情形（A、B、C、D 狀態）的相鄰缺陷點為允收範圍，包含相鄰的亮點與暗點，且於總數量中計為 2 個缺陷點。



(1) 未於上述定義、但被認定為問題的缺陷，應由雙方共同檢視並協商認定。

位於 Black Matrix 上、顯示區域之外的缺陷，不視為缺陷亦不計入。

[註 5] 根據 LCD 製造商提供的技術資訊，若靜態畫面長時間維持不變，LCD 顯示器可能發生殘影（image retention）。ICOP 建議客戶勿讓 LCD 顯示同一靜態畫面超過 4 小時而完全不變動；若 LCD 需長時間開啟，亦請啟用螢幕保護程式，以避免殘影問題。

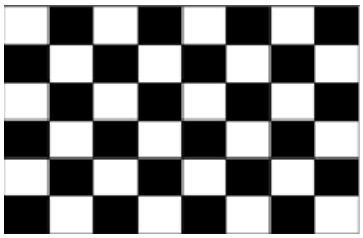
部分殘影在 LCD 關閉一段時間後會自行消失，但若 LCD 已發生烙印（screen burn），部分殘影將無法回復。

以下為 LCD 製造商的測試結果，供客戶參考。

測試項目	條件	備註
高溫運作	70℃ ;240hrs	
高溫儲存	80℃ ; 240hrs	
高溫高濕運作	60℃ ; 90%RH ;240hrs	無凝結
低溫運作	-20℃ ; 240hrs	背光模組全程開啟
低溫儲存	-30℃ ; 240hrs	
熱衝擊	–30℃（0.5 小時）~ 80℃（0.5 小時）；200 循環	
殘影	25℃ ; 4hrs	**註 5-1**
MTBF	20,000Hrs	

[註 5-1]

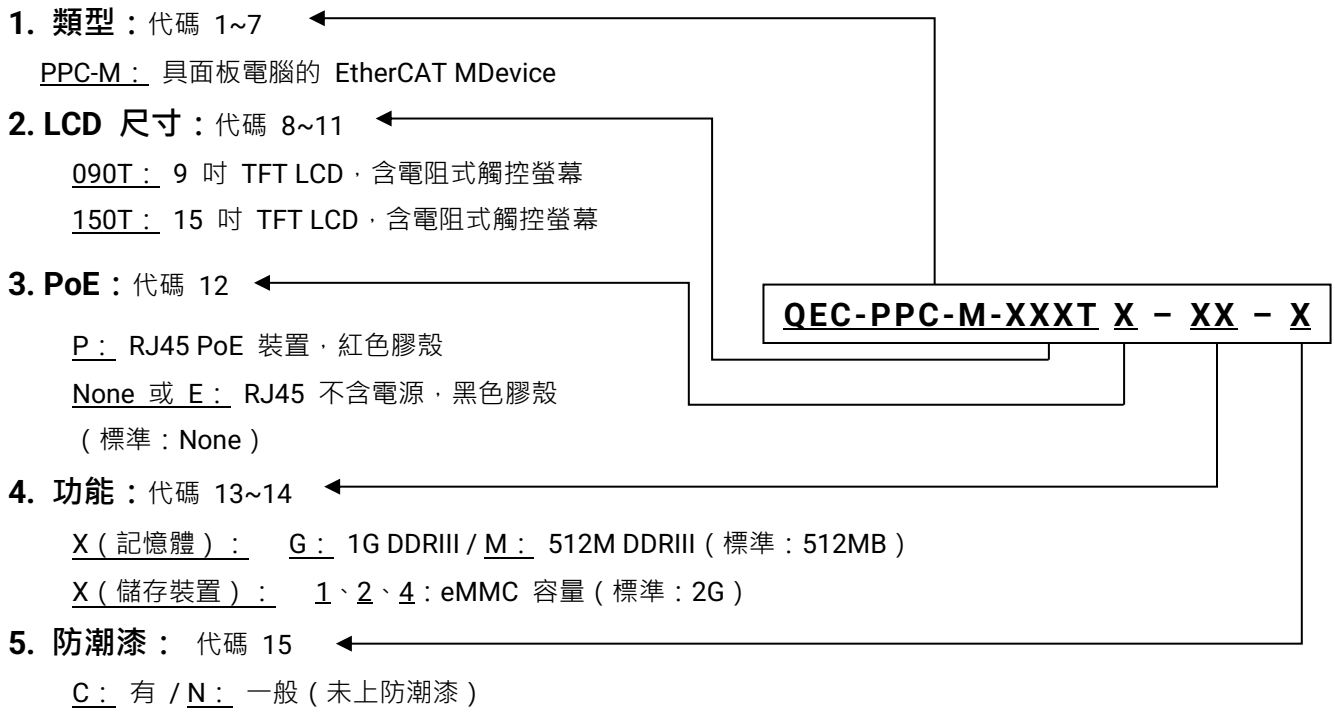
1. 殘影測試條件：25 ℃± 2 ℃。
2. 以測試圖樣持續運作 4 小時後，立即切換為灰階圖樣。
3. 5 分鐘後，mura 須完全消失。





1.5 訂購資訊

類型	LCD 尺寸	(*以下為客製化功能，未使用的欄位無須填寫；若客戶無特殊需求，將直接以標準料號出貨，例如 **QEC-PPC-M-090T**)					
		PoE	-	功能		-	防潮漆
				記憶體	儲存裝置		
QEC-PPC-M	XXXT	X		X	X		X



1.5.1 訂購料號

- **QEC-PPC-M-090T**： 搭載 9 吋 LCD 的 EtherCAT MDevice 面板電腦
- **QEC-PPC-M-090TP**： 搭載 9 吋 LCD 的 EtherCAT MDevice 面板電腦 / PoE
- **QEC-PPC-M-150T**： 搭載 15 吋 LCD 的 EtherCAT MDevice 面板電腦
- **QEC-PPC-M-150TP**： 搭載 15 吋 LCD 的 EtherCAT MDevice 面板電腦 / PoE

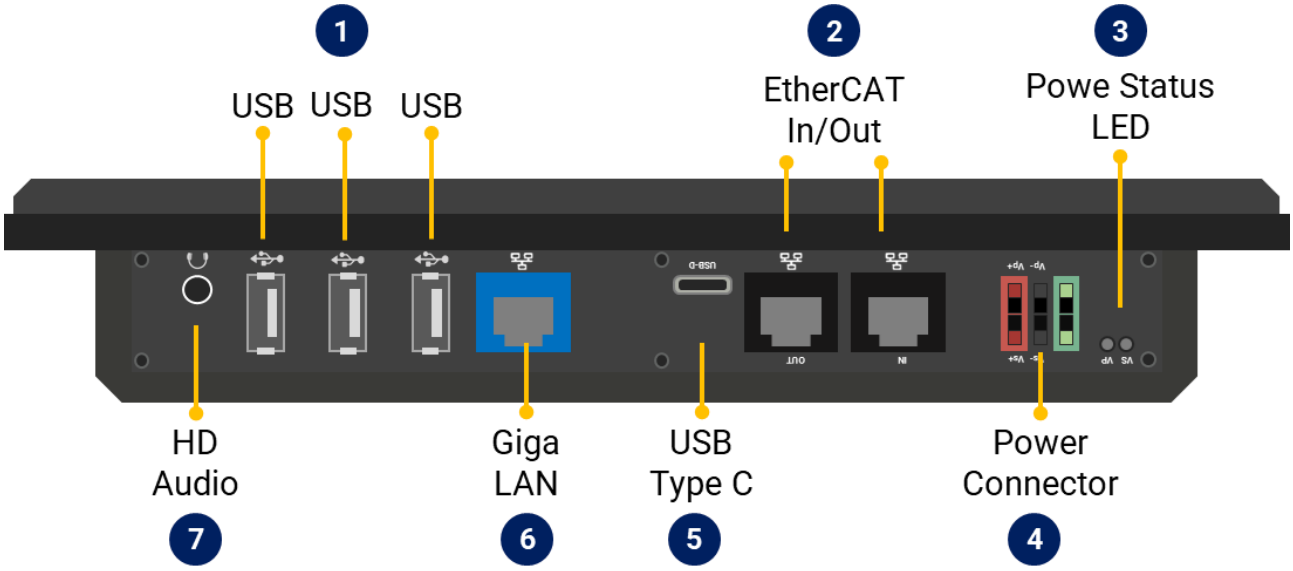
Ch. 2

硬體系統

2.1 一般技術資料

本節說明 QEC-PPC-M 各部位的功能。顯示器（螢幕）規格請參閱 Ch 1.2 規格。

範例以 **QEC-PPC-M-090T** 說明；**QEC-PPC-M-150T** 的功能完全相同。

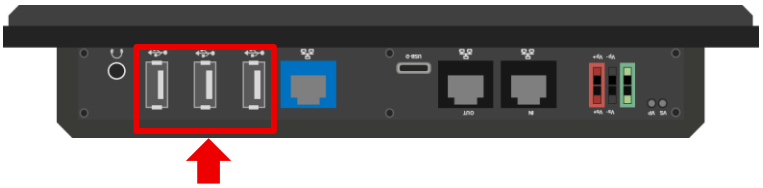


2.2 總覽

編號	說明		類型敘述	腳位
1	USB		USB 2.0	-
2	EtherCAT 介面	IN	外接 RJ45 連接器 （金手指）	8-pin
		OUT		8-pin
3	電源狀態指示燈		外部狀態指示燈	
4	電源連接器		端子台介面	6-pin
5	USB Type-C（除錯與上傳埠）		USB Type-C	-
6	Giga LAN		外接 RJ45 連接器 （金手指）	8-pin
7	HD Audio		外接音訊連接器	-

2.2.1 USB

如圖所示，共有三組支援熱插拔的標準 USB 2.0 埠。



此埠可用於連接：

- USB 隨身碟：用於檔案儲存、傳輸或存取組態資料。
- 鍵盤 / 滑鼠：供 HMI 顯示、控制或輸入使用。

***USB 隨身碟使用注意事項：**

- 請確認 USB 隨身碟已格式化為支援的檔案系統（例如 FAT32）。
- 大型檔案的傳輸可能受 USB 2.0 的速度限制。

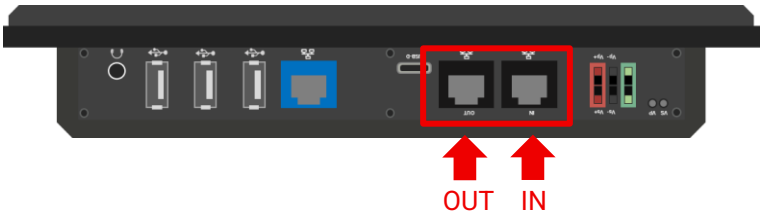
關於 USB 儲存裝置的更多使用細節，請參閱 Ch 4.7 USB 裝置使用說明章節。

***鍵盤 / 滑鼠使用注意事項：**

QEC-PPC-M 系列預設支援 LCD 與觸控。若您需要使用鍵盤與滑鼠功能，請於出貨前告知我們。

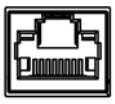
2.2.2 EtherCAT 介面

RJ45 連接器。



關於 EtherCAT 通訊的更多使用細節，請參閱 Ch 4.5 EtherCAT 通訊章節。

2.2.2.1 EC IN

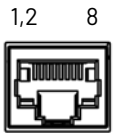


腳位	訊號名稱	腳位	訊號名稱
1	LAN1_TX+	2	LAN1_TX-
3	LAN1_RX+	4	VS+
5	VP+	6	LAN1_RX-
7	VS- (GND)	8	VP- (GND)

*PoE LAN 為紅色膠殼；一般 LAN 為黑色膠殼。

*L4、L5、L7、L8 為選配腳位，供 RJ45 Power IN/OUT 使用。

2.2.2.2 EC OUT



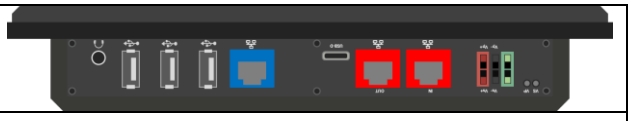
腳位	訊號名稱	腳位	訊號名稱
1	LAN2_TX+	2	LAN2_TX-
3	LAN2_RX+	4	VS+
5	VP+	6	LAN2_RX-
7	VS- (GND)	8	VP- (GND)

*PoE LAN 為紅色膠殼；一般 LAN 為黑色膠殼。

*L4、L5、L7、L8 為選配腳位，供 RJ45 Power IN/OUT 使用。

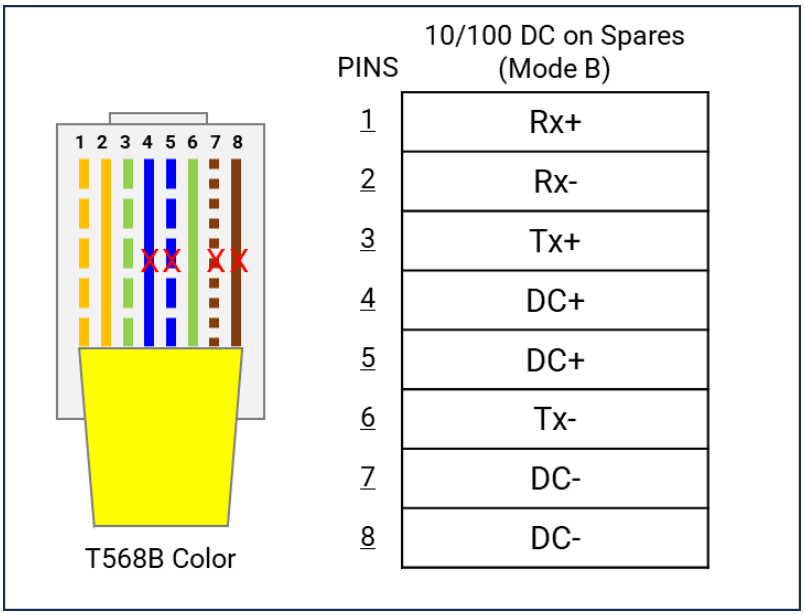
注意. QEC 的 PoE（Power over Ethernet）

在 QEC 產品的安裝上，使用者可輕易分辨 PoE 與非 PoE：RJ45 膠殼為紅色者為 PoE 型，膠殼為黑色者為非 PoE 型。

	
非 PoE 型	PoE 型

PoE（Power over Ethernet）是一種透過網路線供電的功能。QEC 可選配 PoE 功能以減少配線。實務上 PoE 需依系統設備選用，因此在評估與測試時請注意下列事項：

1. 當 PoE 裝置與非 PoE 裝置連接時，務必將網路線的第 4、5、7、8 腳位斷開（例如支援 PoE 的 QEC EtherCAT MDevice 連接第三方 EtherCAT SubDevice 時）。

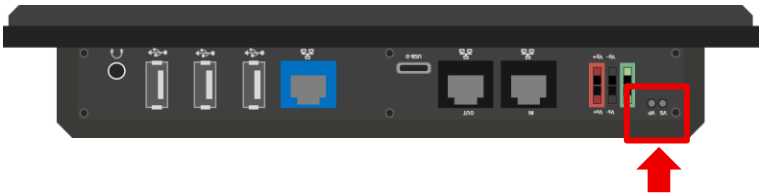


2. QEC 的 PoE 功能與 EtherCAT P 不同且互不相容。QEC 的 PoE 功能以 PoE Type B 為基礎。

3. QEC 的 PoE 供電能力最高為 24V/3A。

2.2.3 電源狀態指示燈

電源狀態指示燈說明。



Vs 為系統電源；Vp 為周邊電源與備援電源。

電源輸入為 24V（典型值）。指示燈狀態可提供高 / 低電壓警示。

*注意： 請務必同時接上 Vs 與 Vp。

若出現上述任一情形，請透過 info@icop.com.tw 與 ICOP 聯繫，以進行診斷與處理。

2.2.4 電源連接器

Euroblock 連接器。

4-pin 電源輸入 / 輸出與 2-pin FGND。



Vs 為系統電源；Vp 為周邊電源與備援電源。

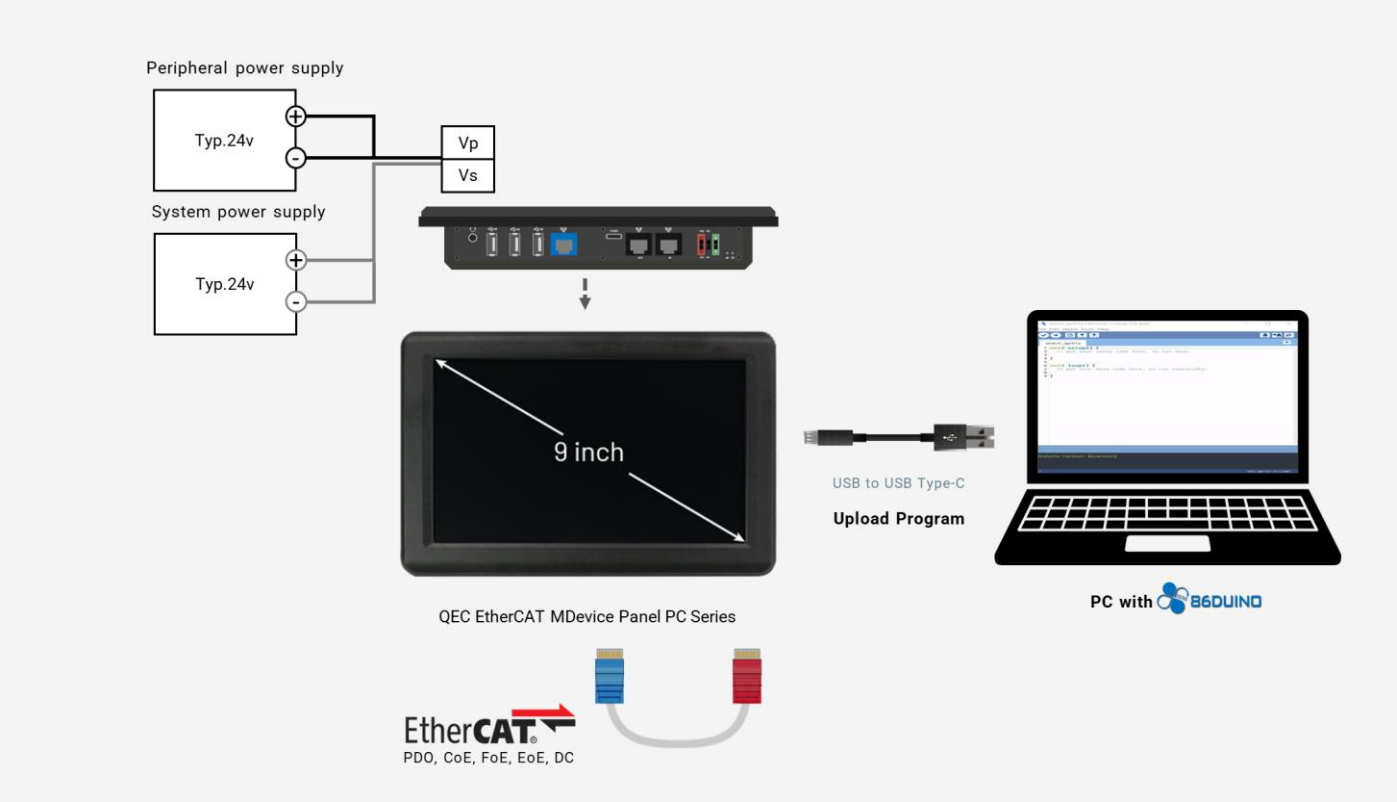


腳位	訊號名稱	腳位	訊號名稱	腳位	訊號名稱
1	FGND	3	Vs- (GND)	5	Vs+
2	FGND	4	Vp- (GND)	6	Vp+

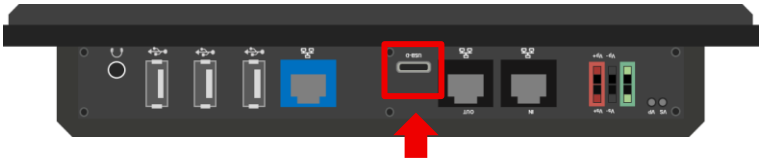
* 電源輸入電壓為 +12 至 +50VDC (典型值 +24VDC)

2.2.5 USB Type C

QEC-PPC-M 系列具備 USB Type-C 埠，主要用於程式上傳與除錯。您可使用 USB 轉 USB Type-C 連接線將裝置連接至電腦，以上傳程式碼並進行系統設定。



USB Type-C 埠的位置如圖所示。

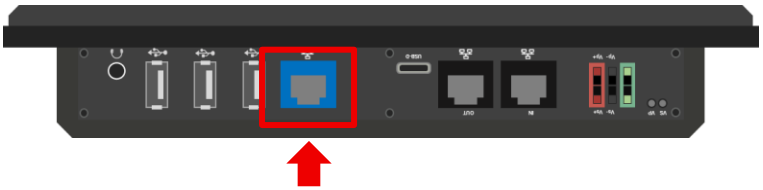


QEC-PPC-M 系列的程式撰寫與設定方式，請參閱 Ch. 4 快速上手。

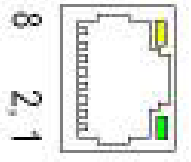
2.2.6 Giga LAN

QEC-PPC-M 系列具備**一組 Giga LAN 埠**，專供一般網路用途的對外 Ethernet 通訊使用。

Giga LAN 埠為藍色。



腳位定義：



腳位	訊號名稱	腳位	訊號名稱
L1	GTX+	L2	GTX-
L3	GRX+	L4	GTXC+
L5	GTXC-	L6	GRX-
L7	GRXD+	L8	GRXD-

Giga LAN 埠支援高速 Ethernet 對外通訊。若要驅動 Giga LAN，可參考 Ethernet library 或 Modbus Library。

關於 Ethernet 或 Modbus TCP 的更多使用細節，請參閱 Ch 4.8 Giga LAN 設定章節。

2.2.7 HD Audio

QEC-PPC-M 系列具備 HD Audio（Line-Out）埠，可連接喇叭或耳機等外部音訊裝置。

HD Audio 埠的位置如圖所示。



相容於標準 3.5mm 音訊接頭。

關於 HD Audio 的更多使用細節，請參閱 Ch 4.10 音訊使用說明章節。

2.3 Extension I/O（選配）

2.3.1 Arduino 腳位定義

如圖所示，我們將 QEC MDevice PPC 系列上所有的 Arduino 腳位都保留在機殼內部。



腳位分為兩類：**Arduino 標準腳位**與 **86Duino 專屬腳位**。

使用者可透過 **86Duino Coding IDE 軟體**輕鬆控制這些腳位。若要驅動 Arduino 腳位，可參考 Libraries 與 Language 文件。

***Arduino 腳位使用注意事項：**

在 QEC 系統上使用 `digitalWrite()` 或 `pinMode()` 等 Arduino 風格函式時，必須依照 86Duino 的腳位對應表。

例如，若要控制接在 J5 Pin 3 (GP00) 的 GPIO，程式中應使用腳位編號 2，寫法如下：

```
digitalWrite(2, HIGH);
```

請參考下方的 86Duino 腳位配置表，確認實體腳位位置與軟體腳位編號之間的正確對應關係。

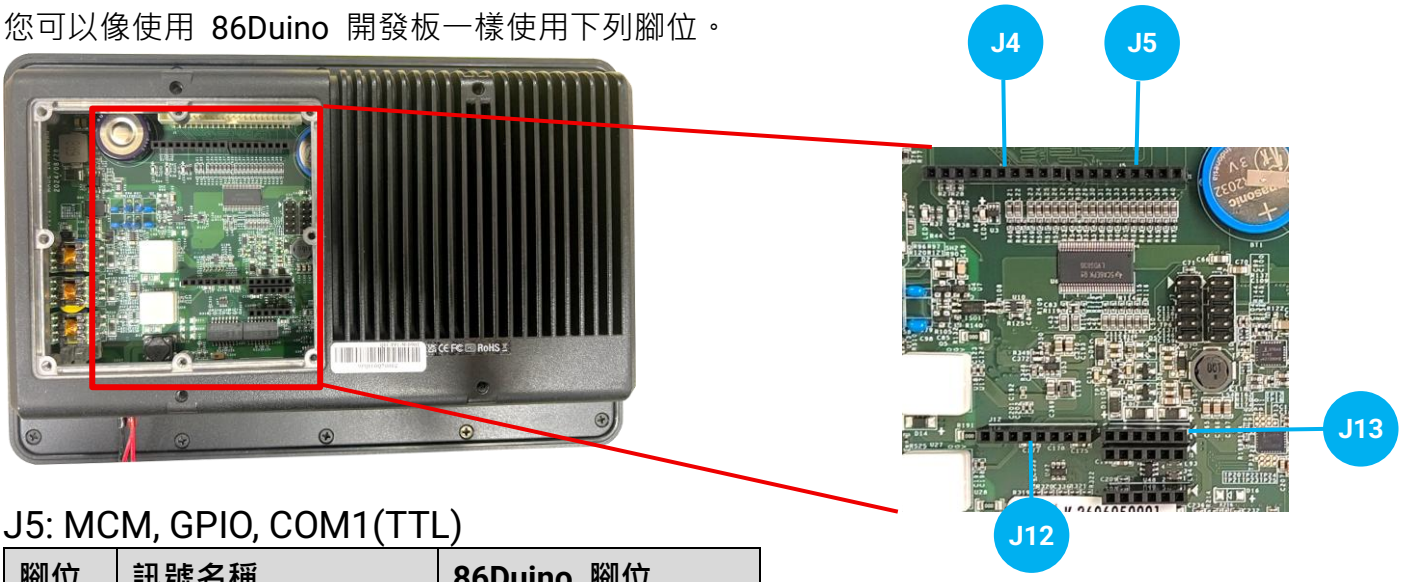
表：86Duino 腳位配置

連接器	腳位	訊號	86Duino 腳位配置
J5	1	RXD1#	0 (Serial1)
	2	TXD1#	1 (Serial1)
	3	GP00	2
	4	MCM-3	3
	5	GP02	4
	6	MCM-5	5
	7	MCM-6	6
	8	GP05	7
J4	1	GP30	8
	2	MCM-9	9
	3	MCM-10	10
	4	MCM-11	11
	5	GP31	12
	6	MCM-13	13
	7	GND	-
	8	-	-
	9	I2C0_SDA	（線材）
	10	I2C0_SCL	（線材）
J13	6	GP40ADC	14 (A0)
	5	GP41ADC	15 (A1)
	4	GP42ADC	16 (A2)
	3	GP43ADC	17 (A3)
	2	GP56ADC	18 (A4)
	1	GP57ADC	19 (A5)
J14	1	-	-
	2	GP35	23
	3	GP36	24
	4	GP37	25
	5	GND	-
	6	VCC	-
J16	1	CAN1_L	(CAN1)
	2	CAN1_H	(CAN1)
	3	GND	-
	4	CAN0_L	(CAN)
	5	CAN0_H	(CAN)
	6	VCC3	-

連接器	腳位	訊號	86Duino 腳位配置
J7	1	SPI0_DI	(SPI)
	2	VCC	-
	3	SPI0_CLK	(SPI)
	4	SPI0_DO	(SPI)
	5	RESET-	-
	6	GND	-
	7	SPI0_CS	(SPI)
	8	-	-
	9	-	-
	10	-	-
J8	1	SPI1_DI	(SPI1)
	2	VCC	-
	3	SPI1_CLK	(SPI1)
	4	SPI1_DO	(SPI1)
	5	RESET-	-
	6	GND	-
	7	SPI1_CS	(SPI1)
	8	-	-
	9	RS485+	(Serial485 / SerialCOM)
	10	RS485-	(Serial485 / SerialCOM)

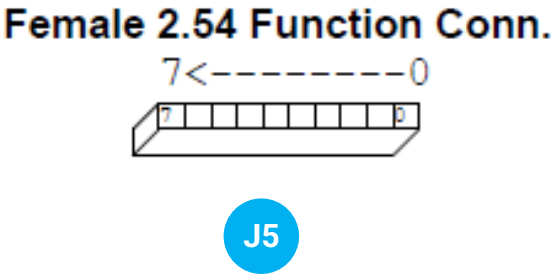
2.3.1.1 Arduino 標準腳位

您可以像使用 86Duino 開發板一樣使用下列腳位。



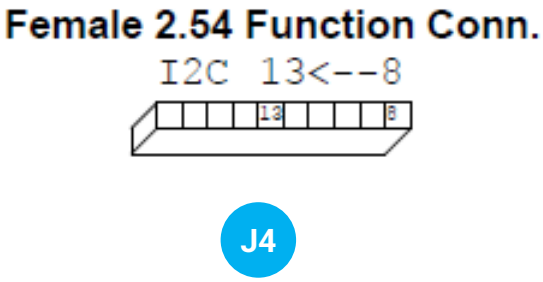
J5: MCM, GPIO, COM1(TTL)

腳位	訊號名稱	86Duino 腳位
1	RXD1#	0 (Serial1)
2	TXD1#	1 (Serial1)
3	GP00	2
4	MCM-3	3
5	GP02	4
6	MCM-5	5
7	MCM-6	6
8	GP05	7



J4: I2C0, MCM, GPIO

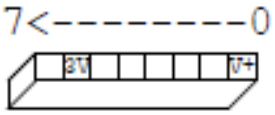
腳位	訊號名稱	86Duino 腳位
1	GP30	8
2	MCM-9	9
3	MCM-10	10
4	MCM-11	11
5	GP31	12
6	MCM-13	13
7	GND	-
8	-	-
9	I2C0_SDA	（線材）
10	I2C0_SCL	（線材）



*MCM 訊號等同於 Arduino 的 PWM 訊號。

J12: Power source, RESET-

腳位	訊號名稱	86Duino 腳位
1	VCC	-
2	GND	-
3	GND	-
4	VCC	-
5	VCC3	-
6	RESET-	-
7	VCC3	-
8	-	-

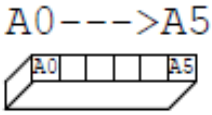


Female 2.54 Function Conn.

J13

J13: ADC/GPIO

腳位	訊號名稱	86Duino 腳位
6	GP40ADC	14 (A0)
5	GP41ADC	15 (A1)
4	GP42ADC	16 (A2)
3	GP43ADC	17 (A3)
2	GP56ADC	18 (A4)
1	GP57ADC	19 (A5)

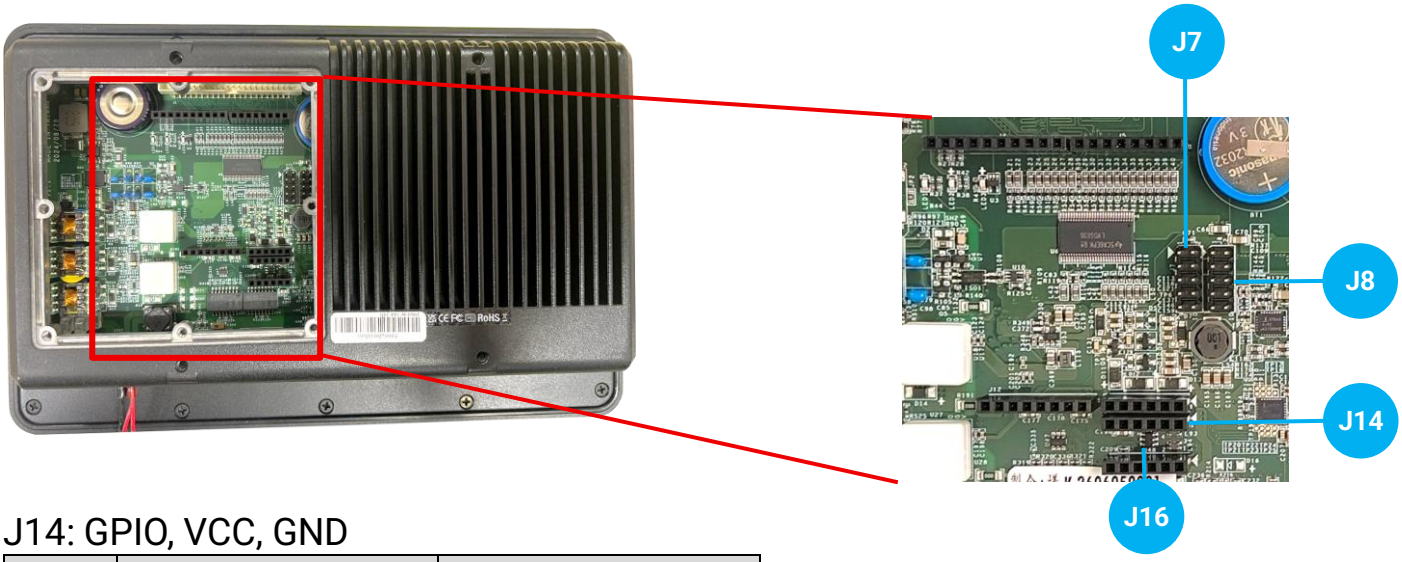


Female 2.54 Function Conn.

J14

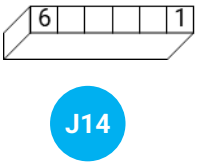
2.3.1.2 QEC 86Duino 專屬腳位

QEC MDevice PPC 系列另有提供 86Duino 專屬函式庫所使用的腳位。



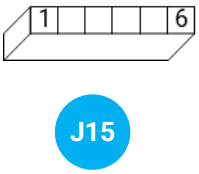
J14: GPIO, VCC, GND

腳位	訊號名稱	86Duino 腳位
1	-	-
2	GP35	23
3	GP36	24
4	GP37	25
5	GND	-
6	VCC	-



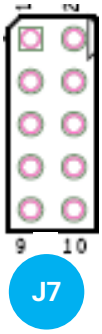
J16: CAN0 and CAN1 bus

腳位	訊號名稱	86Duino 腳位
1	CAN1_L	(CAN1)
2	CAN1_H	(CAN1)
3	GND	-
4	CAN0_L	(CAN)
5	CAN0_H	(CAN)
6	VCC3	-



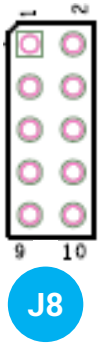
J7: SPI, RESET-

腳位	訊號名稱	86Duino 腳位
1	SPI0_DI	(SPI)
2	VCC	-
3	SPI0_CLK	(SPI)
4	SPI0_DO	(SPI)
5	RESET-	-
6	GND	-
7	SPI0_CS	(SPI)
8	-	-
9	-	-
10	-	-



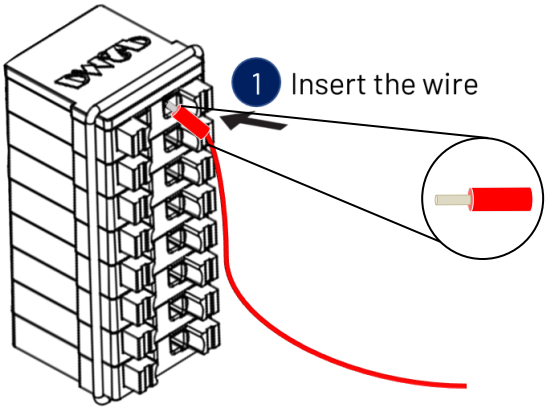
J8: SPI, RESET-, RS485+/-

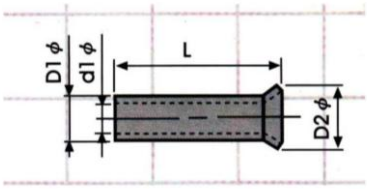
腳位	訊號名稱	86Duino 腳位
1	SPI1_DI	(SPI1)
2	VCC	-
3	SPI1_CLK	(SPI1)
4	SPI1_DO	(SPI1)
5	RESET-	-
6	GND	-
7	SPI1_CS	(SPI1)
8	-	-
9	RS485+	(Serial485 / SerialCOM)
10	RS485-	(Serial485 / SerialCOM)



2.4 連接器配線

2.4.1 將線材接入連接器

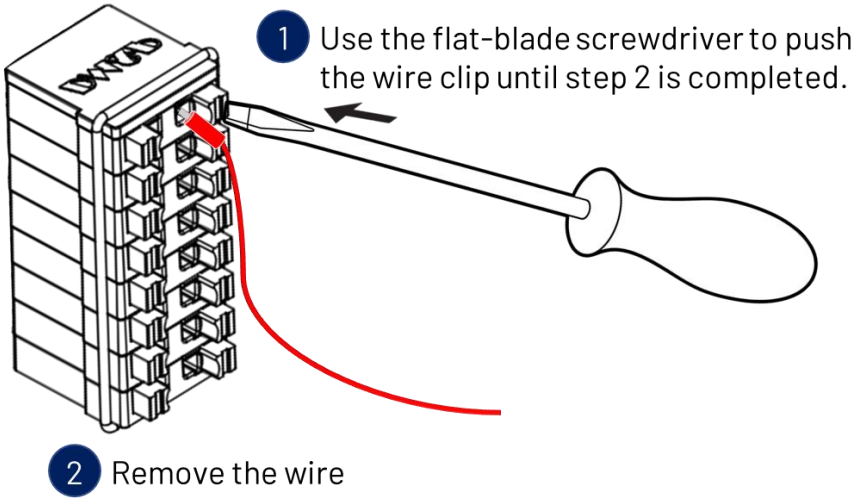




絕緣端子尺寸（mm）

位置	L	ØD1	Ød1	ØD2
CN 0.5-6	6.0	1.3	1.0	1.9
CN 0.5-8	8.0	1.3	1.0	1.9
CN 0.5-10	10.0	1.3	1.0	1.9

2.4.2 將線材自連接器移除



Ch. 3

硬體安裝

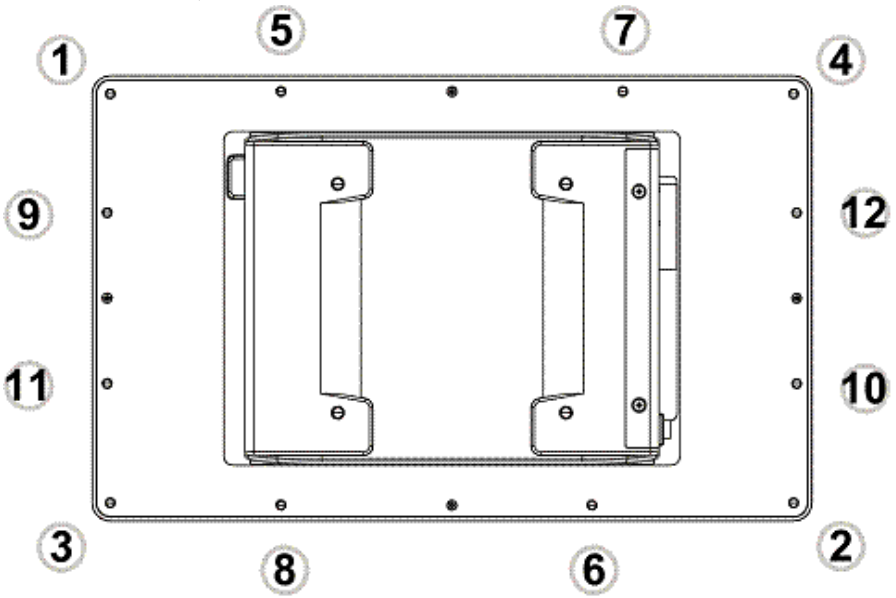
3.1 安裝說明

本程序適用於搭載 9 吋或 15 吋觸控顯示器的 QEC-PPC-M 系列機種。
安裝方式。

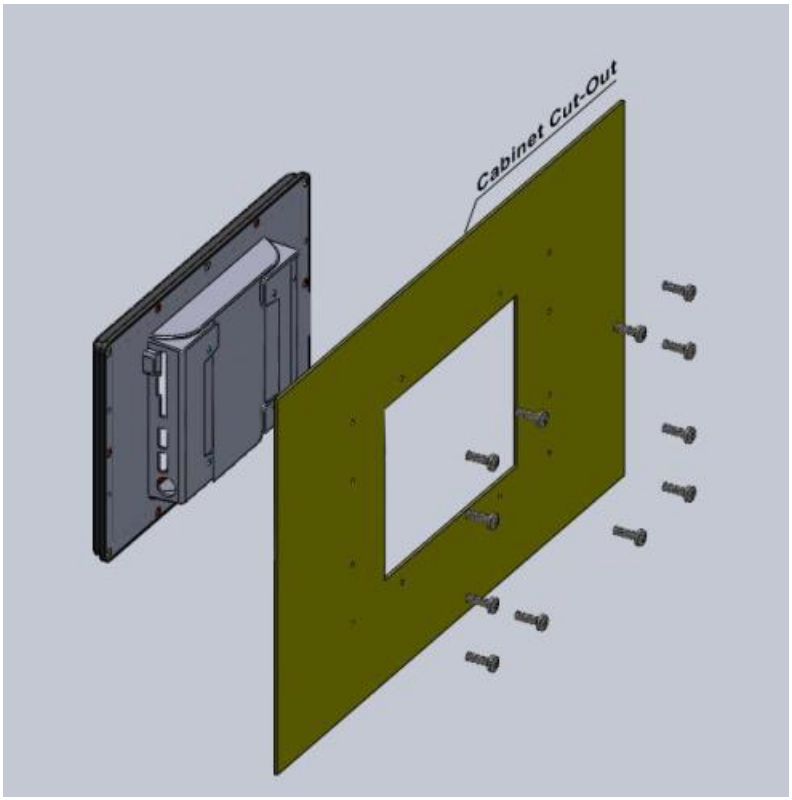
3.1.1 一般準則

採前面板嵌入式安裝，含防水墊圈，使用 **M3** 螺絲固定。

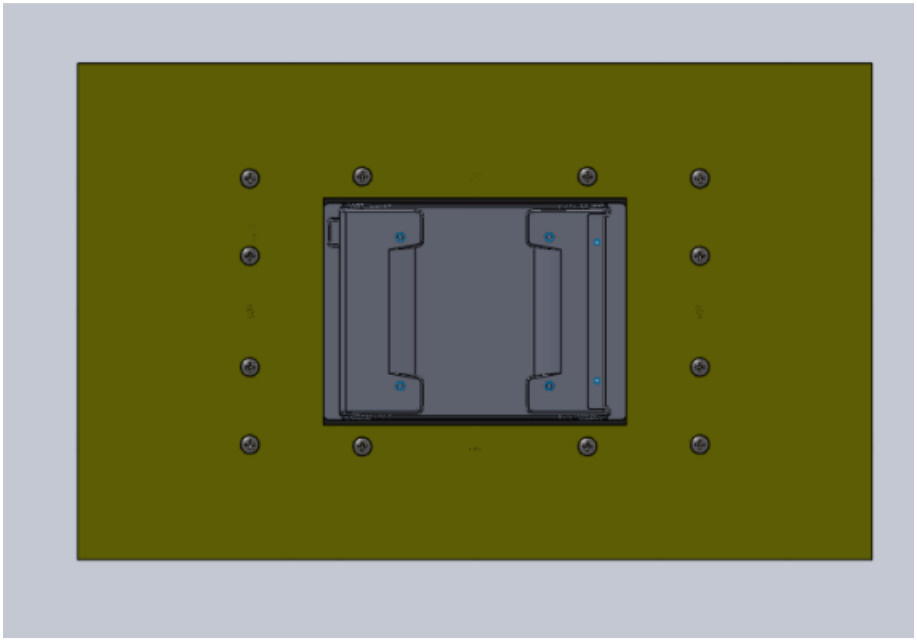
3.1.1.1 QEC-PPC-M-090T



1. 在面板上裁切安裝開孔。(註 1)
2. 如下圖所示，必要時請以對角順序檢查並拆下十二顆 M3 螺絲。
3. 將 QEC-PPC-M-090T 正面朝下置於乾淨平整的表面上。
4. 將面板開孔由 QEC-PPC-M-090T 背面套入，直到面板完全貼合於防水墊圈上，並確認螺絲孔與 QEC-PPC-M-090T 上的螺絲孔對齊。
5. 螺絲規格為 M3*L (L = 面板厚度 + 6.0mm) (註 2)
6. 將十二顆 M3 螺絲全部裝入螺絲孔。(註 2)
7. 先以手指將 M3 螺絲鎖上，再以 M3 螺絲起子依對角順序鎖緊 (參見下圖) ; 最大扭力為 1.18Nm (12 kgf-cm)。

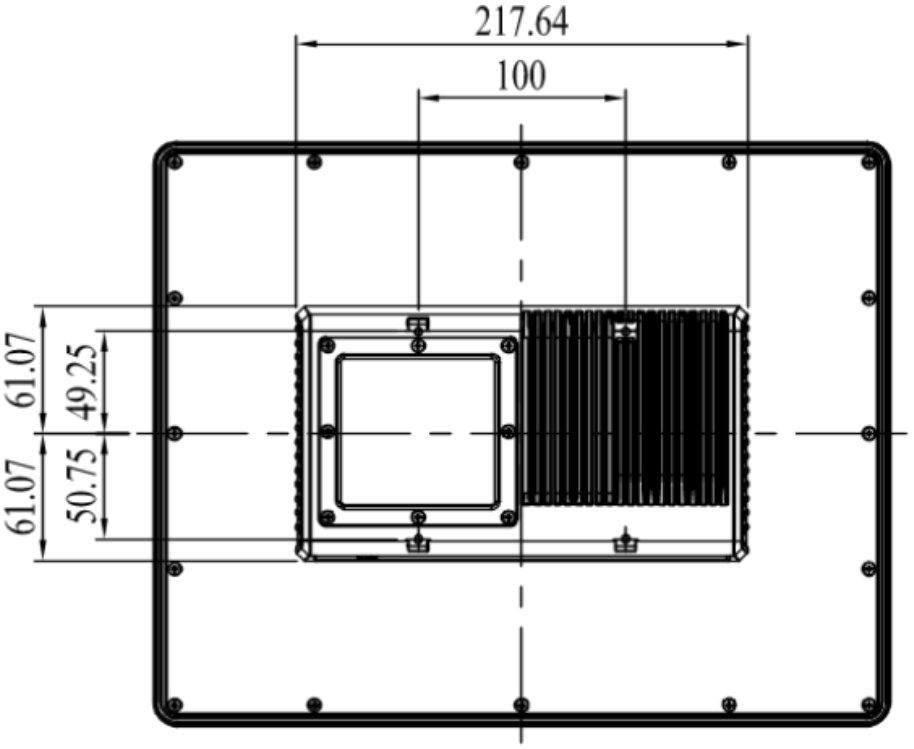


***註 1**：強烈建議由專業機械加工廠進行面板安裝開孔的裁切。

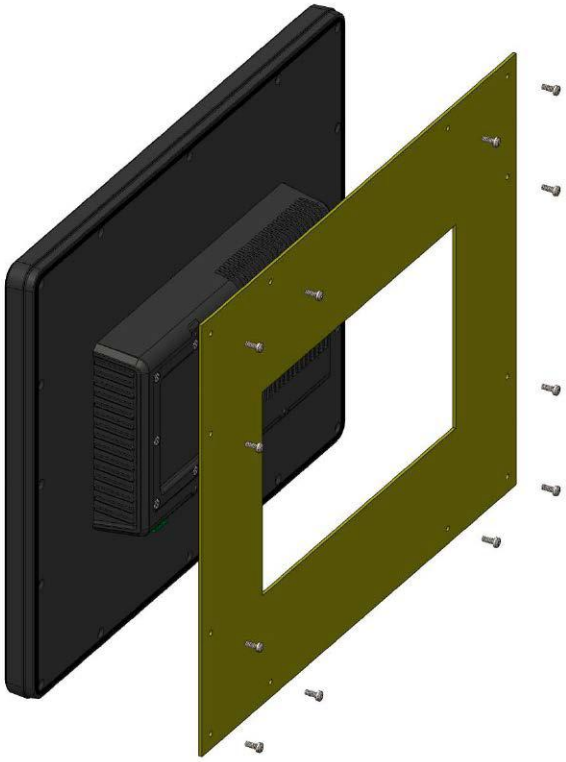


***註 2：** 十二顆 M3 螺絲的長度需依安裝面板的厚度而定。例如：QEC-PPC-M-090T 標準 M3 螺絲長度為 6mm；若您的安裝面板厚度為 3mm、墊圈厚度為 1mm，則須改用 10mm 的 M3 螺絲。

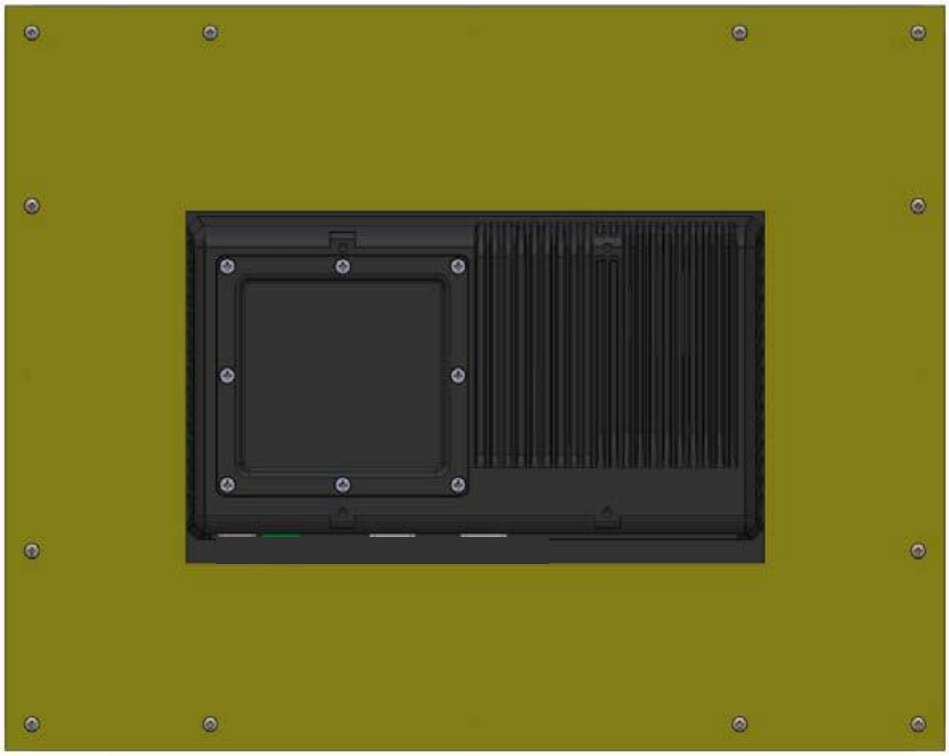
3.1.1.2 QEC-PPC-M-150T



1. 在面板上裁切安裝開孔。(註 1)
2. 如下圖所示，必要時請以對角順序檢查並拆下十二顆 M3 螺絲。
3. 將 QEC-PPC-M-150T 正面朝下置於乾淨平整的表面上。
4. 將面板開孔由 QEC-PPC-M-150T 背面套入，直到面板完全貼合於防水墊圈上，並確認螺絲孔與 QEC-PPC-M-150T 上的螺絲孔對齊。
5. 螺絲規格為 M3*L (L = 面板厚度 + 6.0mm) (註 2)
6. 將十二顆 M3 螺絲全部裝入螺絲孔。(註 2)
7. 先以手指將 M3 螺絲鎖上，再以 M3 螺絲起子依對角順序鎖緊 (參見下圖) ; 最大扭力為 0.44Nm (4.5 kgf-cm)。



*註 1：強烈建議由專業機械加工廠進行面板安裝開孔的裁切。

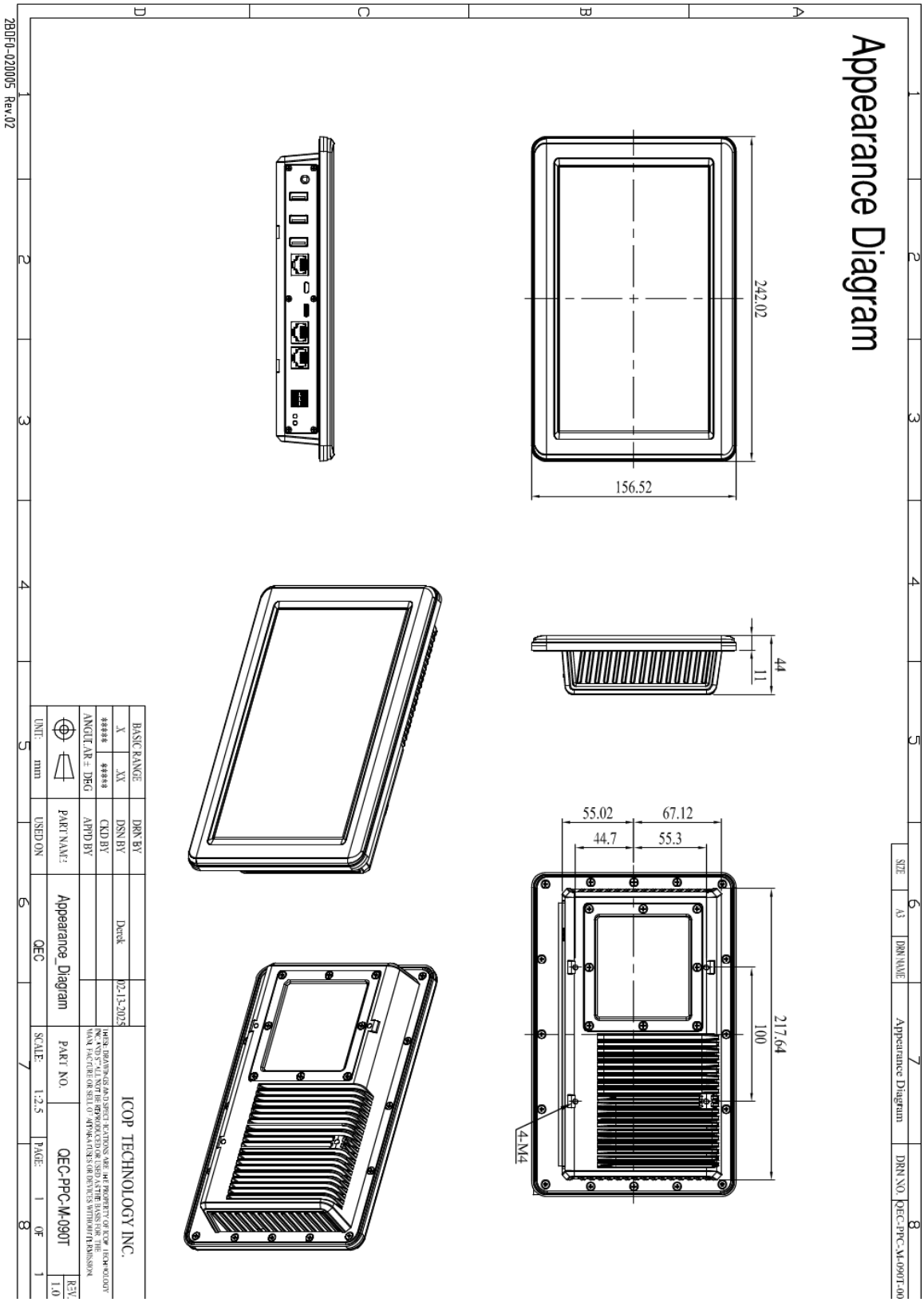


*註 2： 十二顆 M3 螺絲的長度需依安裝面板的厚度而定。例如：QEC-PPC-M-150T 標準 M3 螺絲長度為 6mm；若您的安裝面板厚度為 3mm、墊圈厚度為 1mm，則須改用 10mm 的 M3 螺絲。

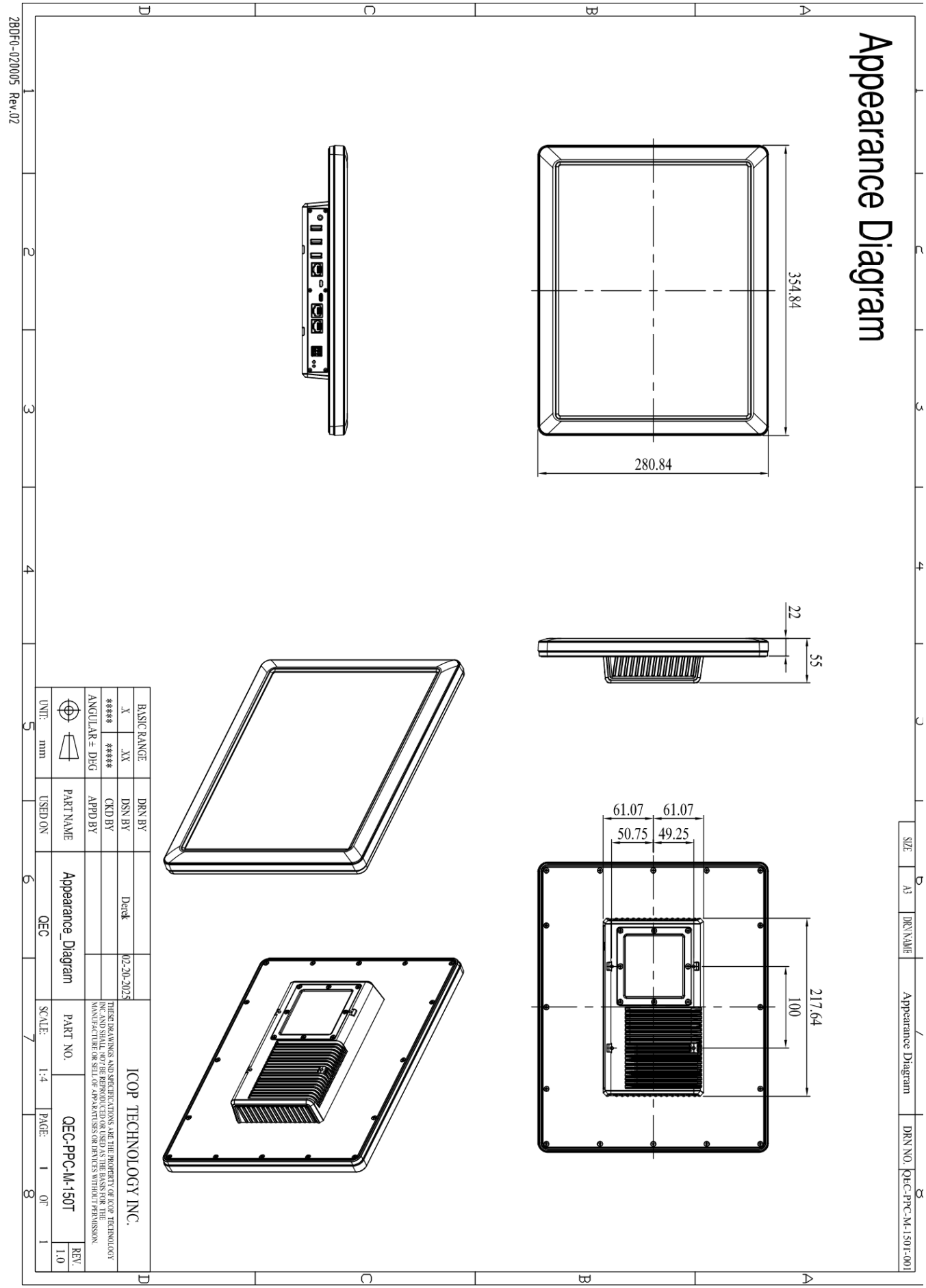
3.1.2 安裝尺寸

安裝開孔與尺寸總覽。

3.1.2.1 QEC-PPC-M-090T



3.1.2.2 QEC-PPC-M-150T



Ch. 4

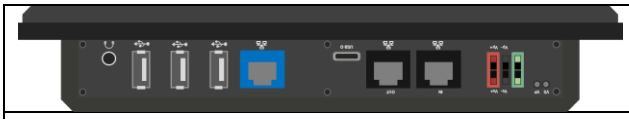
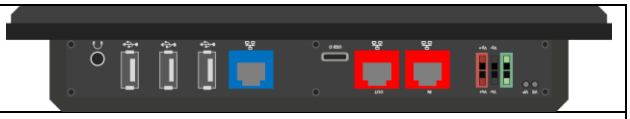
快速上手

（本章提供多語版本）

本章說明如何開始使用 QEC-PPC-M 系列及其開發軟體 86Duino Coding IDE 501+。

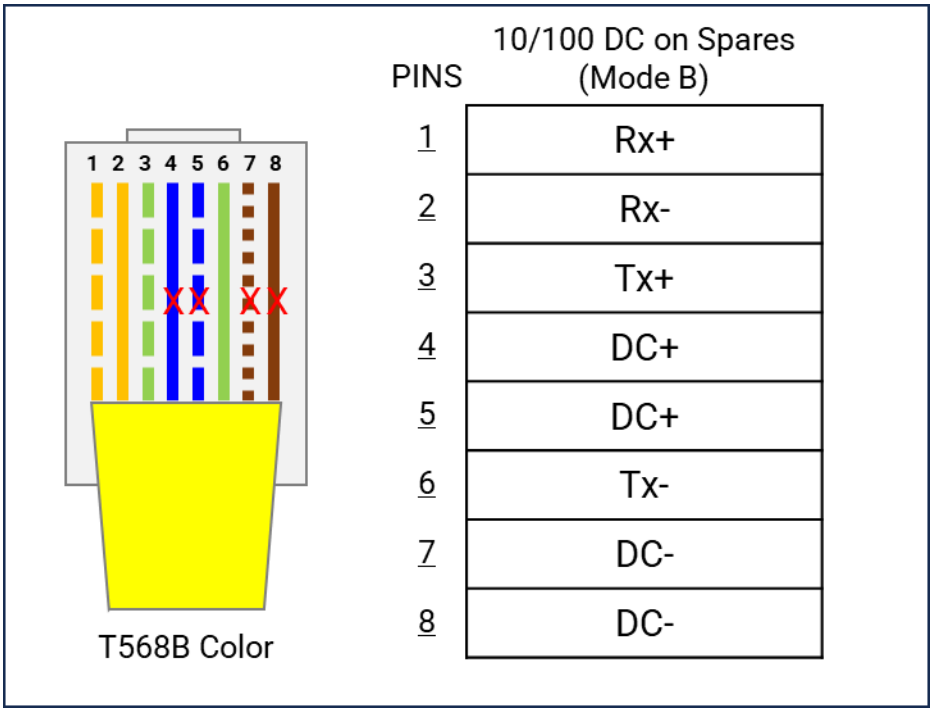
注意. QEC 的 PoE（Power over Ethernet）

在 QEC 產品的安裝上，使用者可輕易分辨 PoE 與非 PoE：RJ45 膠殼為紅色者為 PoE 型，膠殼為黑色者為非 PoE 型。

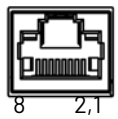
	
非 PoE 型	PoE 型

PoE（Power over Ethernet）是一種透過網路線供電的功能。QEC 可選配 PoE 功能以減少配線。實務上 PoE 需依系統設備選用，因此在評估與測試時請注意下列事項：

1. 當 PoE 裝置與非 PoE 裝置連接時，務必將網路線的第 4、5、7、8 腳位斷開（例如支援 PoE 的 QEC EtherCAT MDevice 連接第三方 EtherCAT SubDevice 時）。



2. QEC 的 PoE 功能與 EtherCAT P 不同且互不相容。QEC 的 PoE 功能以 PoE Type B 為基礎，腳位功能如下：



腳位	訊號名稱	腳位	訊號名稱
1	LAN1_TX+	2	LAN1_TX-
3	LAN1_RX+	4	VS+
5	VP+	6	LAN1_RX-
7	VS- (GND)	8	VP- (GND)

* PoE LAN 為紅色膠殼；一般 LAN 為黑色膠殼。

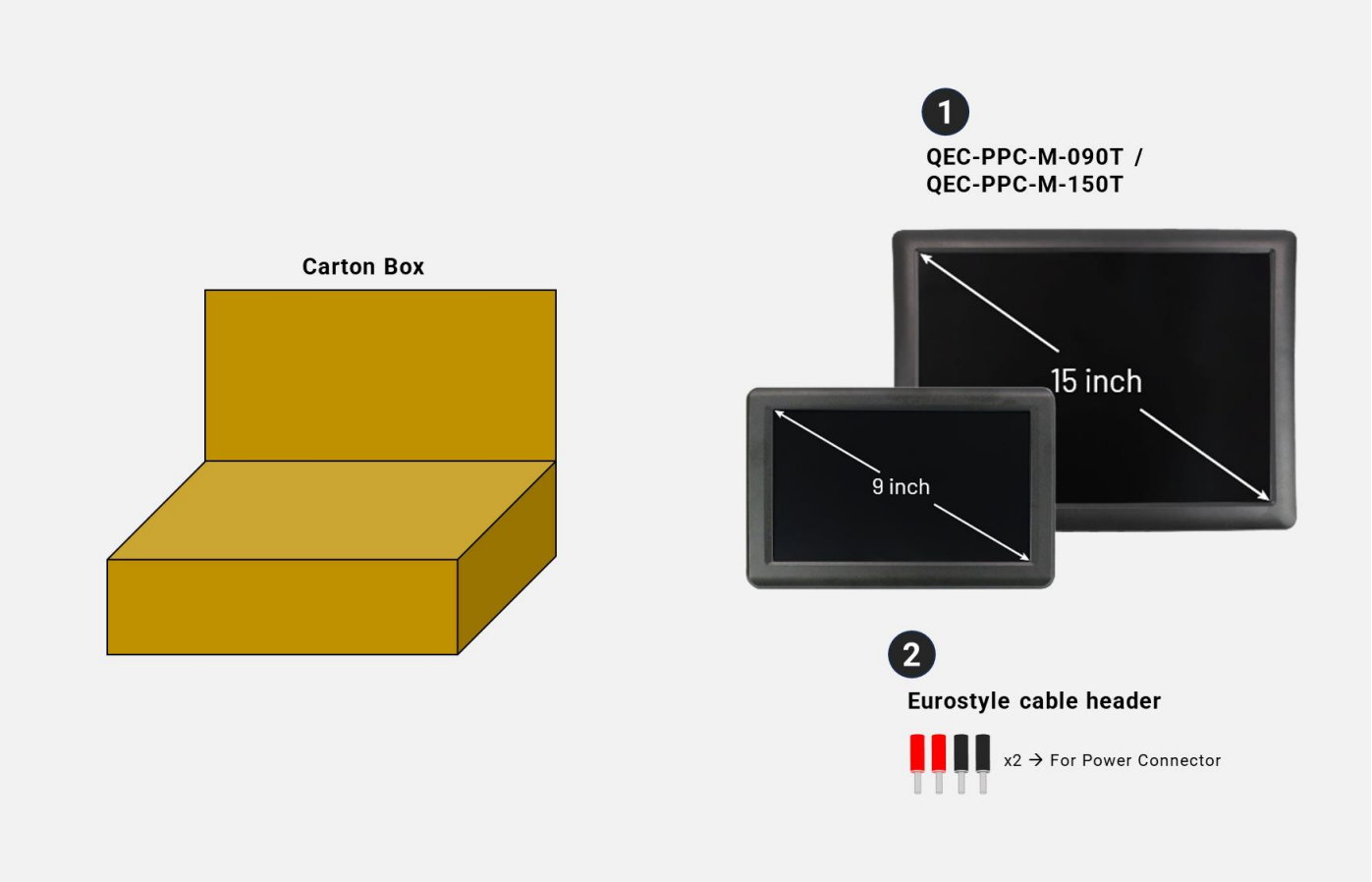
* L4、L5、L7、L8 為選配腳位，供 RJ45 Power IN/OUT 使用。

3. QEC 的 PoE 供電能力最高為 24V/3A。

4.1 包裝內容物

包裝內含下列項目：

1. QEC-PPC-M-090T / 150T 主機 x1
2. 線材組（Euro 型連接器）

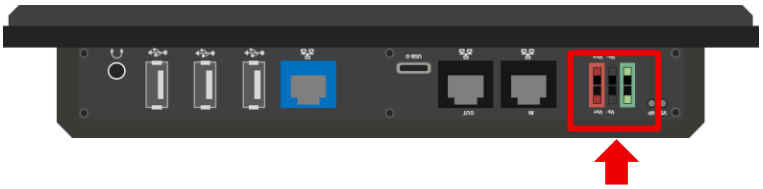


若包裝內容物有缺件或損壞，請與我們的業務窗口聯繫。此外，出貨用的包材與紙箱建議保留，以供日後存放或運送使用。

4.2 硬體配置

QEC-PPC-M 系列裝置出貨前已預先安裝好開發環境。使用者需自行下載軟體（請參閱 4.3 軟體 / 開發環境），並依照本手冊完成裝置設定。

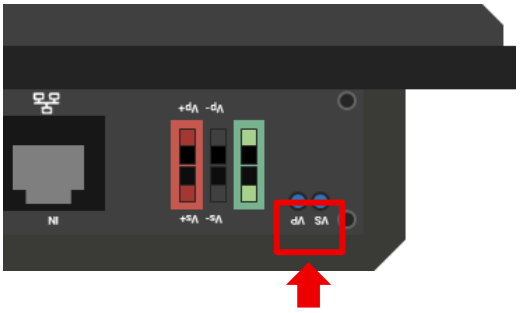
接上電源。



*注意： Vs 為系統電源；Vp 為周邊電源與備援電源。

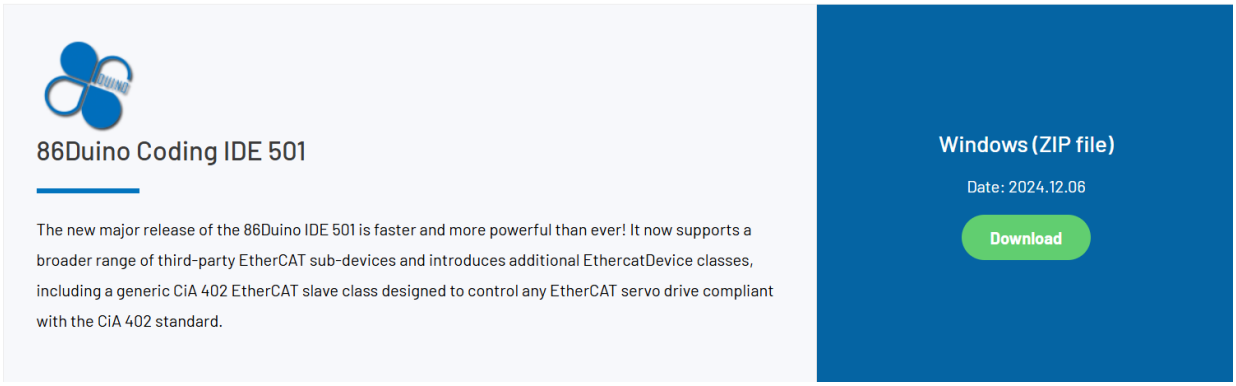
QEC MDevice 具有 Vs 與 Vp 兩組電源，兩者的電壓需求皆為 19V 至 50V 寬電壓範圍。

上電後，畫面即會亮起。



4.3 軟體 / 開發環境

請由 https://www.qec.tw/software/ 下載 86Duino IDE。



下載後請將 zip 檔解壓縮，無須另行安裝其他軟體，直接雙擊 **86duino.exe** 即可啟動 IDE。



***注意：** 若 Windows 出現警告訊息，請點選「其他資訊」，再點選一次「仍要執行」按鈕。

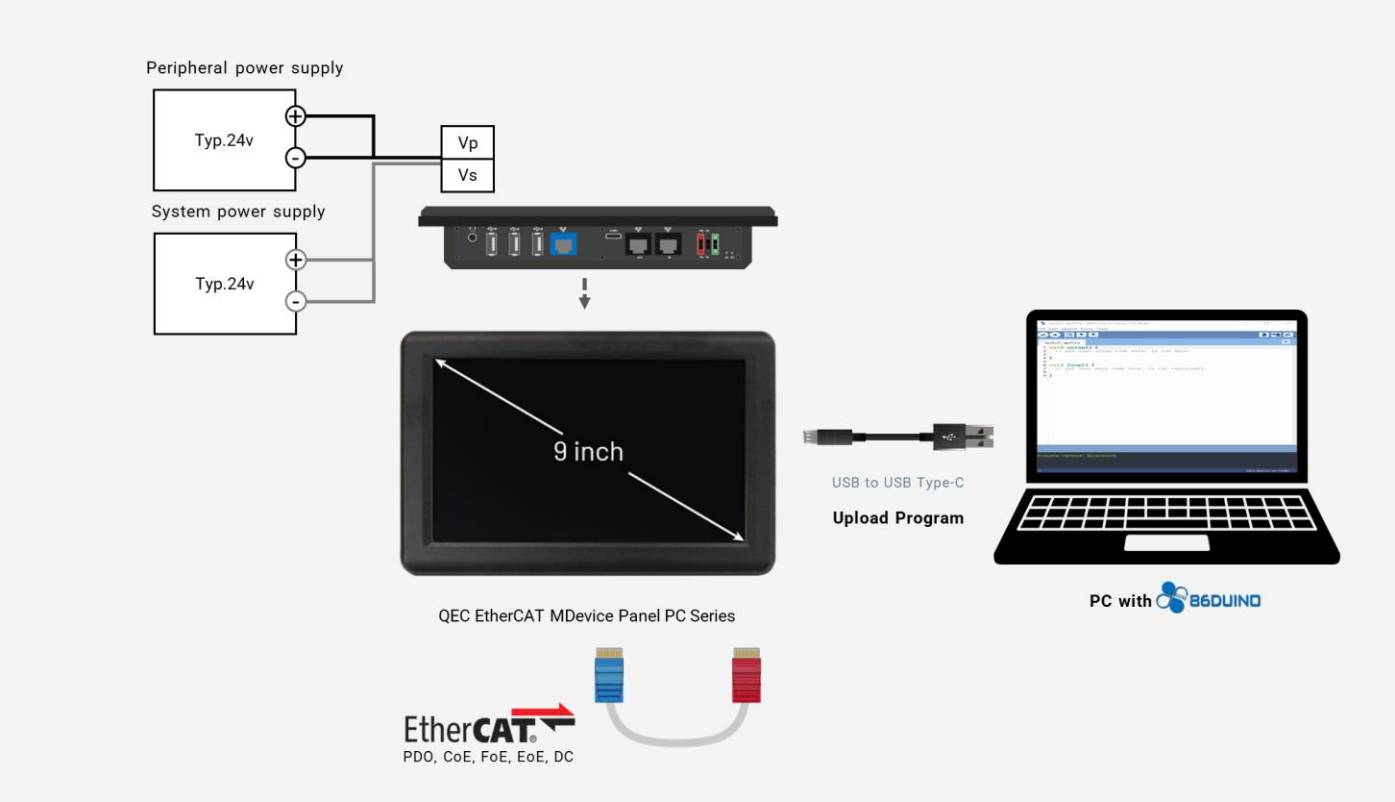
86Duino Coding IDE 501+ 的畫面如下。



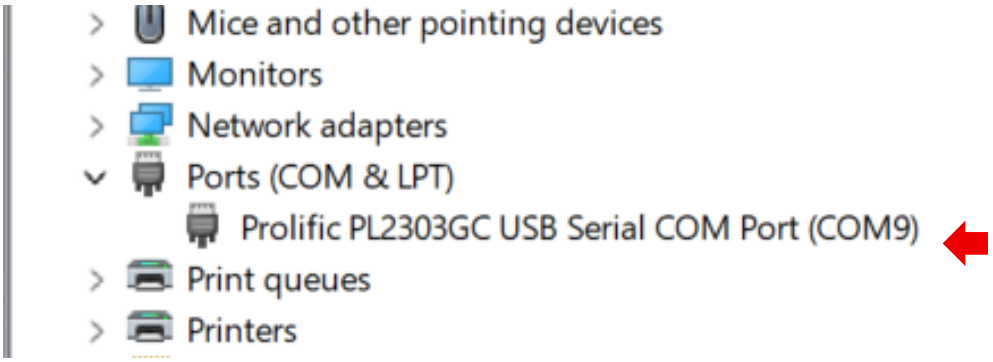
4.4 連接至 PC 並設定開發環境

請依下列步驟完成環境設定：

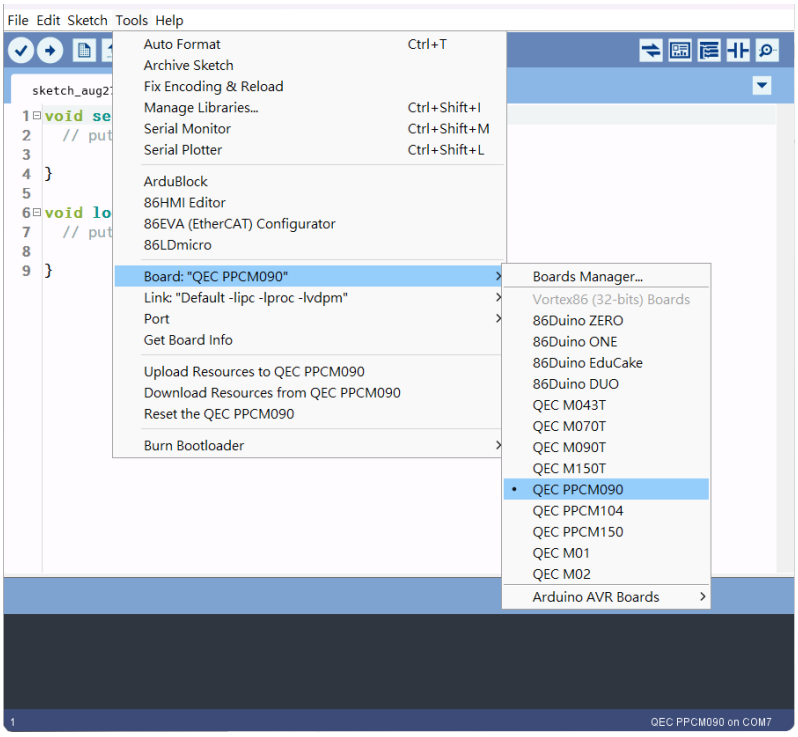
1. 使用 USB Type-C 轉 USB 連接線，將 QEC-PPC-M 裝置連接至已安裝 86Duino IDE 的 PC。



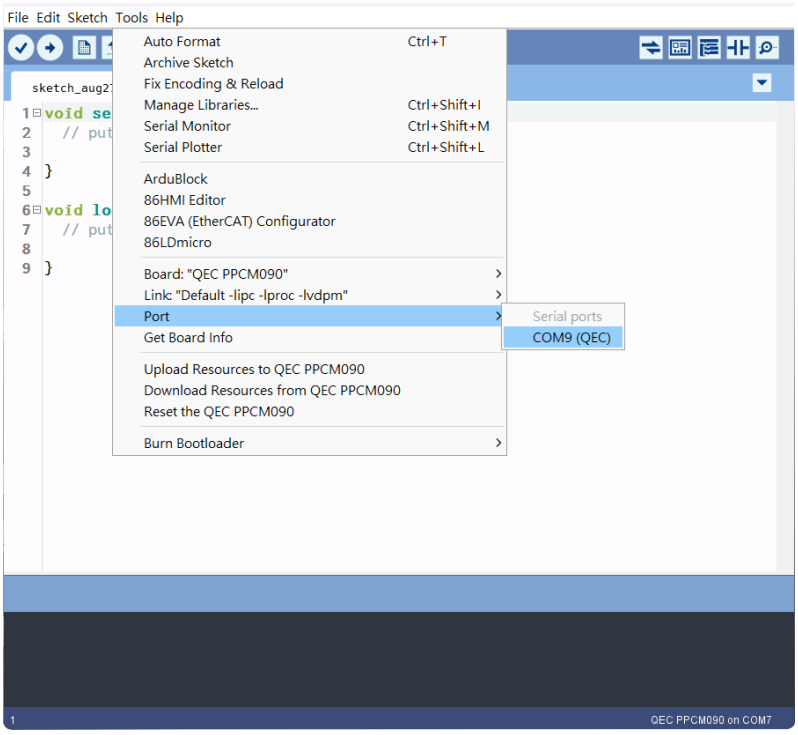
2. 開啟 QEC 電源。
3. 在 PC 上開啟「**裝置管理員**」→「**連接埠 (COM 和 LPT)**」並展開，應可看到已偵測到「**Prolific PL2303GC USB Serial COM Port (COMx)**」；若未出現，則需安裝對應的驅動程式。（Windows 版 PL2303 驅動程式可由此處下載）



4. 開啟 86Duino IDE。
5. 選擇正確的開發板：在 IDE 選單中選擇「**Tools**」→「**Board**」→「QEC PPCM090T」(或您所使用的 QEC MDevice 型號)。



6. 選擇連接埠：在 IDE 選單中選擇「**Tools**」→「**Port**」，並選取連接 QEC MDevice 的 USB 埠 (本例為 COM9 (QEC))。



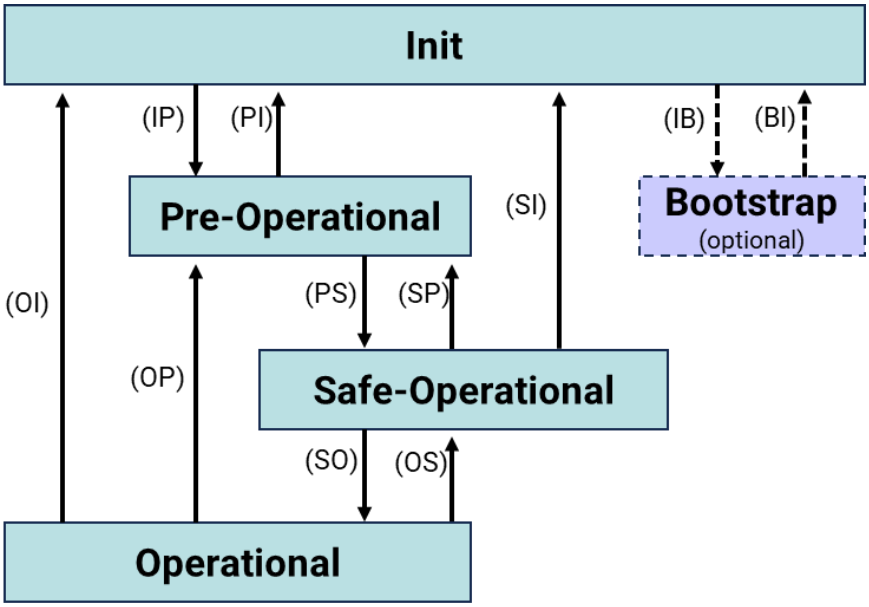
4.5 EtherCAT 通訊

本節介紹透過 QEC EtherCAT MDevice 設定 EtherCAT SubDevice 的兩種主要方式：直接撰寫程式碼，以及搭配 86EVA 使用程式碼。兩種方式皆兼顧彈性與效率，您可依熟悉程度與實際需求選用。

4.5.1 EtherCAT State Machine (ESM) 控制

本節說明如何透過 QEC EtherCAT MDevice 設定 EtherCAT SubDevice 並切換其各種運作狀態。理解 EtherCAT 網路中的狀態轉換與運作流程相當重要。

EtherCAT SubDevice 的狀態由 EtherCAT State Machine (ESM) 控制。在不同狀態下，EtherCAT SubDevice 可存取或執行的功能各不相同。EtherCAT MDevice 必須在每個狀態下對裝置送出特定命令，在 SubDevice 開機階段尤其如此。



狀態可區分如下：

- Init
- Pre-Operational
- Safe-Operational 以及
- Operational
- Boot

每個 EtherCAT SubDevice 開機後的正常狀態為 OP 狀態。

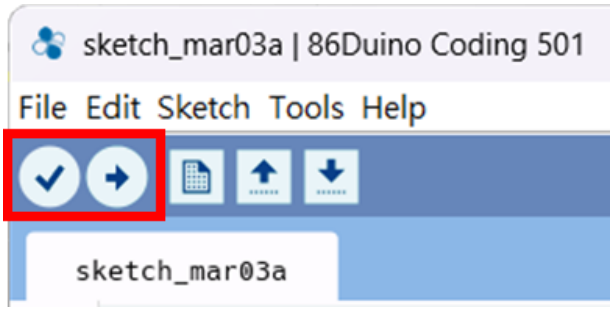
QEC EtherCAT MDevice 可透過 EtherCAT library 在 EtherCAT 網路中進行設定，並在 86Duino IDE 中撰寫控制動作。86Duino 開發環境主要分為兩個部分：`setup()` 與 `loop()`，分別對應初始化與主程式。

```
1  void setup() {
2    // put your setup code here, to run once:
3
4  }
5
6  void loop() {
7    // put your main code here, to run repeatedly
8
9  }
```

操作 EtherCAT 網路前，必須先完成一次組態設定。EtherCAT 裝置的流程應由 Init 狀態進行到 OP 狀態。

若要實作 EtherCAT 通訊，使用者必須使用 86Duino Coding IDE 501+ 的 EtherCAT Library。詳細資訊請參閱 Ch.5 軟體功能。

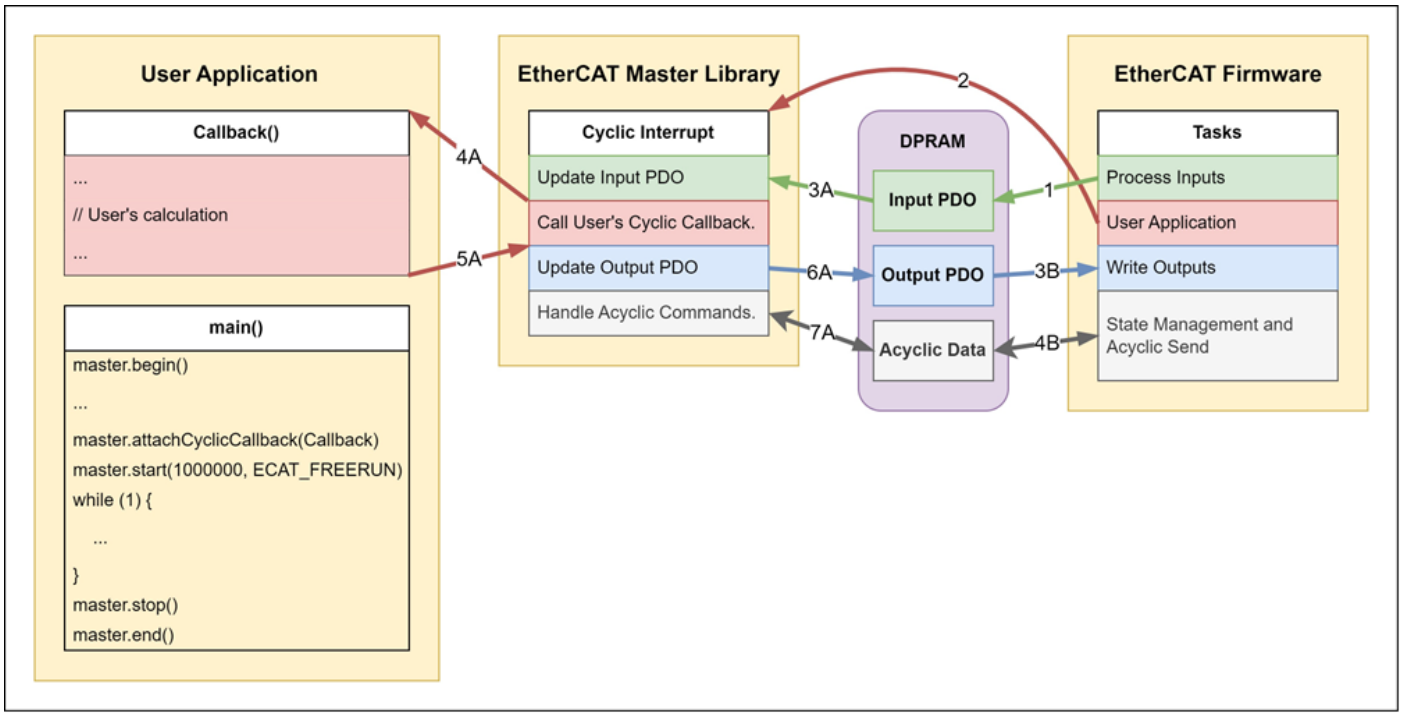
***注意：** 程式碼撰寫完成後，請點選工具列的 進行編譯；確認編譯完成且無錯誤後，即可點選進行上傳。



4.5.1.1 範例 1：Free Run 模式

在 Free Run 模式下，本地週期由應用控制器的本地計時器中斷觸發。MDevice 可選擇性地修改 cycle time，以改變計時器中斷的週期。在 Free Run 模式下，本地週期的運作獨立於通訊週期與 / 或 MDevice 週期。

QEC MDevice 的 Free Run 模式示意圖：



這是未啟用雙系統同步的 free-run 模式。如圖所示，編號箭頭代表動作順序。不過流程在第 3 步會出現分支：EtherCAT firmware 對 EtherCAT MDevice library 觸發 cyclic interrupt（第 2 步）後，並不會等待 EtherCAT MDevice library，而是直接繼續執行下一個動作。兩個系統各自獨立運作，彼此不同步。若使用者已註冊 cyclic callback，cyclic interrupt 會加以呼叫，如第 4A 步所示。

範例程式碼如下：

```
#include "Ethercat.h"

EthercatMaster master;

void CyclicCallback() {
  // ...
}

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

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

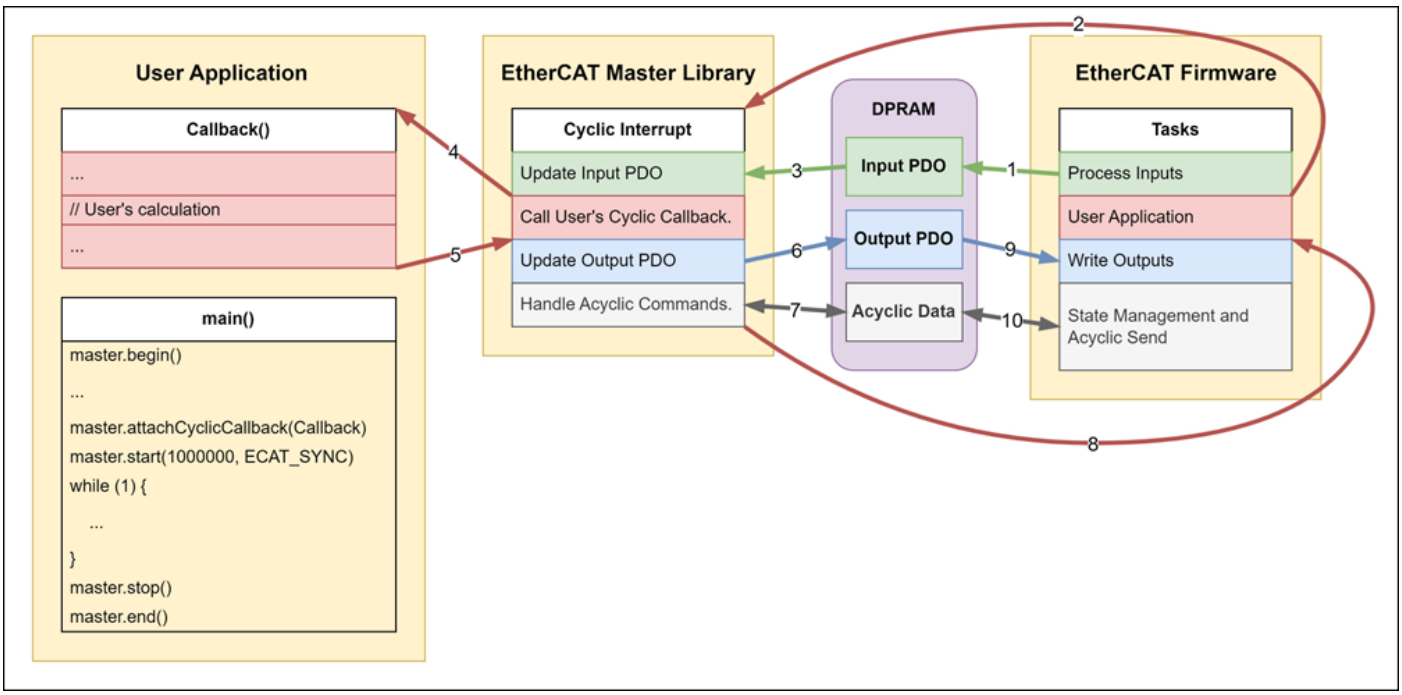
4.5.1.2 範例 2：SYNC 模式

本地週期在收到 SM2 事件（具週期性輸出）或 SM3 事件（無週期性輸出）時啟動。若有輸出資料，SubDevice 一般會與 SM2 事件同步；若無輸出資料，則與 SM3 事件同步，例如僅有週期性輸入的情況。

此模式提供下列選項：

- 與 SM2/3 事件同步
- 與 SM2/3 事件同步，並位移 Input Latch 時間

QEC MDevice 的 SYNC 模式示意圖：



此模式提供最高程度的雙系統同步。如圖所示，編號箭頭代表動作順序，流程中沒有分支。EtherCAT firmware 對 EtherCAT MDevice library 觸發 cyclic interrupt（第 2 步）後，會等待 EtherCAT MDevice library 回覆 ACK（第 8 步）才繼續執行下一個動作。若使用者已註冊 cyclic callback，cyclic interrupt 會加以呼叫，如第 4 步所示。只要使用者在 cyclic callback 中讀取當前的輸入 process data、完成處理、計算出輸出 process data 並寫回，當前週期就會將輸出 process data 送往 EtherCAT 網路，滿足即時控制系統的需求。

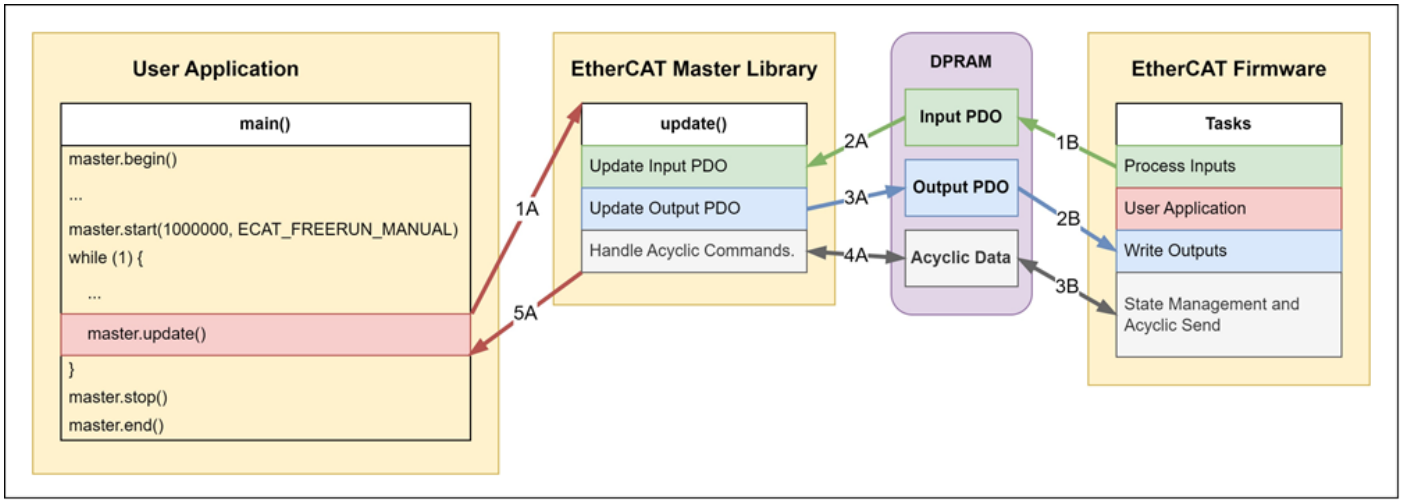
範例程式碼如下：

```
#include "Ethercat.h"

EthercatMaster master;

void CyclicCallback() {
  // ...
}

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

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

4.5.1.3 範例 3：Free Run Manual 模式

這同樣是未啟用雙系統同步的 free-run 模式。與 ECAT_FREERUN 模式的主要差異在於，本模式沒有 cyclic interrupt 來更新 process data 與處理非週期性命令，使用者必須自行呼叫 `EthercatMaster::update()` 來更新 process data 並處理非週期性命令。此外，由於本模式沒有 cyclic interrupt，cyclic callback 也不會被呼叫。如圖中編號箭頭所示，兩個系統各自獨立運作，彼此不同步。

QEC MDevice 的 Free Run Manual 模式示意圖：



範例程式碼如下：

```
#include "Ethercat.h"

EthercatMaster master;

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

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

4.5.2 SubDevice 資訊

QEC MDevice 的 EtherCAT library 提供多個函式，可取得網路上 EtherCAT SubDevice 的相關資訊，包括查詢網路上的 SubDevice 數量、依序號取得 SubDevice 的 Vendor ID、Product Code 與 Alias Address，以及利用上述資訊反查 SubDevice 編號。這些資訊可用於判斷 SubDevice 的類型，並選用適當的 EtherCAT SubDevice 類別加以掛載。

4.5.2.1 範例 1：使用 EthercatMaster 類別

使用 EthercatMaster 類別顯示 SubDevice 資訊。

範例程式碼如下：

```
#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:

}
```

4.5.2.2 範例 2：使用 EthercatDevice_Generic 類別

使用 EthercatDevice_Generic 類別顯示 SubDevice 資訊。

範例程式碼如下：

```
#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:

}
```

4.5.2.3 範例 3：使用 EthercatDevice_CiA402 類別

使用 EthercatDevice_CiA402 類別顯示 SubDevice 資訊。

範例程式碼如下：

```
#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);
```

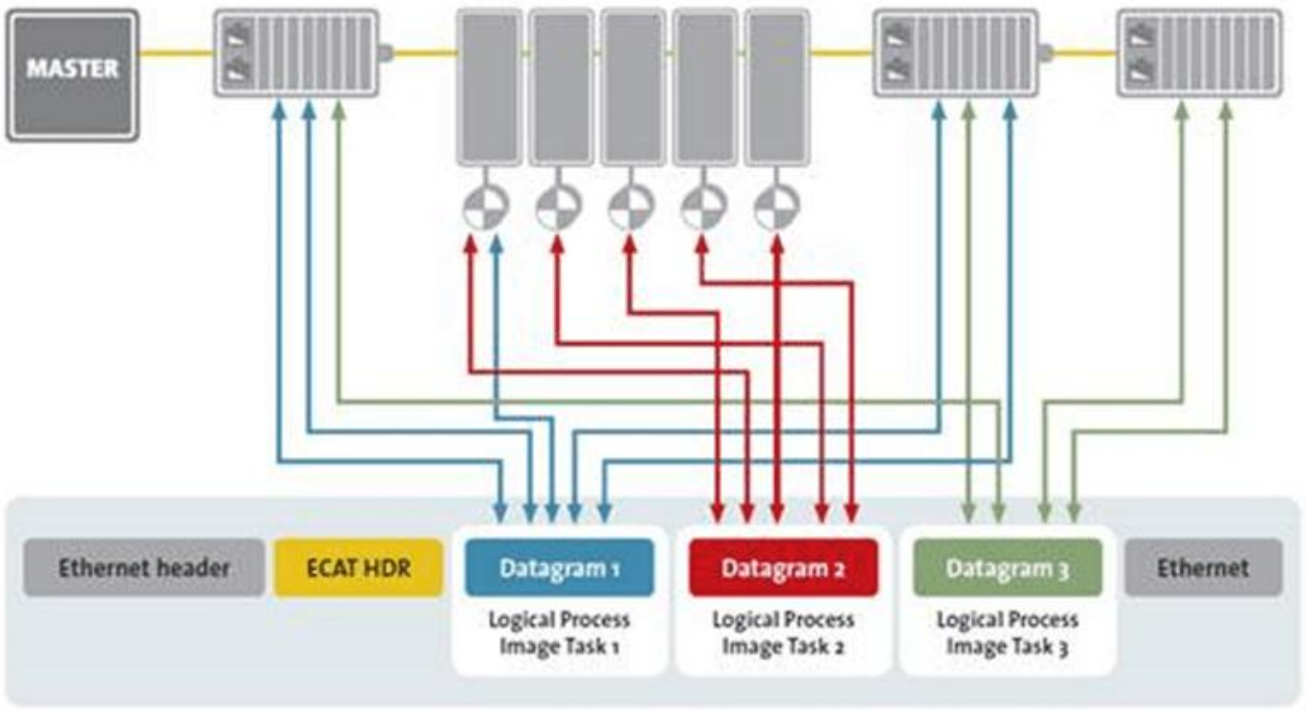
```
    Serial.print("        EoE Details: 0x");
    Serial.println(slave.getEoEDetails(), HEX);

    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:

}
```

4.5.3 Process Data Objects (PDO) 函式

Process Data 指的是 EtherCAT 網路中 MDevice 與 SubDevice 之間交換的即時通訊資料，內容涵蓋控制、監控與通訊用途的資訊。EtherCAT MDevice 會週期性地傳送 process data，以控制並監控所有 SubDevice，確保高同步性與低延遲。

EtherCAT SubDevice Controller (ESC) 中的 Fieldbus Memory Management Unit (FMMU) 可將雙埠記憶體映射至邏輯位址。所有 SubDevice 節點會檢查 EtherCAT MDevice 送出的 EtherCAT frame，比對 process data 的邏輯位址與 FMMU 中設定的位址；若比對相符，輸出的 process data 會被寫入雙埠記憶體，而輸入的 process data 則會插入 EtherCAT frame 中。

整體而言，process data 是 EtherCAT 技術的核心環節，適用於機器人控制、CNC 控制、自動化控制等領域的即時應用。



EtherCAT：網路封包與資料的交換。(資料來源：http://www.ethercat.org/)

4.5.3.1 範例 1：從 Input PDO 讀取一個 bit

使用 pdoBitRead() 從 Input PDO 讀取一個 bit。

一台 16 通道數位輸入的 EtherCAT SubDevice 具有 2 byte 的 Input PDO，每個 bit 對應一個數位輸入通道。程式會以 1 Hz 的頻率印出通道 0 與通道 9 的狀態。

B15	B14	B13	B12	B11	B10	**B9**	B8	B7	B6	B5	B4	B3	B2	B1	**B0**

範例程式碼如下：

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
    Serial.begin(115200); // Initialize Serial Monitor
    while (!Serial);      // Wait for Serial Monitor to be ready

    master.begin();       // Initialize EtherCAT Master
    slave.attach(0, master); // Attach the first EtherCAT slave to the master
    master.start();       // Start EtherCAT communication
}

void loop() {
    // Read and display the state of PDO bits
    Serial.print("Bit0 => ");
    Serial.print(slave.pdoBitRead(0)); // Read PDO bit 0
    Serial.print(", Bit9 => ");
    Serial.println(slave.pdoBitRead(9)); // Read PDO bit 9

    delay(1000); // Wait for 1 second
}
```

4.5.3.2 範例 2：從 Input PDO 讀取 byte 資料

使用 pdoRead8() 從 Input PDO 讀取資料。

一台 16 通道數位輸入的 EtherCAT SubDevice 具有 2 byte 的 Input PDO，每個 bit 對應一個數位輸入通道。程式會以 1 Hz 的頻率印出通道 0 與通道 9 的狀態。

B15	B14	B13	B12	B11	B10	**B9**	B8	B7	B6	B5	B4	B3	B2	B1	**B0**

範例程式碼如下：

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  Serial.begin(115200); // Initialize Serial Monitor
  while (!Serial);      // Wait for Serial Monitor to be ready

  master.begin();       // Initialize EtherCAT Master
  slave.attach(0, master); // Attach the first EtherCAT slave to the master
  master.start();       // Start EtherCAT communication
}

void loop() {
  // Read specific bits using pdoRead8 and print their values
  Serial.print("Bit0 => ");
  Serial.print((slave.pdoRead8(0) >> 0) & 1); // Read PDO byte 0, extract bit
0
  Serial.print(", Bit9 => ");
  Serial.println((slave.pdoRead8(1) >> 1) & 1); // Read PDO byte 1, extract
bit 9

  delay(1000); // Wait for 1 second before reading again
}
```

4.5.3.3 範例 3：將一個 bit 寫入 Output PDO

使用 `pdoBitWrite()` 將一個 bit 寫入 Output PDO。

一台 16 通道數位輸出的 EtherCAT SubDevice 具有 2 byte 的 Output PDO，每個 bit 對應一個數位輸出通道。程式會以 1 Hz 的頻率切換通道 0 與通道 9 的狀態。

B15	B14	B13	B12	B11	B10	**B9**	B8	B7	B6	B5	B4	B3	B2	B1	**B0**

範例程式碼如下：

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
  master.begin();       // Initialize EtherCAT Master
  slave.attach(0, master); // Attach the first EtherCAT slave to the master
  master.start();       // Start EtherCAT communication
}

void loop() {
  // Set Bit 0 and Bit 9 to 1
  slave.pdoBitWrite(0, 1); // Write 1 to Bit 0
  slave.pdoBitWrite(9, 1); // Write 1 to Bit 9
  delay(1000);             // Wait for 1 second

  // Set Bit 0 and Bit 9 to 0
  slave.pdoBitWrite(0, 0); // Write 0 to Bit 0
  slave.pdoBitWrite(9, 0); // Write 0 to Bit 9
  delay(1000);             // Wait for 1 second
}
```

4.5.3.4 範例 4：將 byte 資料寫入 Output PDO

使用 `pdoWrite8()` 將資料寫入 Output PDO。

一台 16 通道數位輸出的 EtherCAT SubDevice 具有 2 byte 的 Output PDO，每個 bit 對應一個數位輸出通道。程式會以 1 Hz 的頻率切換通道 0 與通道 9 的狀態。

B15	B14	B13	B12	B11	B10	**B9**	B8	B7	B6	B5	B4	B3	B2	B1	**B0**

範例程式碼如下：

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

void setup() {
    master.begin();       // Initialize EtherCAT Master
    slave.attach(0, master);
    master.start();       // Start EtherCAT communication
}

void loop() {
    // Write 0x01 to PDO byte 0 and 0x02 to PDO byte 1
    slave.pdoWrite8(0, 0x01); // Set byte 0 to 0x01
    slave.pdoWrite8(1, 0x02); // Set byte 1 to 0x02
    delay(1000);              // Wait for 1 second

    // Write 0x00 to PDO byte 0 and byte 1
    slave.pdoWrite8(0, 0x00); // Set byte 0 to 0x00
    slave.pdoWrite8(1, 0x00); // Set byte 1 to 0x00
    delay(1000);              // Wait for 1 second
}
```

4.5.4 CANopen over EtherCAT (CoE) 功能

CoE（CAN application over EtherCAT）是架構在 EtherCAT 網路之上的 CANopen 協定，可在 EtherCAT 網路上以 CANopen 協定進行通訊。Object Dictionary 中存放參數、應用資料，以及 process data 介面與應用資料之間的對應資訊（PDO mapping），其項目可透過 Service Data Objects (SDO) 存取。

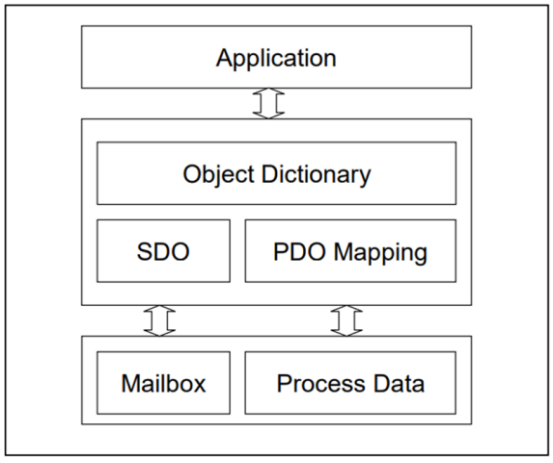
CANopen 是架構在 Controller Area Network (CAN) bus 之上的高階通訊協定，常用於工業應用中控制系統與裝置之間的通訊。它定義了一組通訊物件、資料型別與網路管理功能，以利裝置之間的資料交換、組態設定與控制。

CANopen 協定包含下列面向：

- Object Dictionary：定義裝置之間交換的所有資料物件與參數。Object Dictionary 涵蓋變數、參數、事件、功能等多種類型的物件。

- PDO (Process Data Object)：用於即時資料傳輸。PDO 讓裝置之間能以固定週期或事件觸發的方式傳送資料，實現即時控制與資料交換。

- SDO (Service Data Object)：用於設定與管理裝置參數。SDO 提供讀取、寫入與參數組態等功能，讓裝置能動態交換組態資訊。



SDO 服務主要包含兩類命令：SDO command 用於存取 Object Dictionary 中的物件，SDO information command 則用於取得這些物件的詳細資訊。

4.5.4.1 範例 1：SDO Upload

使用 `sdoUpload8()` 執行 SDO Upload。

`sdoUpload16()`、`sdoUpload32()` 與 `sdoUpload64()` 的用法與 `sdoUpload8()` 相同，差別僅在於回傳值的型別。

範例程式碼如下：

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

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

  master.begin();       // Initialize EtherCAT Master
  slave.attach(0, master); // Attach the first EtherCAT slave to the master

  // Read and print the value of SDO 0x1C12.0
  Serial.print("1C12h.0 => ");
  Serial.println(slave.sdoUpload8(0x1C12, 0x00)); // Upload 8-bit SDO data
from 0x1C12.0

  // Read and print the value of SDO 0x1C13.0
  Serial.print("1C13h.0 => ");
  Serial.println(slave.sdoUpload8(0x1C13, 0x00)); // Upload 8-bit SDO data
from 0x1C13.0
}

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

}
```

4.5.4.2 範例 2：帶 abort code 的 SDO Upload

使用 `sdoUpload()` 執行帶 abort code 的 SDO Upload。

對一個不存在的物件發出 SDO Upload 命令以讀取數值，預期會取得 0x06020000 的 abort code。關於 abort code 的更多資訊，請參閱 SDO Abort Code。

範例程式碼如下：

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

uint32_t abortcode;   // Variable to store the abort code
uint8_t value;        // Variable to store the uploaded value

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

  master.begin();       // Initialize EtherCAT Master
  slave.attach(0, master); // Attach the first EtherCAT slave to the master

  // Attempt to upload SDO data
  if (slave.sdoUpload(0xFFFF, 0xFF, &value, sizeof(value), &abortcode) ==
ECAT_ERR_DEVICE_COE_ERROR) {
    Serial.print("Abort Code: 0x");
    Serial.println(abortcode, HEX); // Print the abort code in hexadecimal
format
  }
}

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

}
```

4.5.4.3 範例 3：SDO Download

使用 `sdoDownload8()` 執行 SDO Download。

`sdoDownload16()`、`sdoDownload32()` 與 `sdoDownload64()` 的用法與 `sdoDownload8()` 相同，差別僅在於輸入參數的型別。

範例程式碼如下：

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

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

  slave.sdoDownload8(0x1C12, 0x00, 0);
  slave.sdoDownload8(0x1C13, 0x00, 0);
}

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

}
```

4.5.4.4 範例 4：帶 abort code 的 SDO Download

使用 `sdoDownload()` 執行帶 abort code 的 SDO Download。

對一個不存在的物件發出 SDO Download 命令以寫入數值，預期會取得 0x06020000 的 abort code。關於 abort code 的更多資訊，請參閱 SDO Abort Code。

範例程式碼如下：

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

uint32_t abortcode;
uint8_t value;

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

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

  if (slave.sdoDownload(0xFFFF, 0xFF, &value, sizeof(value), &abortcode) ==
ECAT_ERR_DEVICE_COE_ERROR) {
    Serial.print("Abort Code: 0x");
    Serial.println(abortcode, HEX); // Print the abort code in hexadecimal
format
  }
}

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

}
```

4.5.4.5 範例 5：印出 PDO mapping 組態

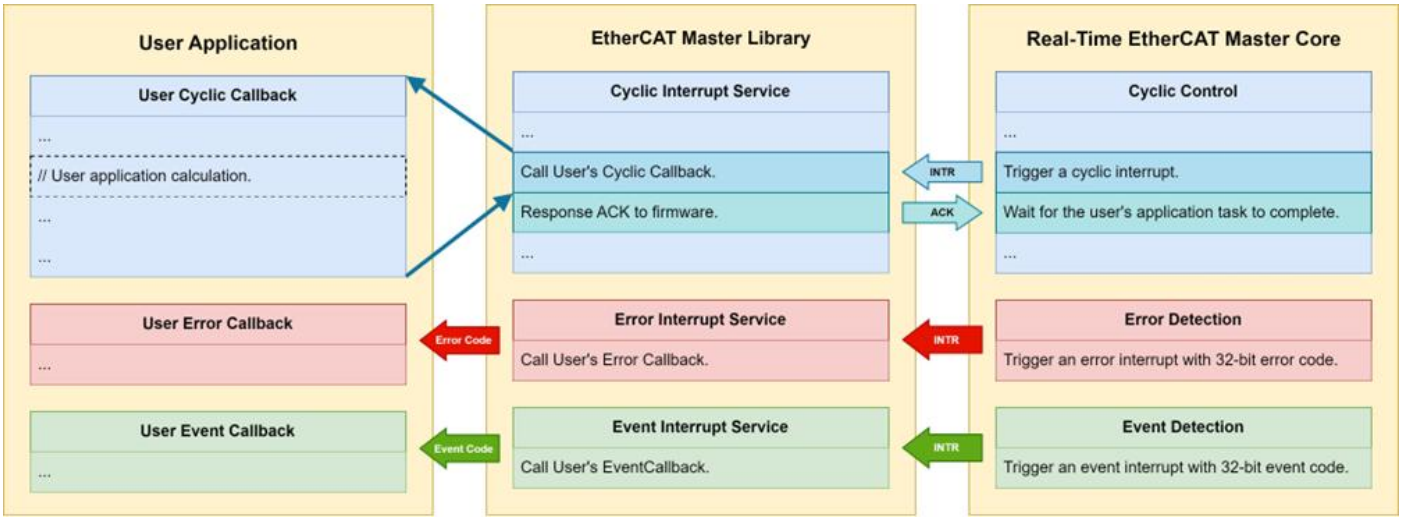
印出 PDO mapping 組態。

範例程式碼如下：

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

uint8_t assign_nr, mapping_nr;
uint16_t mapping;
uint32_t entry;

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

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

  // Process RxPDO Assignments
  assign_nr = slave.sdoUpload8(0x1C12, 0x00);
  for (int m = 0; m < assign_nr; m++) {
    mapping = slave.sdoUpload16(0x1C12, m + 1);
    Serial.print("  RxPDO");
    Serial.print(m + 1);
    Serial.print(" (");
    Serial.print(mapping, HEX);
    Serial.println("h)");

    mapping_nr = slave.sdoUpload8(mapping, 0x00);
    for (int n = 0; n < mapping_nr; n++) {
      entry = slave.sdoUpload32(mapping, n + 1);
      Serial.print("    ");
      Serial.print(entry, HEX);
      Serial.println("h");
    }
  }

  // Process TxPDO Assignments
  assign_nr = slave.sdoUpload8(0x1C13, 0x00);
  for (int m = 0; m < assign_nr; m++) {
    mapping = slave.sdoUpload16(0x1C13, m + 1);
    Serial.print("  TxPDO");
    Serial.print(m + 1);
    Serial.print(" (");
    Serial.print(mapping, HEX);
    Serial.println("h)");

    mapping_nr = slave.sdoUpload8(mapping, 0x00);
    for (int n = 0; n < mapping_nr; n++) {
      entry = slave.sdoUpload32(mapping, n + 1);
      Serial.print("    ");
```

```
      Serial.print(entry, HEX);
      Serial.println("h");
    }
  }
}

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

}
```

4.5.5 Cyclic Callback 函式



本 library 提供下列三種 callback：

- **Cyclic Callback**
  Cyclic Callback 的用途，是讓使用者能實作運動控制、CNC 控制、機器人控制等週期性控制系統。Real-Time EtherCAT MDevice Core 會依設定的 cycle time 對 EtherCAT MDevice Library 觸發 cyclic interrupt，並等待 ACK 以確保 process data 同步。若使用者已註冊 Cyclic Callback，系統便會加以呼叫，以達成週期性控制。

- **Error Callback**
  當 Real-Time EtherCAT MDevice Core 偵測到錯誤時，會觸發 error interrupt，並將一組 32-bit 的 error code 傳給 EtherCAT MDevice Library。若使用者已註冊 error callback，系統便會呼叫該 callback，以告知使用者具體的錯誤內容。

  Error Callback 支援的 error code 如下：

定義	代碼	說明
ECAT_ERR_WKC_SINGLE_FAULT	2000001	發生 working counter 錯誤。
ECAT_ERR_WKC_MULTIPLE_FAULTS	2000002	發生多次 working counter 錯誤。
ECAT_ERR_SINGLE_LOST_FRAME	2000003	發生 frame 遺失。
ECAT_ERR_MULTIPLE_LOST_FRAMES	2000004	發生多次 frame 遺失。
ECAT_ERR_CABLE_BROKEN	2000007	線路斷線。
ECAT_ERR_WAIT_ACK_TIMEOUT	2001000	Firmware 等待 cyclic interrupt ACK 逾時。

- **Event Callback**

  當 Real-Time EtherCAT MDevice Core 偵測到事件時，會觸發 event interrupt，並將一組 32-bit 的 event code 傳給 EtherCAT MDevice Library。若使用者已註冊 event callback，系統便會呼叫該 callback，以告知使用者具體的事件內容。Event Callback 支援的 event code 如下：

定義	代碼	說明
ECAT_EVT_STATE_CHANGED	1000001	MDevice 的 EtherCAT 狀態已變更。
ECAT_EVT_CABLE_RECONNECTED	1000002	線路已重新連接。

4.5.5.1 範例 1：Cyclic callback

一台 16 通道數位輸出的 EtherCAT SubDevice 具有 2 byte 的 Output PDO，每個 bit 對應一個數位輸出通道。程式會以 1 Hz 的頻率切換通道 0 與通道 9 的狀態。

B15	B14	B13	B12	B11	B10	**B9**	B8	B7	B6	B5	B4	B3	B2	B1	**B0**

範例程式碼如下：

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;
int toggle = 0;       // Toggle variable
int cycle_count = 0;  // Cycle count variable

void myCallback() {
  if (++cycle_count < 1000) // Increment and check the cycle count
    return;
  cycle_count = 0;         // Reset cycle count

  toggle = !toggle;        // Toggle the state
  slave.pdoBitWrite(0, toggle); // Write the toggle value to Bit 0
  slave.pdoBitWrite(9, toggle); // Write the toggle value to Bit 9
}

void setup() {
  master.begin();      // Initialize EtherCAT Master
  slave.attach(0, master); // Attach the first EtherCAT slave to the master

  master.attachCyclicCallback(myCallback); // Attach cyclic callback
  master.start(1000000); // Start EtherCAT Master with 1 ms cycle time
}

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

}
```

4.5.5.2 範例 2：Error callback

每秒印出各類錯誤的發生次數。

範例程式碼如下：

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

// 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);
  while (!Serial);

  master.attachErrorCallback(myErroCallback); // Attach error callback
  master.begin();       // Initialize EtherCAT Master
  master.start();       // Start EtherCAT communication
}

void loop() {
  // Print error counts to Serial Monitor
```

```
  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); // Wait 1 second before the next print
}
```

4.5.5.3 範例 3：Event callback

每秒印出各類事件的發生次數。

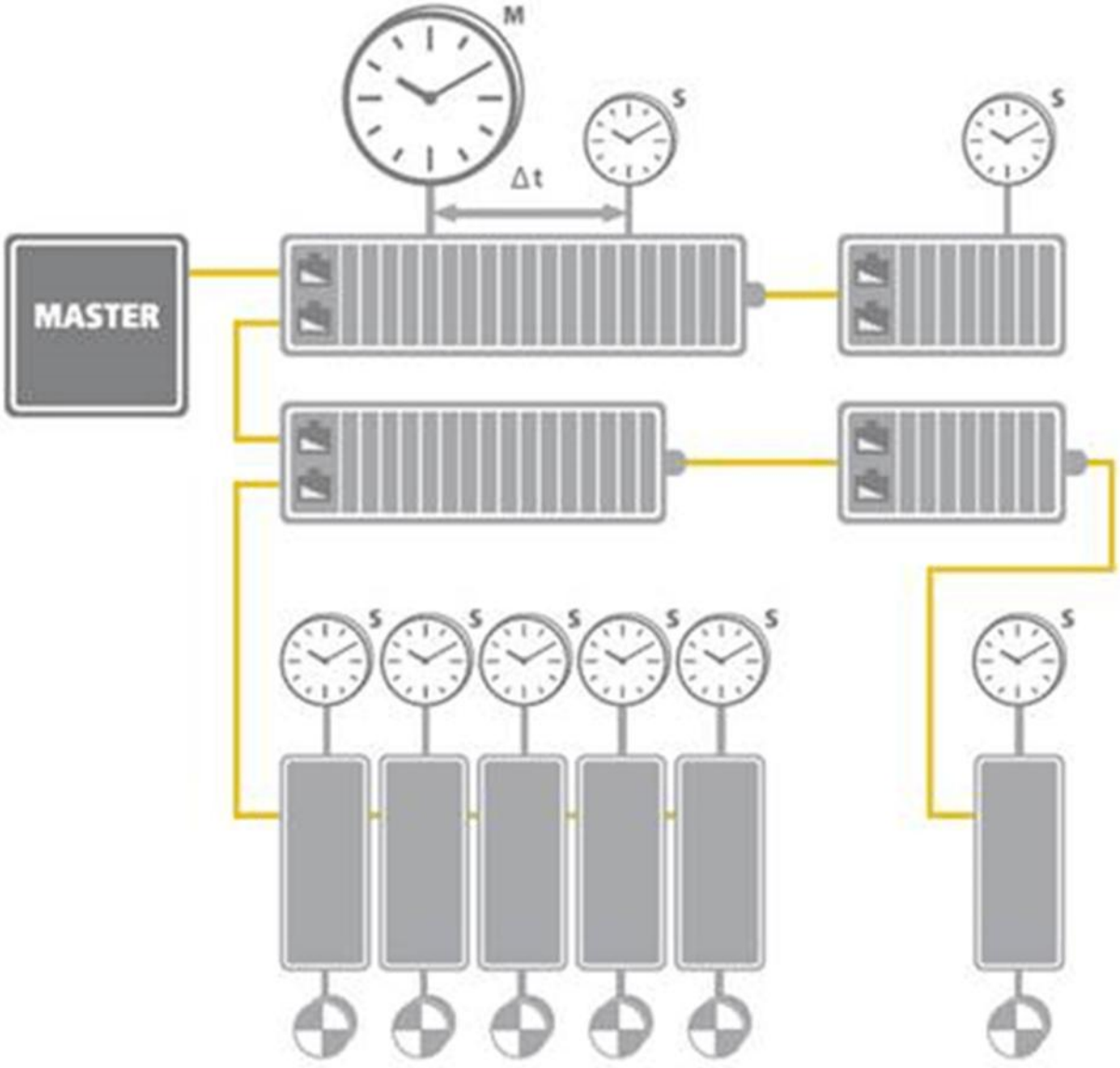
範例程式碼如下：

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic slave;

// 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);
  while (!Serial);

  master.attachEventCallback(myEventCallback); // Attach event callback
  master.begin();       // Initialize EtherCAT Master
  master.start();       // Start EtherCAT communication
}

void loop() {
  // Print event counts to Serial Monitor
  Serial.print("ECAT_EVT_STATE_CHANGED    = ");
  Serial.println(state_changed_cnt);
  Serial.print("ECAT_EVT_CABLE_RECONNECTED = ");
  Serial.println(cable_reconnected_cnt);

  delay(1000); // Wait 1 second before the next update
}
```

4.5.6 Distributed Clock (DC) 組態函式

在空間上分散、卻必須同時動作的製程應用中，精確同步格外重要。例如多軸伺服需執行協調動作的應用即屬此類。

與完全同步式通訊（其品質會立即受通訊錯誤影響）相比，分散式同步時脈對通訊系統的 jitter 具有相當高的容忍度。因此 EtherCAT 用於節點同步的解決方案即是以這類 Distributed Clock (DC) 為基礎。

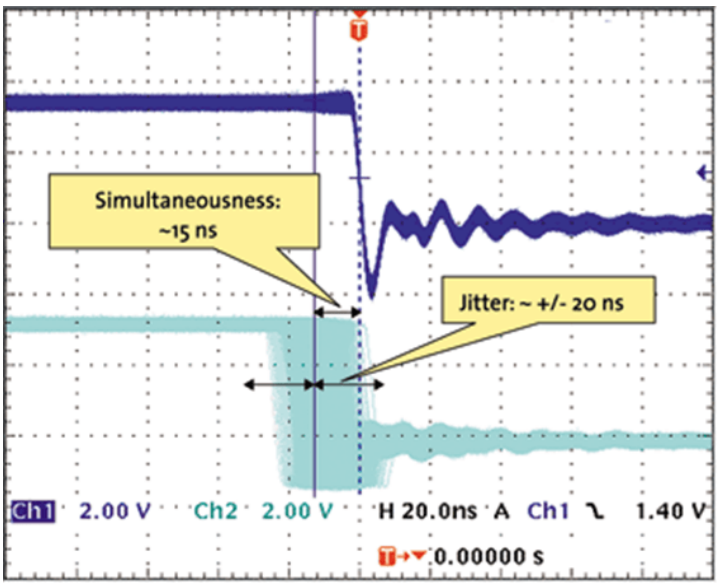


EtherCAT：Distributed Clock (DC) 示意圖。(資料來源：http://www.ethercat.org/)

EtherCAT 的同步機制以 IEEE-1588 Precision Clock Synchronization Protocol 為基礎，並延伸定義出所謂的 Distributed Clock (DC)。簡單來說，每一顆 EtherCAT ESC 都維護一組硬體時脈，最小時間間隔為 1 奈秒（共 64 bit）。EtherCAT SubDevice 所維護的時間稱為 Local system time。

透過精準的網路時間同步機制與動態時間補償機制（*1），EtherCAT DC 技術可確保各 EtherCAT SubDevice 的 local system time 之間的時間差維持在 +/- 20 奈秒以內。下圖為兩台 SubDevice 輸出數位訊號的示波器畫面，可看出兩台 EtherCAT SubDevice 的 I/O 訊號時間差約為 20 奈秒。

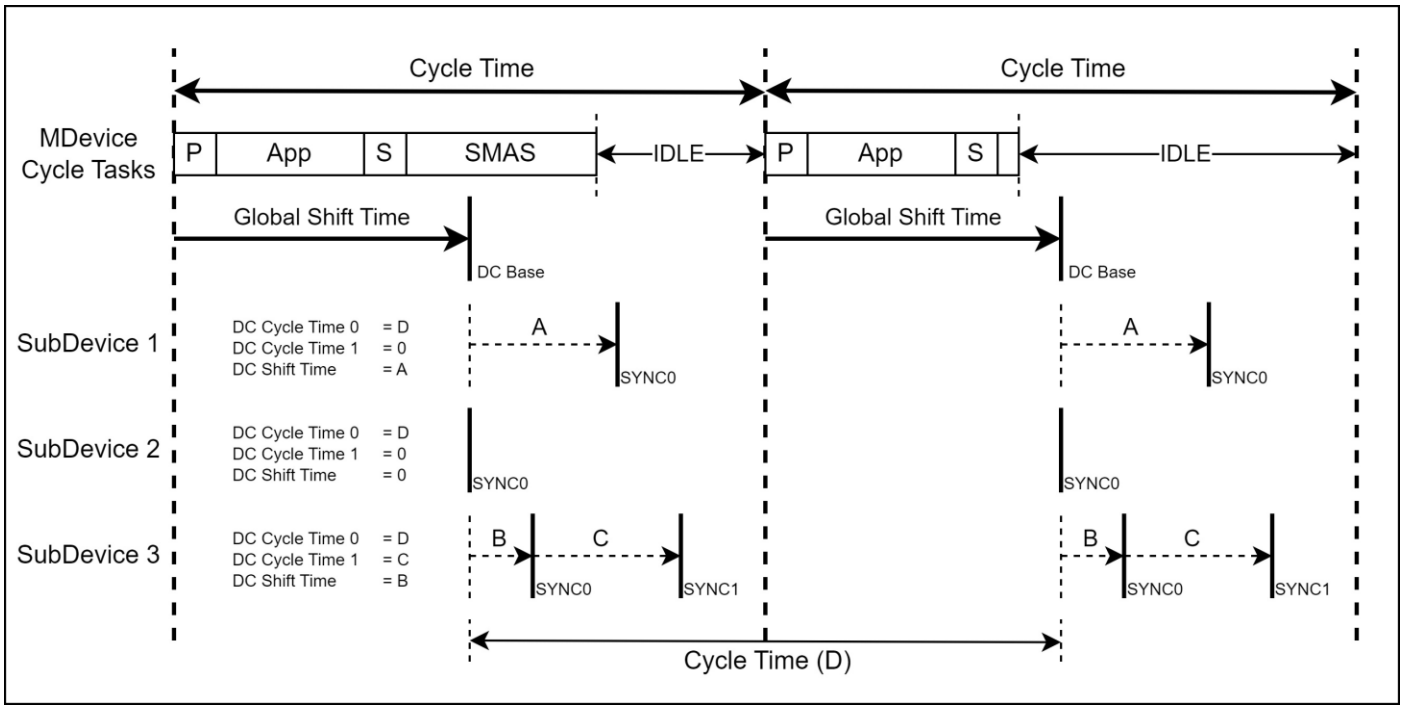
（*1）請參閱 EtherCAT 標準文件 ETG1000.4



同步性與同時性：兩台分散裝置的示波器畫面，兩者之間相隔 300 個節點與 120 公尺的線材。（資料來源：http://www.ethercat.org/）

設定 EtherCAT SubDevice 的 DC 參數。此函式可設定三個 DC 參數：

- ***DC Cycle Time 0*** 用於設定 SYNC0 訊號的週期時間，通常與 EtherCAT 通訊的 cycle time 一致。
- ***DC Cycle Time 1*** 用於設定 SYNC1 訊號的週期時間，也就是 SYNC0 脈波之後所定義的延遲。此參數為選用。
- ***DC Shift Time*** 用於設定 SYNC0 訊號相對於 DC Base 的偏移量。



4.5.6.1 範例 1：啟用 DC 同步

使用 EthercatDevice_Generic 類別對 CiA 402 的 EtherCAT SubDevice 實作位置控制。CiA 402 的控制模式設定為 cyclic synchronous position mode，並啟用 DC 同步以取得精準的時序。

預設的 PDO mapping 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			

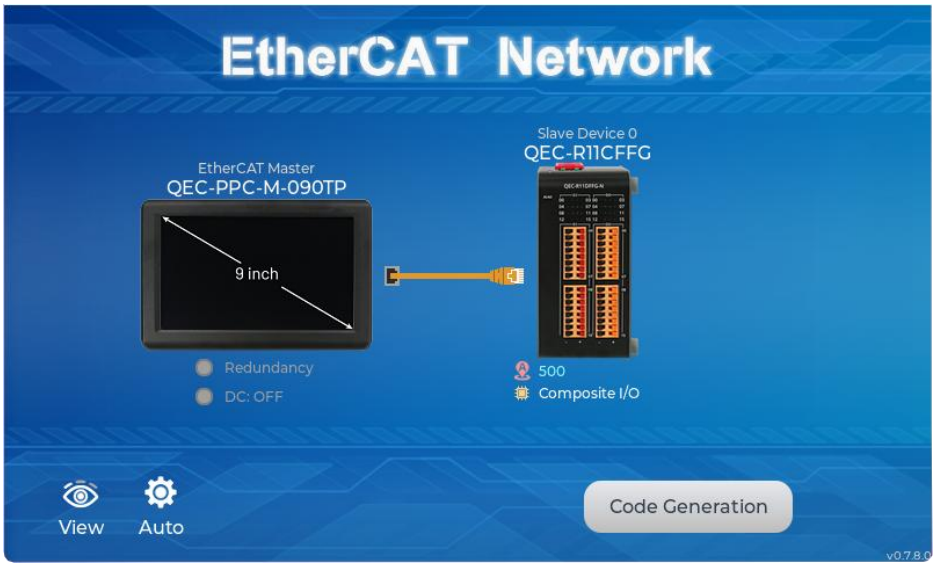
範例程式碼如下：

```
#include <Ethercat.h>

EthercatMaster master;
EthercatDevice_Generic slave;

uint32_t position = 0;

// Cyclic callback function
void myCyclicCallback() {
  // Check if the drive is in the correct state
  if ((slave.pdoRead8(0) & 0x6F) != 0x27)
    return;

  // Increment and write the new position
  slave.pdoWrite32(2, position += 1000);
}

void setup() {
  master.begin();       // Initialize EtherCAT Master

  slave.attach(0, master);   // Attach the first EtherCAT slave
  slave.setDc(1000000);      // Set Distributed Clock synchronization to 1 ms
  slave.sdoDownload8(0x6060, 0x00, 8); // Set operation mode to CSP (Cyclic
Synchronous Position)

  master.attachCyclicCallback(myCyclicCallback); // Attach the cyclic
callback
  master.start(1000000, ECAT_SYNC); // Start EtherCAT Master with 1 ms cycle
time and synchronization

  // Initialize position and control word
  slave.pdoWrite32(2, position = slave.pdoRead32(2)); // Set initial position
  slave.pdoWrite8(0, 0x80); // Reset fault
  delay(1000);
  slave.pdoWrite8(0, 0x06); // Switch to "Shutdown" state
  delay(1000);
  slave.pdoWrite8(0, 0x07); // Switch to "Switch On" state
```

```
  delay(1000);
  slave.pdoWrite8(0, 0x0F); // Switch to "Operation Enable" state
  delay(1000);
}

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

}
```

4.5.7 86EVA：EtherCAT 組態工具

86EVA 是一套圖形化的 EtherCAT 組態工具，以 86Duino IDE 中的 EtherCAT Library 為基礎，也是 86Duino 的開發套件之一。使用者可用它快速完成 EtherCAT 網路組態並開始撰寫程式。



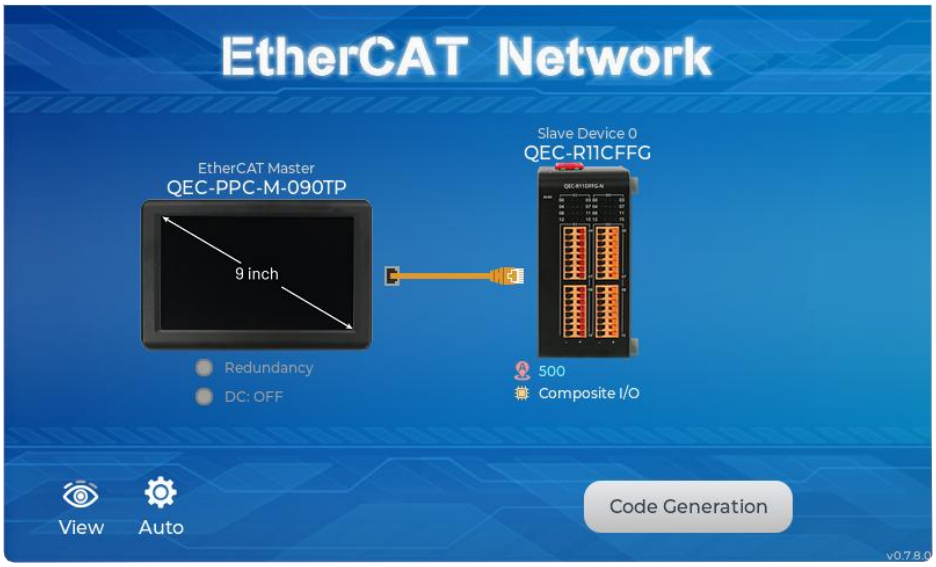
以下關於 86EVA 的說明將聚焦於 QEC EtherCAT MDevice 與 SubDevice，功能包括：

1. 自動產生 Arduino 語言程式碼（透過 EtherCAT-Based Virtual Arduino）
2. 自動掃描網路上的裝置。
3. EtherCAT MDevice 設定：
    - 設定 MDevice 物件名稱
    - 設定 Cycle Time
    - 設定 Redundancy 選項
    - 選用 ENI 檔案
4. EtherCAT SubDevice 設定：
    - 設定 SubDevice 物件名稱
    - 設定 SubDevice Alias Address
    - 可設定 SubDevice 的 I/O Mapping
    - 顯示次要裝置資訊
    - 檢視內部資訊

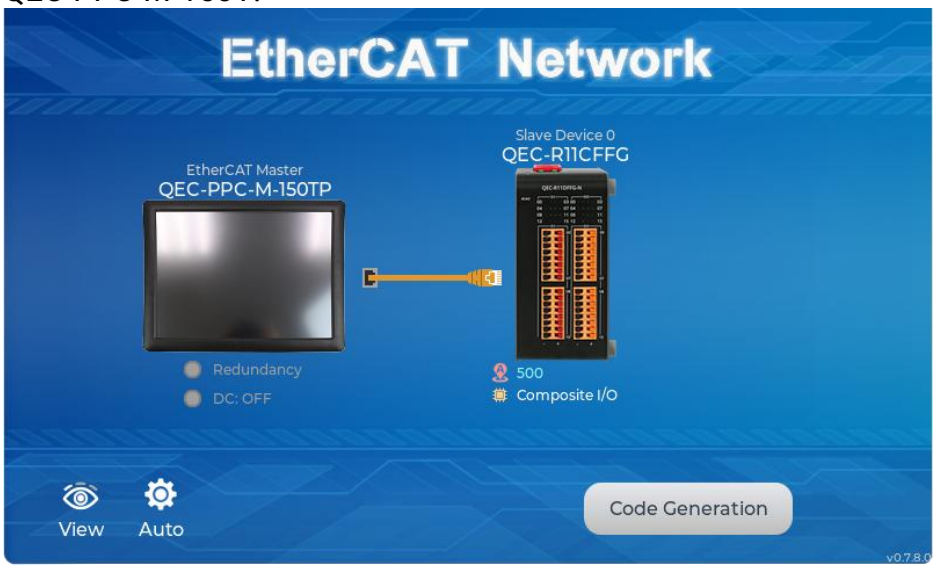
86Duino IDE 開發環境專為 QEC MDevice 設計，涵蓋 **QEC-M-01**、**QEC-M-02**、**QEC-M-070T**、**QEC-PPC-M-090T**、**QEC-M-090T**、**QEC-PPC-M-104T**、**QEC-PPC-M-150T** 與 **QEC-M-150T**。

在 86EVA 中完成連線與掃描後，使用者即可看到 EtherCAT MDevice 的資訊與產品圖示。

- QEC-PPC-M-090T:



- QEC-PPC-M-150T:



本節以 **QEC-PPC-M-090T** 為例說明。

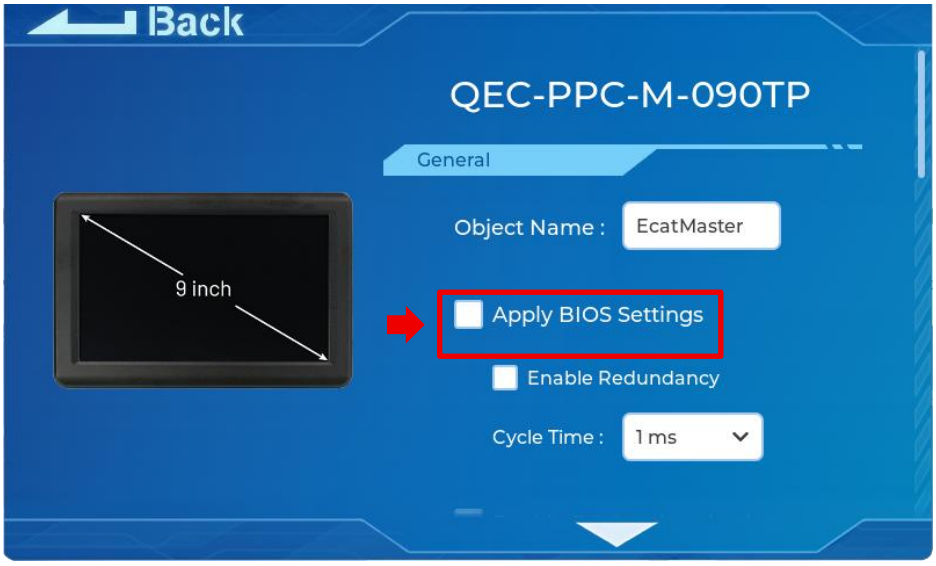
在 QEC-PPC-M-090T 的圖示上按兩下，即可檢視 MDevice 的參數設定。



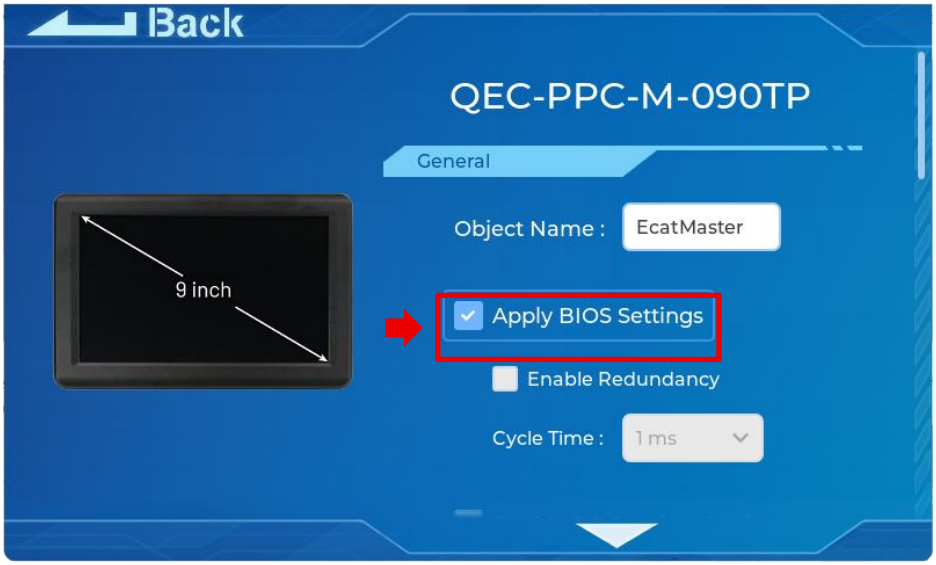
您可以設定 EtherCAT MDevice 物件名稱、套用 BIOS 設定、啟用 EtherCAT Cable Redundancy 選項、設定 EtherCAT Cycle Time 與 DC 同步功能，並選擇 ENI 檔案。

關於 86EVA 工具的詳細資訊，請參閱 86EVA 使用手冊。

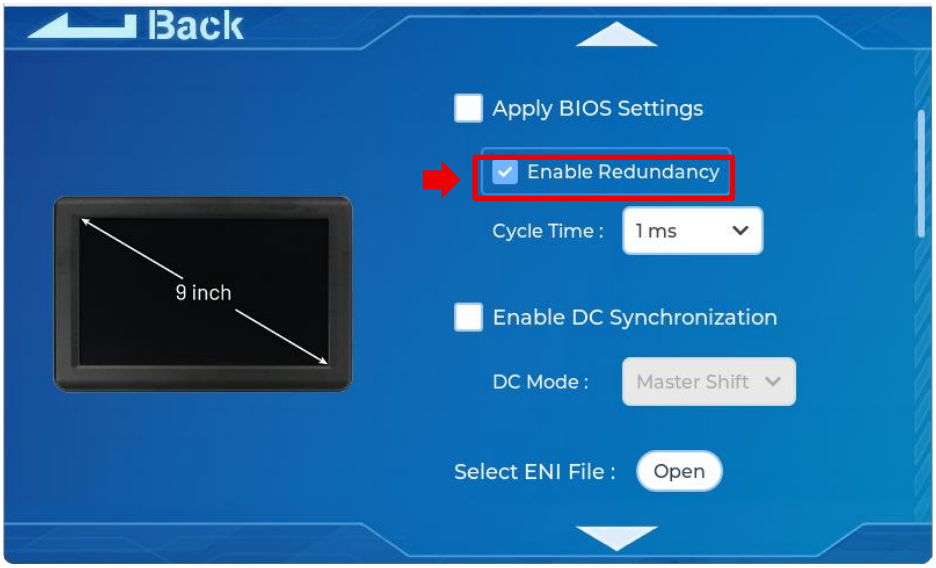
在「**Apply BIOS Settings**」中，您可依照 4.6.3 EtherCAT 頁面的說明，以預設值設定 EtherCAT 相關項目。



若使用者勾選 「**Apply BIOS Settings**」，則勾選方塊 「**Enable Redundancy**」與下拉選單 「**Cycle Time**」將無法使用。



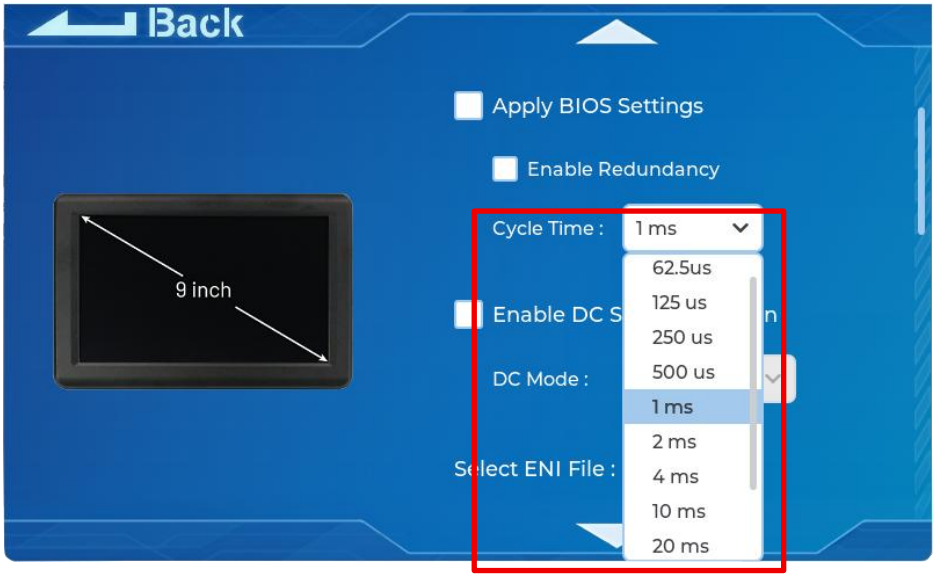
若不使用 BIOS 設定，使用者可自行勾選「**Enable Redundancy**」功能。



此功能會在 MDevice 與 SubDevice 之間建立環狀拓撲，確保在線材或裝置故障時 EtherCAT 通訊仍能持續運作。

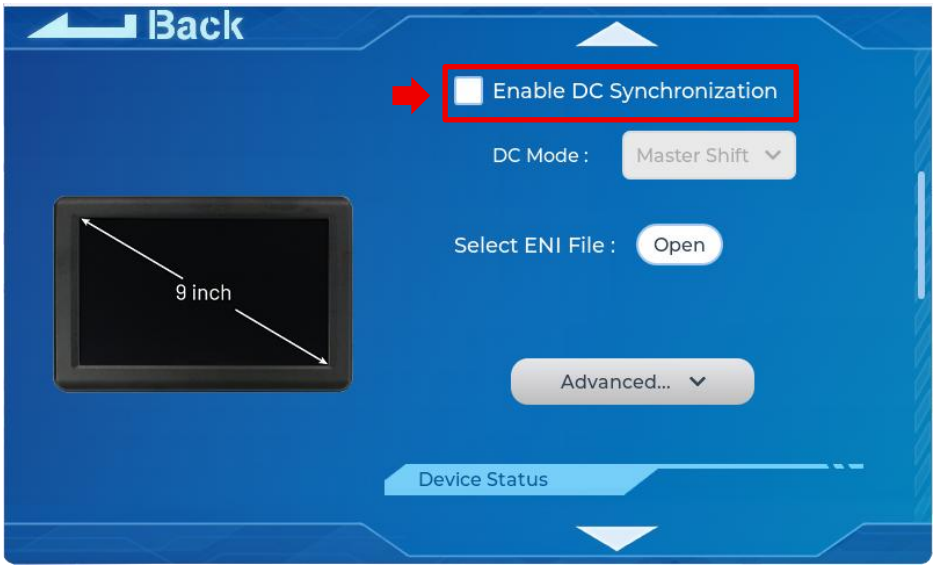
***注意：** 若啟用 Redundancy 選項，必須實際接上第二條 EtherCAT 網路線以形成環狀迴路。未接上備援線材將導致 EtherCAT State Machine 無法切換至 OPERATIONAL 狀態，與 SubDevice 的通訊也會失敗。

若不使用 BIOS 設定，使用者可由下拉選單 「**Cycle time**」選擇 EtherCAT 的 cycle time，可選範圍為 62.5 µs 至 20 ms。

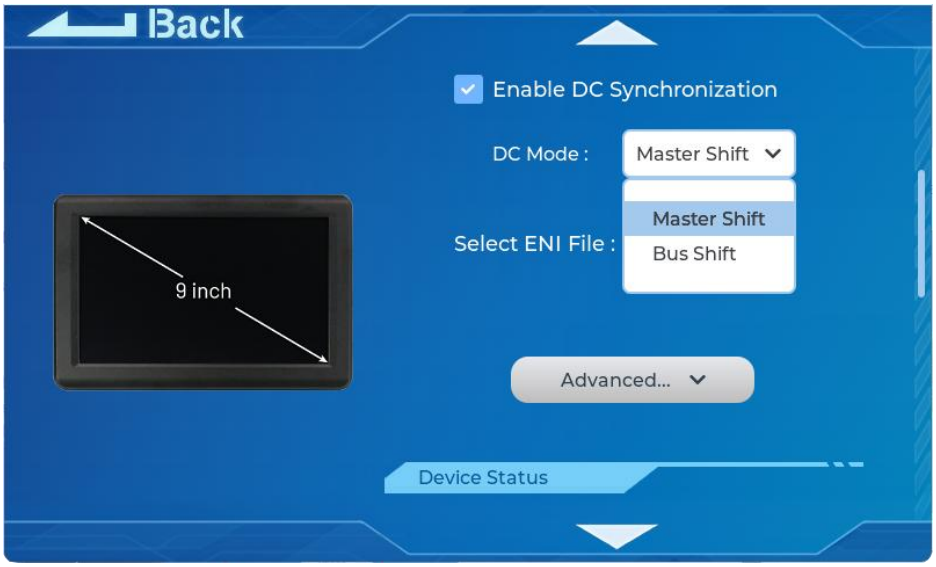


***注意：** cycle time 需配合 SubDevice 的反應時間設定。

在 「**Enable DC Synchronization**」中，您可為目前 EtherCAT 網路上所有支援且能通過 86EVA 驗證的 EtherCAT SubDevice 啟用 DC 功能。

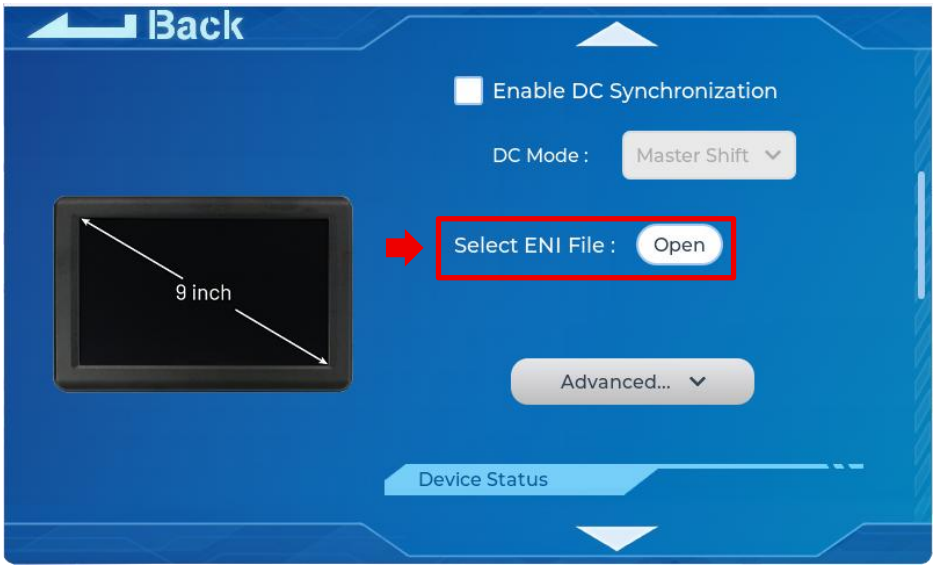


使用者亦可由下拉選單 「**DC Mode**」選擇 DC 模式，選項為「**Bus Shift**」與「**Master Shift**」。



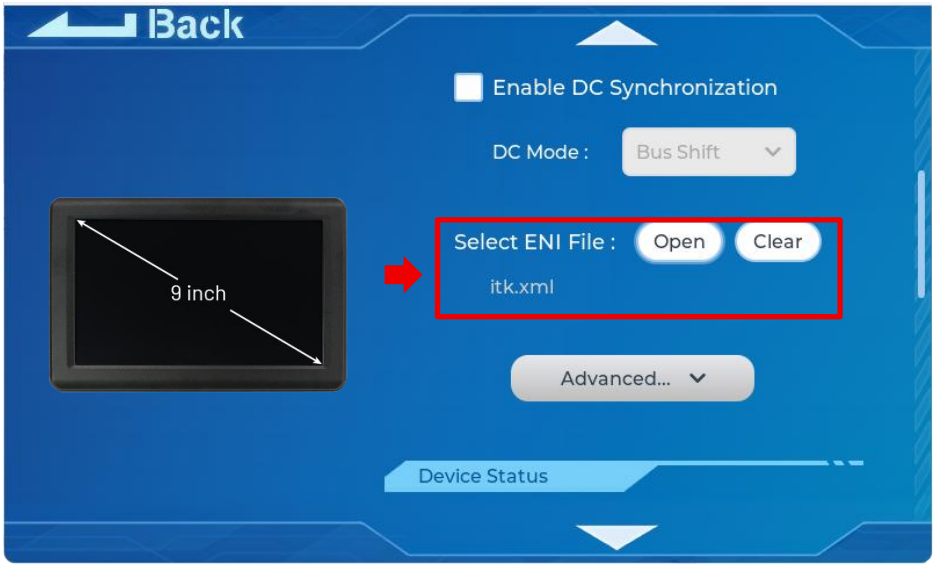
***注意：** DC 功能與模式需依 SubDevice 的規格設定。

在 「**Select ENI File**」中，使用者可上傳 ENI 檔案。關於 ENI 檔案的更多資訊，可參閱 Beckhoff Information System - English。



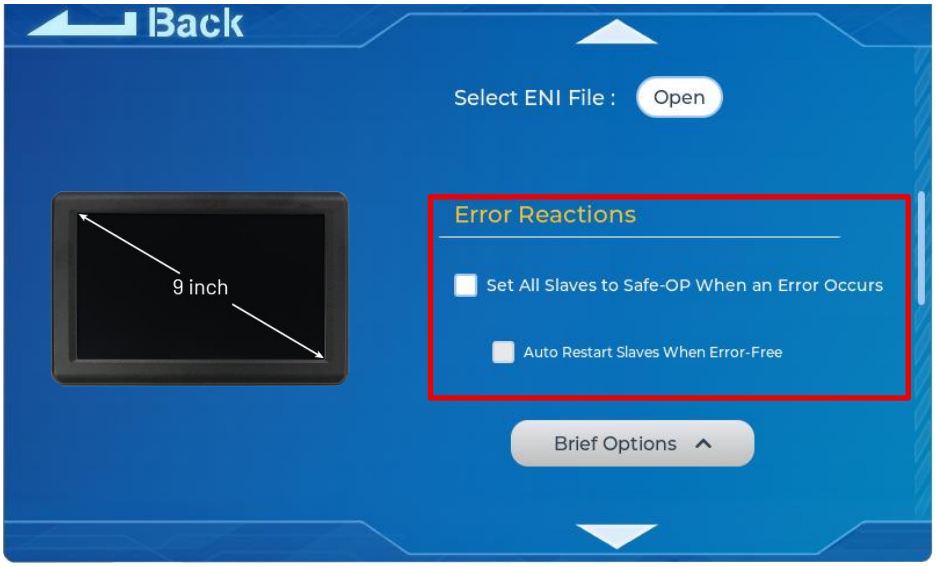
關於如何將 ENI 匯入 86EVA，請參閱 Ch. 4.5.8 將 ENI 匯入 QEC MDevice。

上傳 ENI 檔案後，「**Select ENI file**」標籤下方即會顯示檔案名稱。



點選「**Clear**」按鈕即可刪除已匯入的 ENI 檔案。

在 「**Advanced**」區段中，您可設定 EtherCAT 網路的 Error Reactions。這些設定有助於在發生通訊錯誤時提升系統穩定性與故障回復能力。



Error Reactions 下有兩個選項：

1. **Set All Slaves to Safe-OP When an Error Occurs**
   ：啟用後，只要任一 SubDevice 回報錯誤或 abort code，此選項就會將所有 EtherCAT SubDevice 強制切換至 Safe-Operational (Safe-OP) 狀態，讓執行期發生故障時有一致且可預期的退回行為。
2. **Auto Restart Slaves When Error-Free**
   ：此選項僅在前一項設定啟用時才能勾選。勾選後，一旦錯誤狀況排除，EtherCAT MDevice 會自動嘗試恢復通訊，並將所有 SubDevice 還原至先前的狀態。

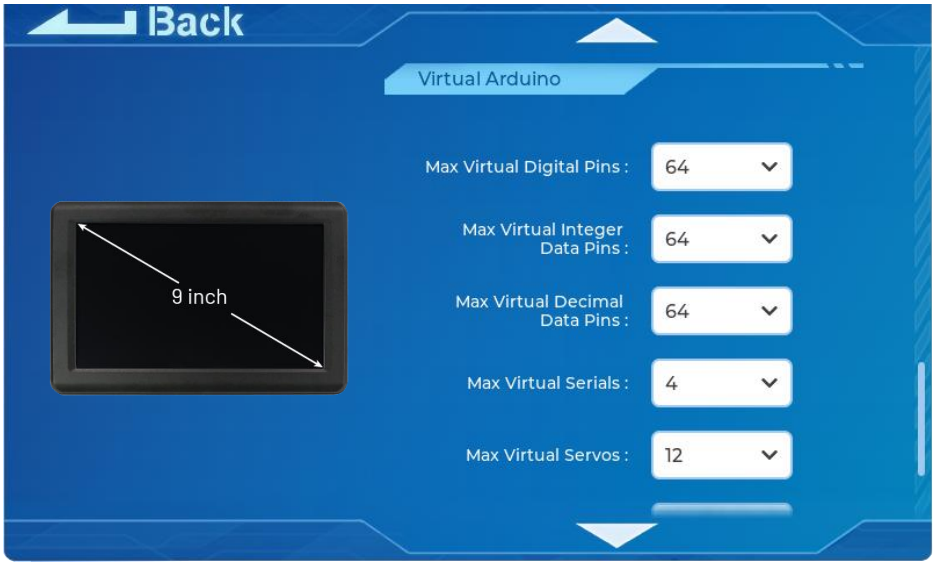
您也可以點選頁面下方的「**Brief Options**」按鈕，收合 Error Reactions 區段以簡化畫面。



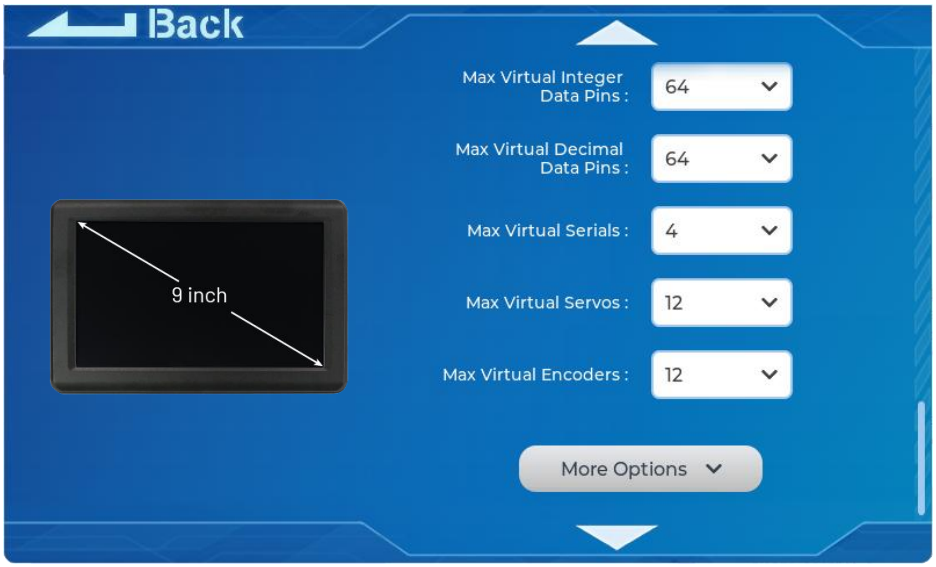
您也可以在 「**Device Status**」區域查看電壓、電流與系統溫度。



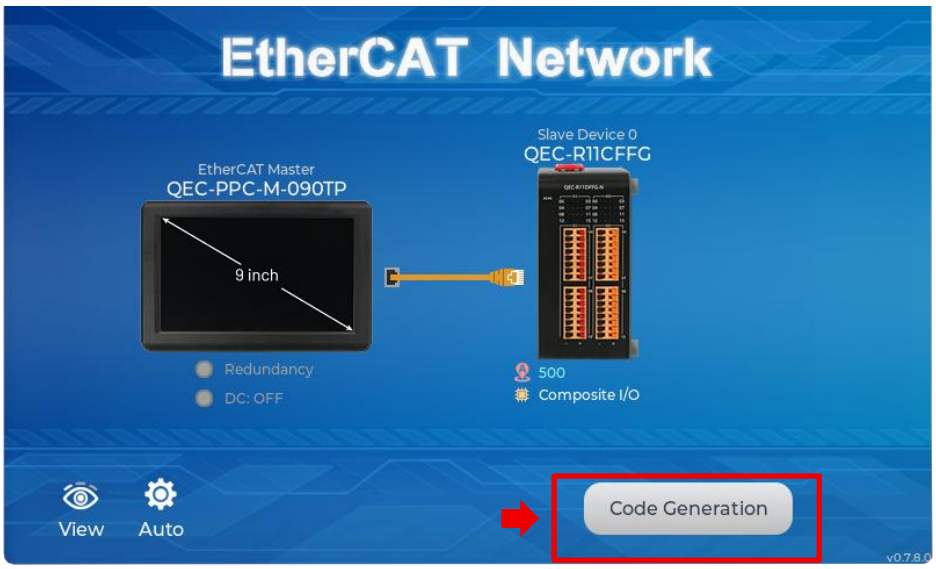
使用者亦可設定 **Virtual Arduino** 功能的最大數量。



點選「**More Options**」按鈕可顯示更多 Virtual Arduino 物件設定。



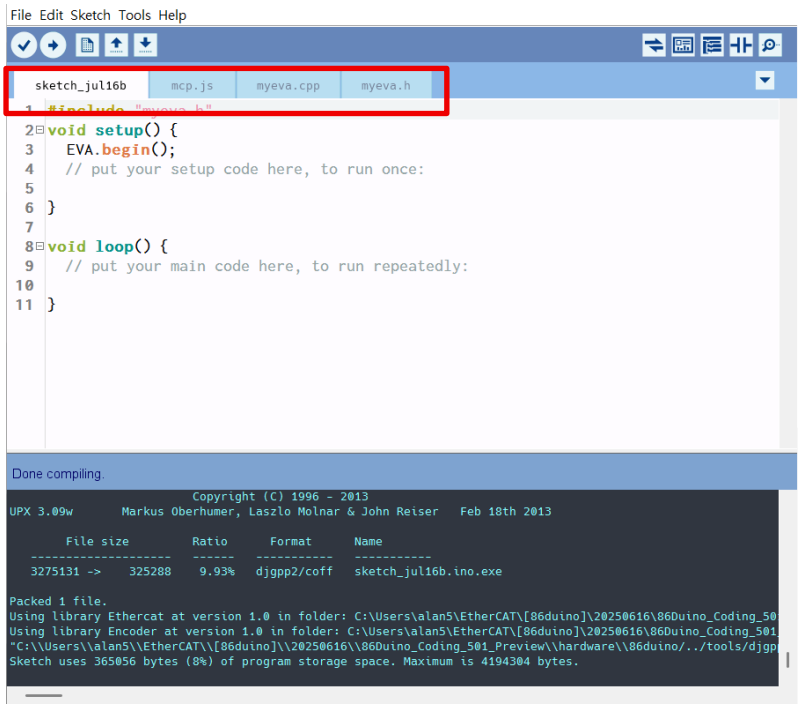
完成 EtherCAT 組態設定後，回到主畫面並按下右下角的 「**Code Generation**」按鈕。



完成後，雙擊「**OK**」按鈕即可關閉 86EVA，否則視窗將於 10 秒後自動關閉。

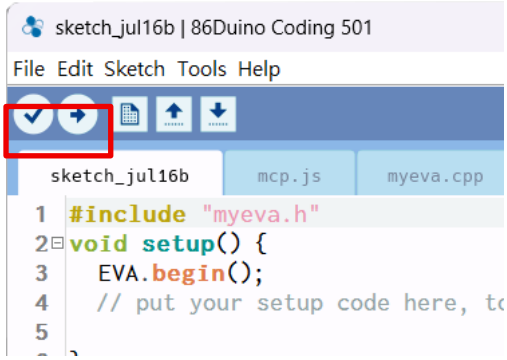


產生的程式碼與檔案如下：



a. sketch_jul23d：主專案（.ino，檔名依您的專案名稱而定）。
b. myeva.cpp：86EVA 的 C++ 程式碼。
c. myeva.h：86EVA 的標頭檔。

程式碼完成後，請點選工具列的 進行編譯；確認編譯完成且無錯誤後，即可點選 進行上傳。上傳完成後程式便會開始執行。



更詳細的資訊請參閱 86EVA, EtherCAT-Based Virtual Arduino。

4.5.7.1 疑難排解：無法順利上傳程式碼

當您無法順利上傳程式碼時，請開啟 86EVA 檢查 QEC EtherCAT MDevice 的環境是否異常。如下圖所示，請嘗試更新 QEC EtherCAT MDevice 的環境，更新內容包含下列三個項目：Bootloader、EtherCAT firmware 與 EtherCAT tool。



以下進一步說明更新的操作方式：

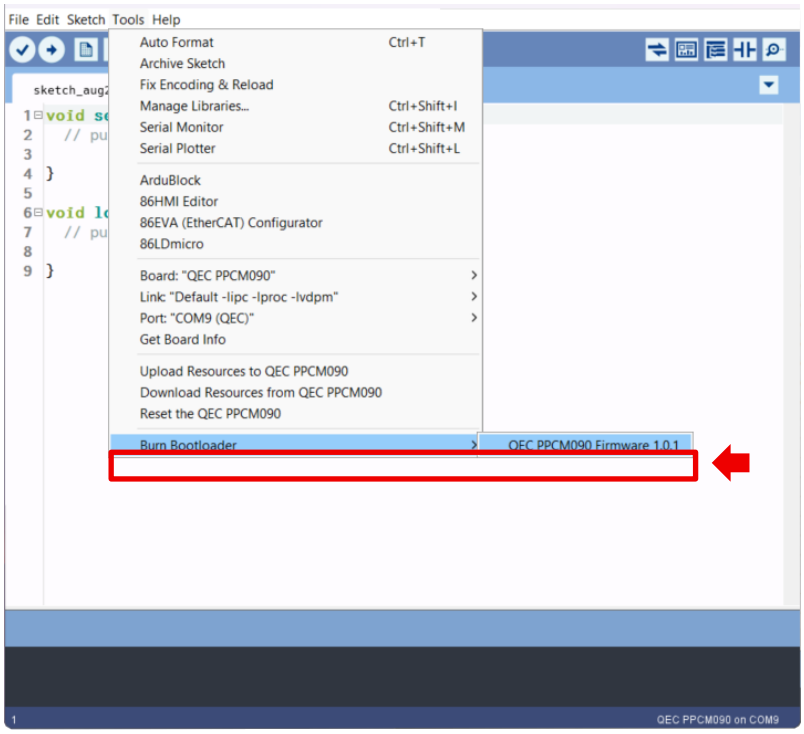
步驟 1：設定 QEC-M

1. 下載並安裝 86Duino IDE 501+（或更新版本）：可由 Software 頁面下載。
2. 連接 QEC MDevice：使用 USB 連接線將 QEC MDevice 連接至電腦。
3. 開啟 86Duino IDE：安裝完成後，開啟 86Duino IDE 軟體。
4. 選擇開發板：在 IDE 選單中選擇「**Tools**」→「**Board**」→「**QEC-PPC-M-090T**」（或您所使用的 QEC MDevice 型號）。
5. 選擇連接埠：在 IDE 選單中選擇「**Tools**」→「**Port**」，並選取 QEC MDevice 所連接的 USB 埠。

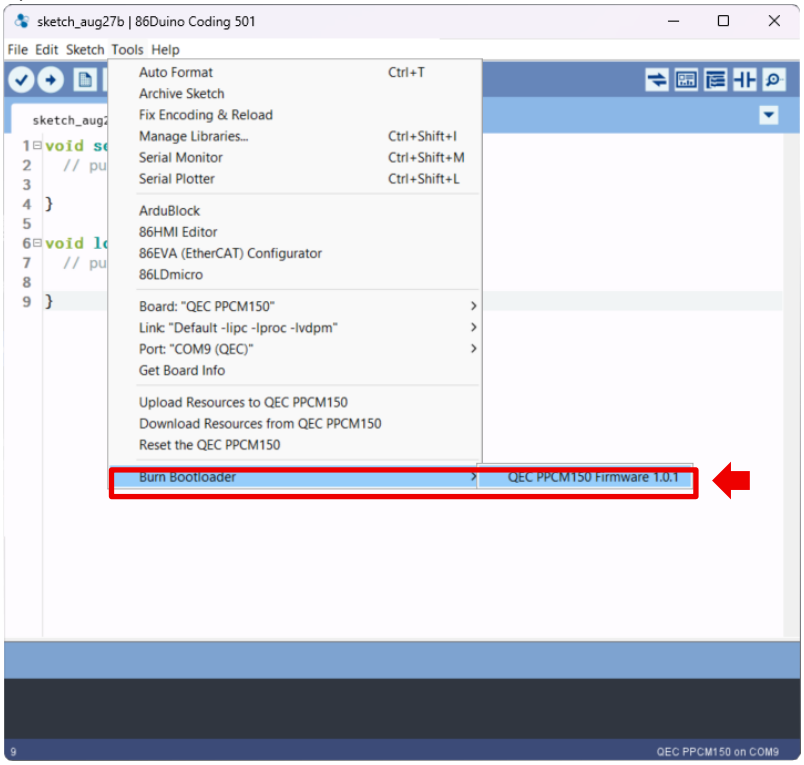
步驟 2：點選「Burn Bootloader」按鈕

連接至您的 QEC MDevice 後，選擇「**Tools**」→「**Burn Bootloader**」，畫面會顯示目前選取的 QEC MDevice 名稱。點選後即開始更新程序，約需 5 至 20 分鐘。

- QEC-PPC-M-090T:



- QEC-PPC-M-150T



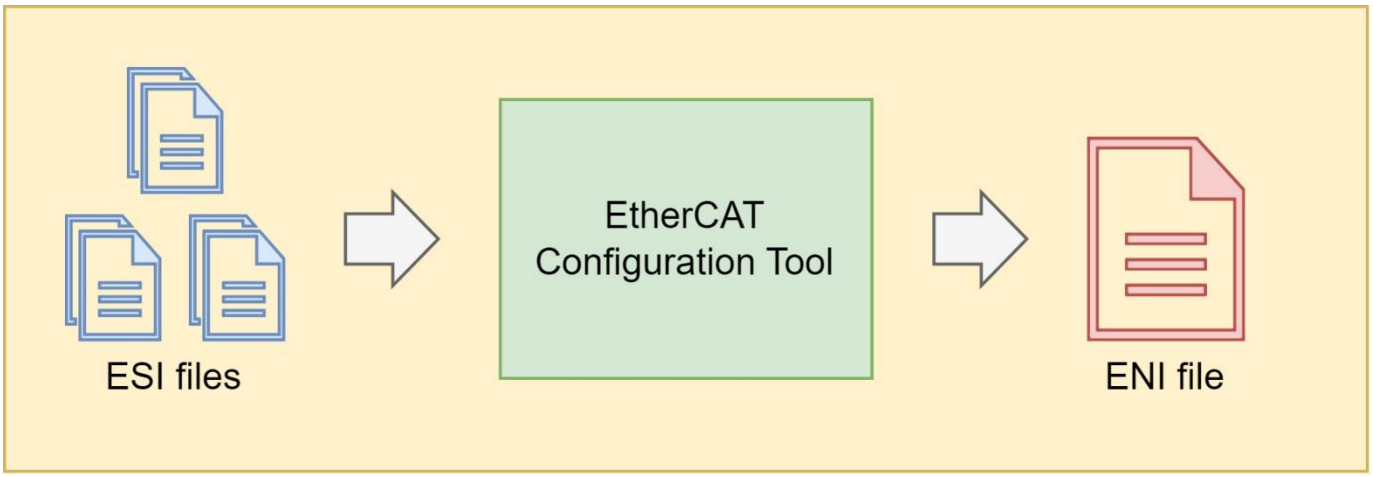
步驟 3：完成更新



完成上述步驟後，您的 QEC-M 裝置便已成功更新至最新版本的開發環境。

4.5.8 將 ENI 匯入 QEC MDevice

EtherCAT Network Information (ENI) 檔案包含設定 EtherCAT 網路所需的必要資訊。這份以 XML 為基礎的檔案，記錄了 MDevice 的一般資訊，以及所有連接之 SubDevice 的組態。使用 EtherCAT 組態工具，您可以讀取 ESI 檔案或對網路進行線上掃描以偵測所有已連接的 SubDevice，接著設定 PDO mapping、啟用 DC 等相關 EtherCAT 設定，並匯出 ENI 檔案。



EtherCAT Technology Group 要求 EtherCAT MDevice 軟體在 Network Configuration 項目中，至少須支援下列其中一種方式：線上掃描（Online Scanning）或讀取 ENI（Reading ENI）。
QEC EtherCAT 在 86Duino IDE 501+ 中的 EtherCAT Library 兩種方式皆支援。不過在讀取 ENI 時，目前 library 僅會由 ENI 檔案中擷取部分資訊用於網路組態。詳細說明請參閱 A.1 關於 86Duino IDE 的 ENI 組態。

4.5.8.1 方法 1：使用程式碼

完成 86Duino IDE 環境設定後，請將 ENI 檔案放入 USB 隨身碟，並插入您的 QEC EtherCAT MDevice。

*注意：透過 QEC MDevice 讀取 USB 隨身碟時，路徑為 P:\\。)

步驟 1：撰寫匯入程式碼：

在您的專案程式碼中，使用 EthercatMaster::begin() 函式匯入 ENI 檔案。

範例程式碼如下：

```
#include "Ethercat.h"

#define Device 4

EthercatMaster master;
EthercatDevice_Generic slave[Device];

void setup() {
    Serial.begin(115200);
    Serial.println(master.begin(ECAT_ETH_0, "P:\\test.xml"));
// Note: the USB disk readings via QEC Master will be under the P:\\ path.

    for (int i = 0; i < Device; i++)
    {
        Serial.println(slave[i].attach(i, master));
    }
    Serial.println(master.start());
    Serial.println("...OK");
}

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

步驟 2：上傳程式碼：

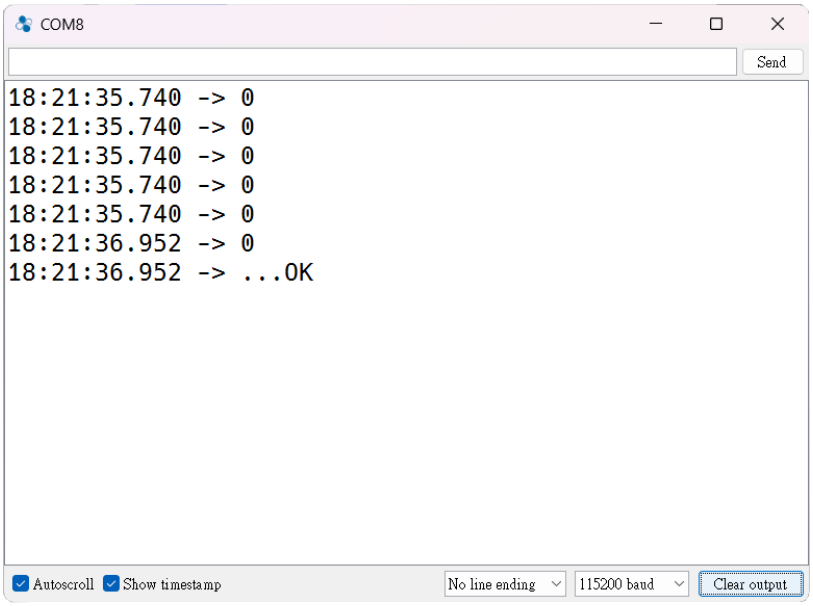
將程式碼上傳至您的 QEC EtherCAT MDevice。



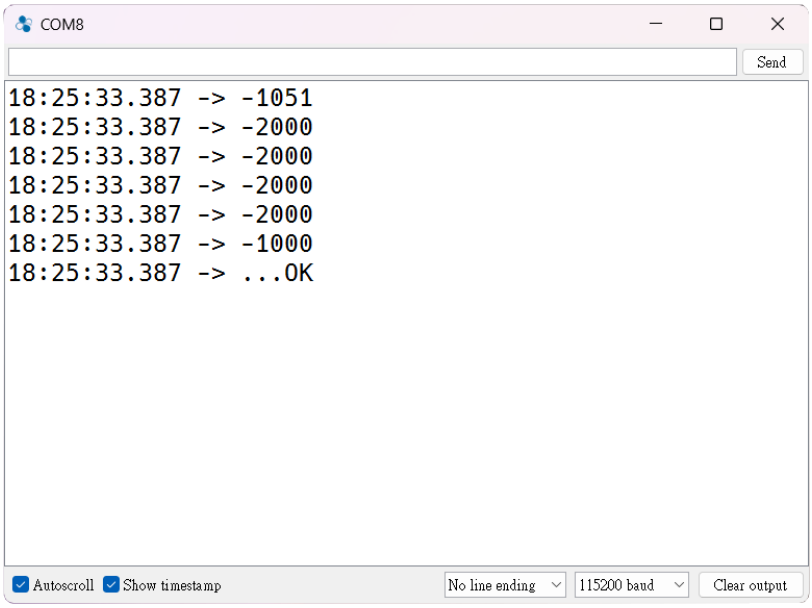
步驟 3：驗證 ENI 檔案是否匯入成功：

透過 Serial Monitor 確認 ENI 檔案已正確匯入，且網路運作正常。

若 ENI 檔案匯入成功，Serial Monitor 會顯示下列回傳值：



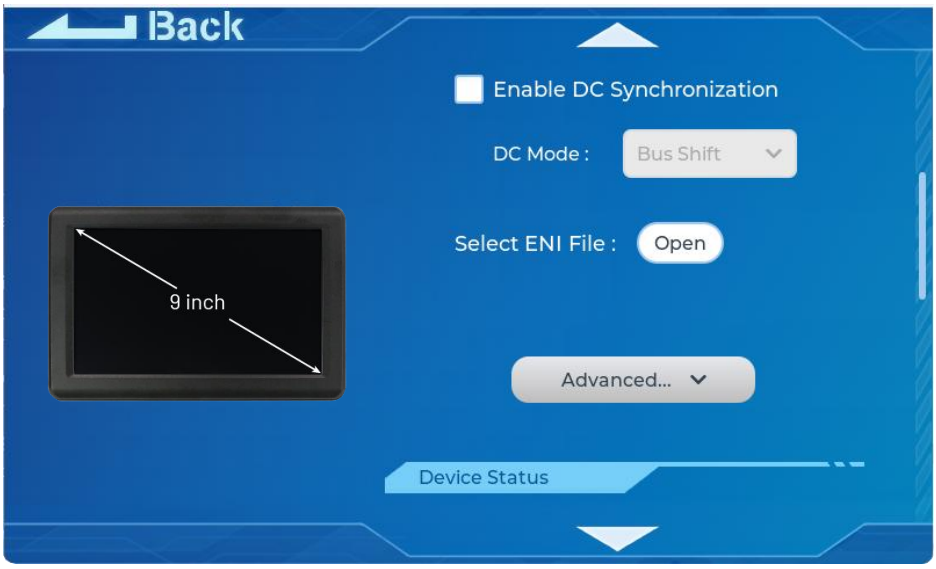
若出現錯誤，例如 -1051（代表 `ECAT_ERR_MASTER_ENI_MISMATCH`），其他錯誤碼與其代表的意義請參閱 EtherCAT API 使用手冊。



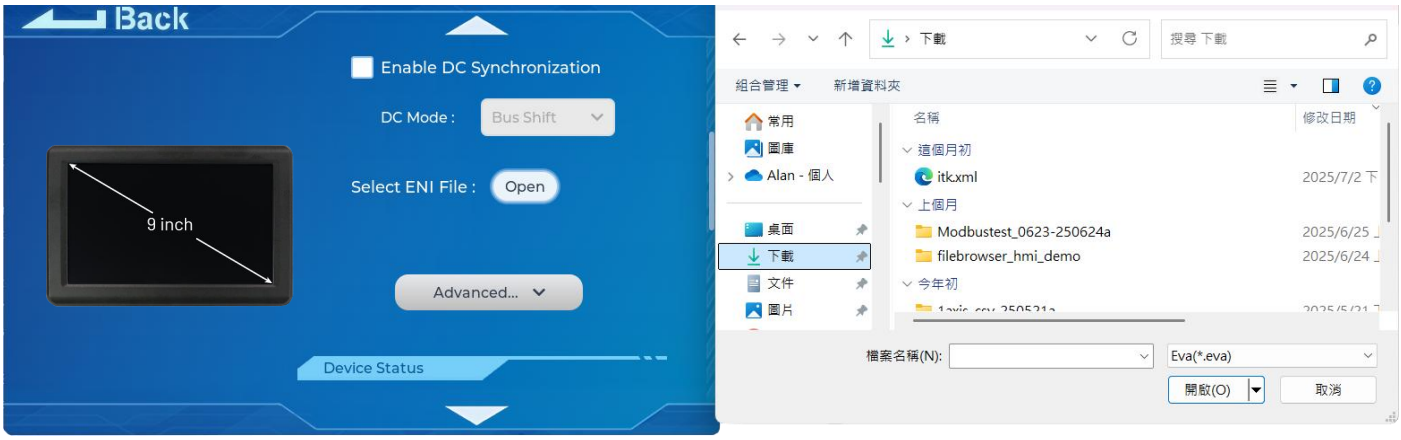
4.5.8.2 方法 2：使用 86EVA

86EVA 的設定方式請參閱 Ch 4.5.7 86EVA。

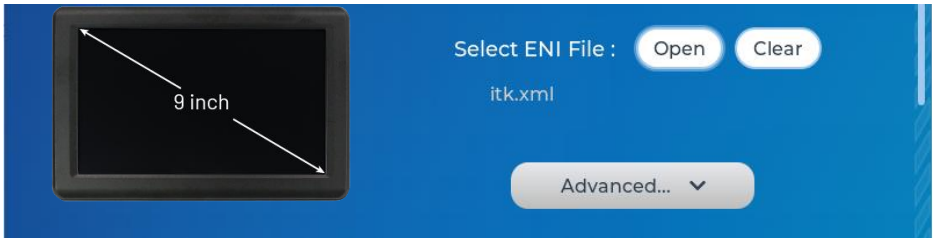
開啟 86EVA 並成功掃描 EtherCAT 網路後，請進入 QEC MDevice 的設定頁面，並往下捲動至「**Select ENI file**」區段。



點選「**Select ENI file**」旁的「**Open**」按鈕，畫面會出現開啟檔案視窗。瀏覽至您儲存 ENI 檔案的位置並開啟。



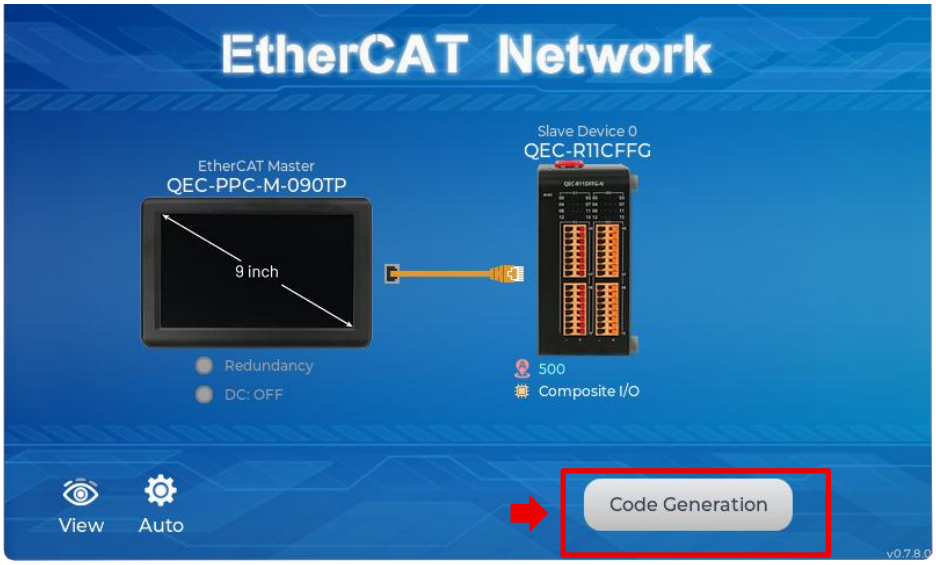
將 ENI 檔案匯入 86EVA 後，即可看到 ENI 檔案名稱。



請點選左上角的「**Back**」按鈕返回。



回到主畫面並按下右下角的「Code Generation」按鈕。



完成後，雙擊 OK 按鈕即可關閉 86EVA，否則視窗將於 10 秒後自動關閉。

產生的程式碼與檔案如下：



a. sketch_jul23d：主專案（.ino，檔名依您的專案名稱而定）。

b. myeva.cpp：86EVA 的 C++ 程式碼。

c. myeva.h：86EVA 的標頭檔。

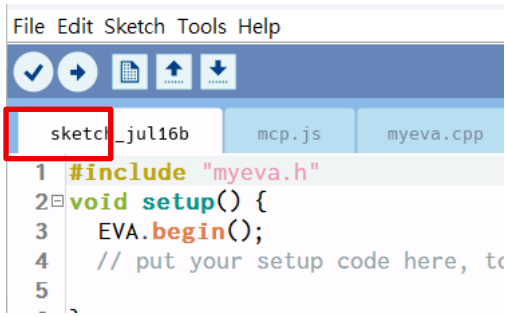
接著開始撰寫程式碼，設定您的 EtherCAT 系統。

請在 `void setup() { }` 中的 `EVA.begin();` 之前加入 `Serial.begin(115200);`，如此 EVA 程式的回傳值才能顯示於 Serial Monitor（鮑率 115200）。

範例程式碼如下：

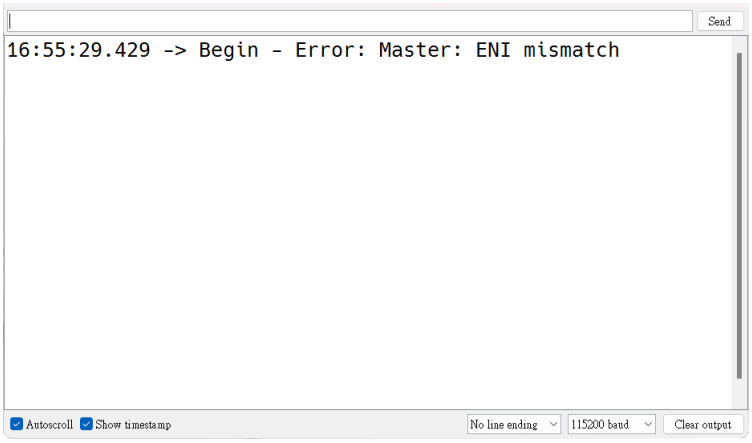
```
#include "myeva.h"

void setup() {
  Serial.begin(115200);
    EVA.begin();
  // put your setup code here, to run once:
}
void loop() {
  // put your main code here, to run repeatedly:
}
```

程式碼完成後，請點選工具列的 進行編譯；確認編譯完成且無錯誤後，即可點選 進行上傳。上傳完成後程式便會開始執行。



程式上傳成功後，必須透過 Serial Monitor 確認 ENI 檔案已正確匯入，且網路運作正常。

若 ENI 檔案匯入成功，**Serial Monitor 不會有任何回傳**；若發生錯誤，則會直接將錯誤回傳至 Serial Monitor。



4.6 Bootloader Menu 使用說明

本節介紹 Bootloader Menu，它提供友善的操作介面，供您設定與管理 QEC MDevice。



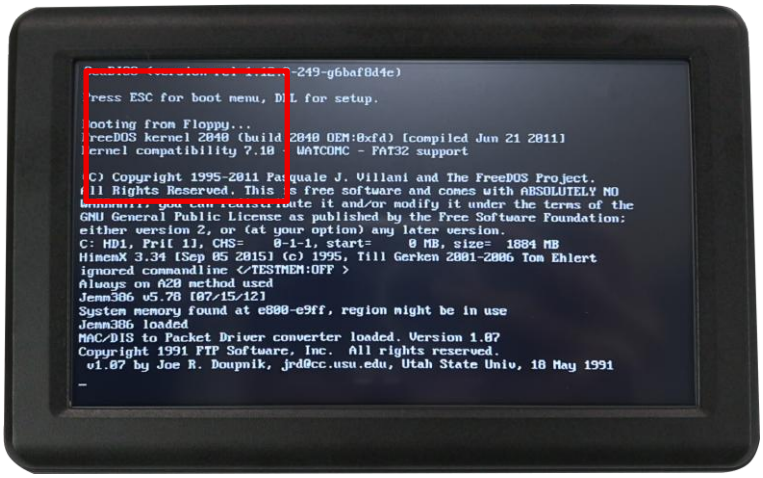
Bootloader 是 86Duino 上電後執行的第一個程式，在 86Duino 硬體與開發環境之間扮演關鍵的橋樑角色。Bootloader 透過 USB Device / Programming Port 與 86Duino Coding IDE 通訊，讓使用者能將 sketch 程式由開發環境順暢地上傳至 86Duino 硬體。
Bootloader 會接收使用者上傳的 sketch 程式，並寫入 86Duino 裝置的板載記憶體。sketch 上傳成功後，Bootloader 會自動執行該程式。

本節詳細說明如何有效操作與運用 Bootloader Menu 的各項功能。

本節以 **QEC-PPC-M-090T** 為例說明。

4.6.1 開啟 Bootloader Menu

QEC-PPC-M MDevice 開機時，持續觸壓 LCD 螢幕左上角即可開啟 Bootloader Menu。



持續按住左上角後，QEC MDevice 畫面上會出現三個提示列。

第一列會顯示「**Keep Pressing**」。



第二列會顯示「**Touch Calibration**」。



第三列會顯示「**Bootloader Menu**」。

在此列時放開螢幕，即可進入 Bootloader Menu 頁面。



Bootloader Menu 的畫面如下圖所示。



以下說明僅以 Bootloader 的畫面為例。

***注意：** 所有 QEC-PPC-M 系列 MDevice 皆可用相同方式開啟 Bootloader Menu。

Bootloader Menu 的上方選單可讓您快速存取主要設定與功能。



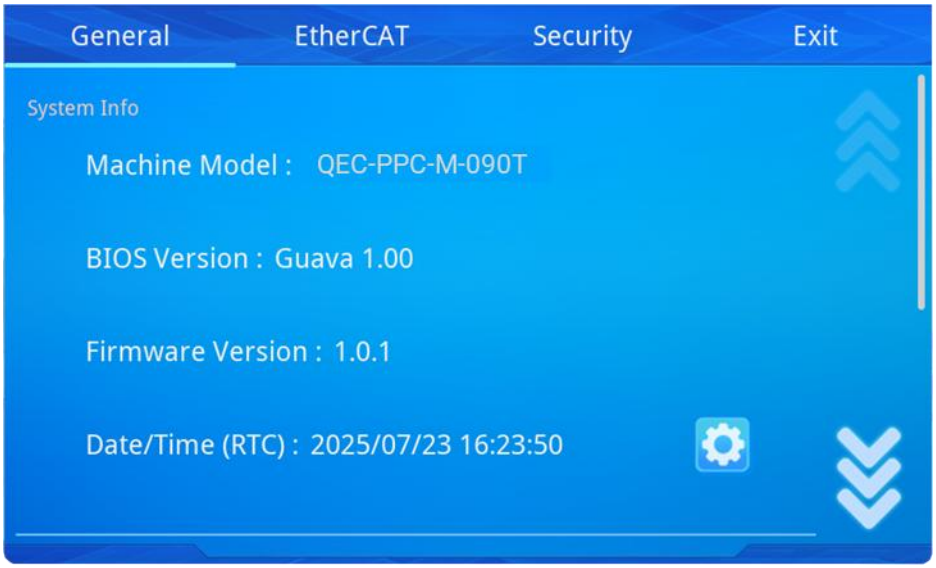
包含下列頁籤：

1. **General**：
    - 檢視系統資訊 (例如機種型號、BIOS 版本、firmware 版本與 RTC)。
    - 存取基本設定，例如啟用 BIOS 選單或允許 IDE 連線。
2. **EtherCAT**：
    - 設定 BIOS 預設的 EtherCAT cycle time，並啟用 / 停用 redundancy。
3. **Security**：
    - 設定或更新 Bootloader 密碼，以保護存取權限。
4. **Exit**：
    - 儲存或捨棄變更並重新開機。

4.6.2 General 頁面

General 頁面顯示重要的系統資訊，並提供基本設定選項。

4.6.2.1 系統資訊



- **Machine Model**：顯示目前的硬體機種（例如 QEC-PPC-M-090T、QEC-PPC-M-150T）。
- **BIOS Version**：顯示已安裝的 BIOS 版本。
- **Firmware Version**：顯示目前安裝的 firmware 版本。
- **Date/Time (RTC)**：顯示系統的即時時脈，使用者可自行調整此設定。

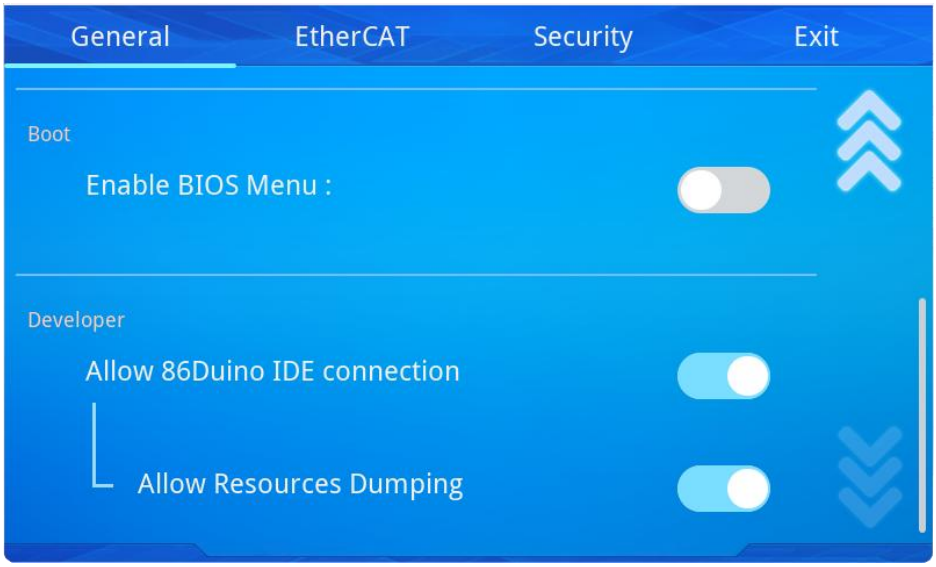
***補充說明：**

調整 Date/Time (RTC) 的方式：

1. 點選齒輪圖示 開啟「Set Date/Time」視窗。
2. 透過畫面上的選項修改日期與時間設定。
3. 點選 OK 儲存變更，或點選 cancel 捨棄變更。



4.6.2.2 Boot 與 Developer 選項



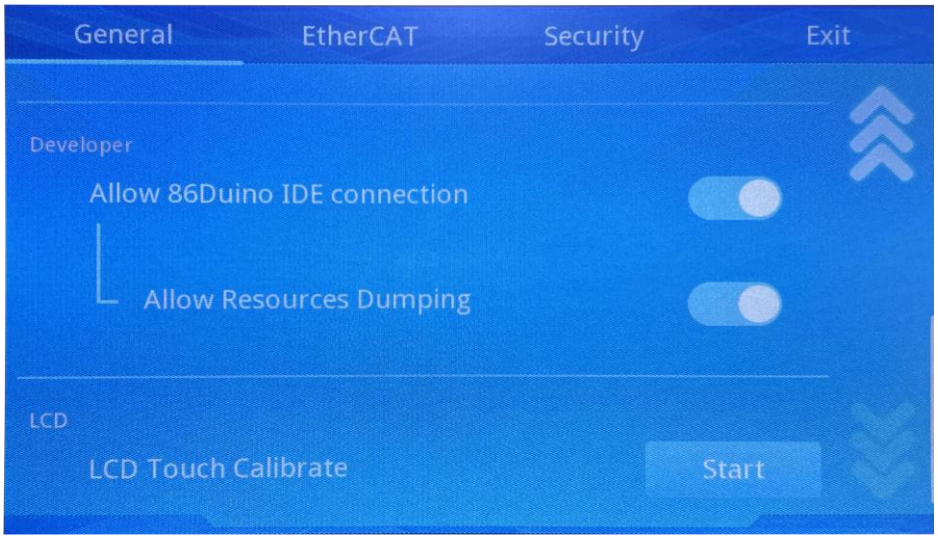
Boot：

- **Enable BIOS Menu**：啟用或停用 BIOS 選單。

Developer：

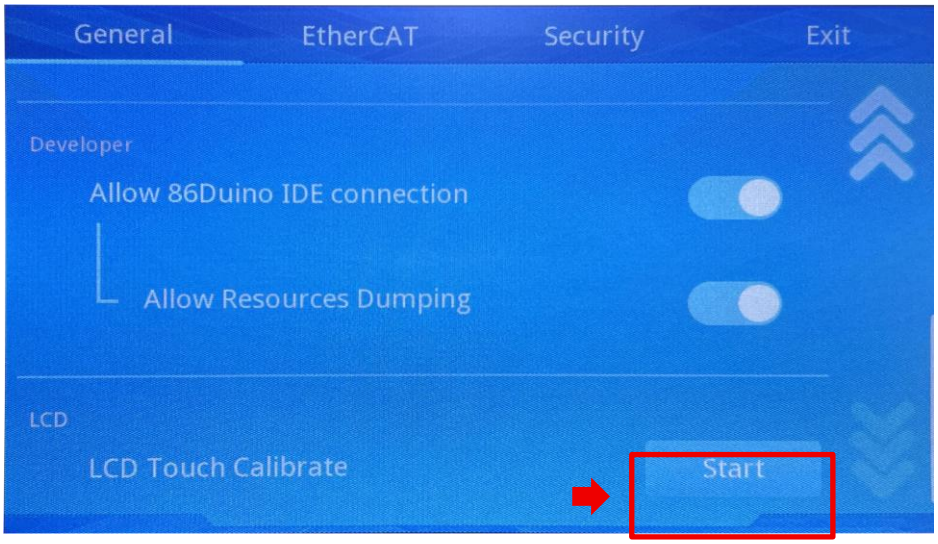
- **Allow 86Duino IDE connection**：供開發用途使用。
- **Allow Resources Dumping**：啟用或停用資源傾印功能，供除錯使用。

4.6.2.3 LCD 選項



- **LCD Touch Calibrate**：啟動 QEC MDevice 的 LCD 觸控校正功能。

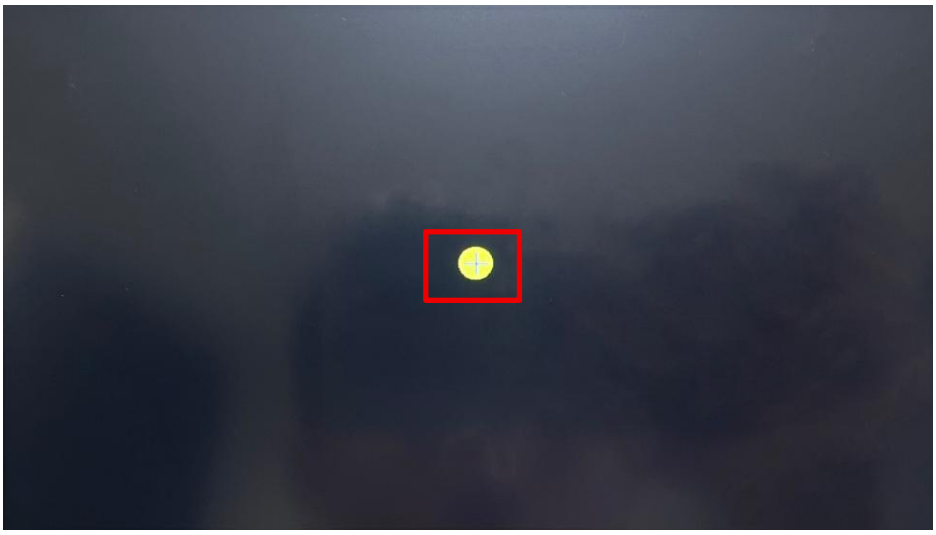
點選 LCD Touch Calibrate 的「**Start**」按鈕



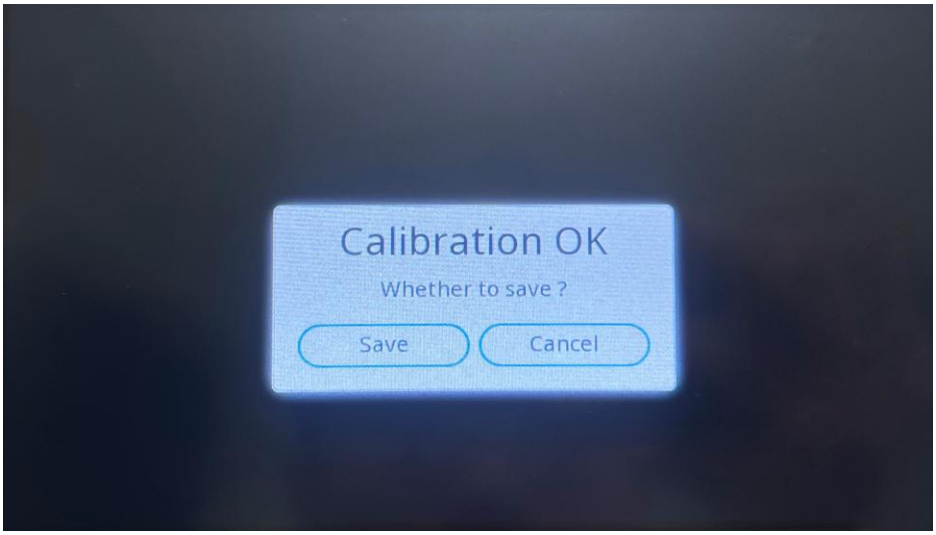
畫面四個角落會依序顯示四個黃色圓圈，請依序觸壓這些圓圈以完成 LCD 觸控校正。



完成四個角落的圓圈後，螢幕中央會再出現一個黃色圓圈，請一併觸壓。



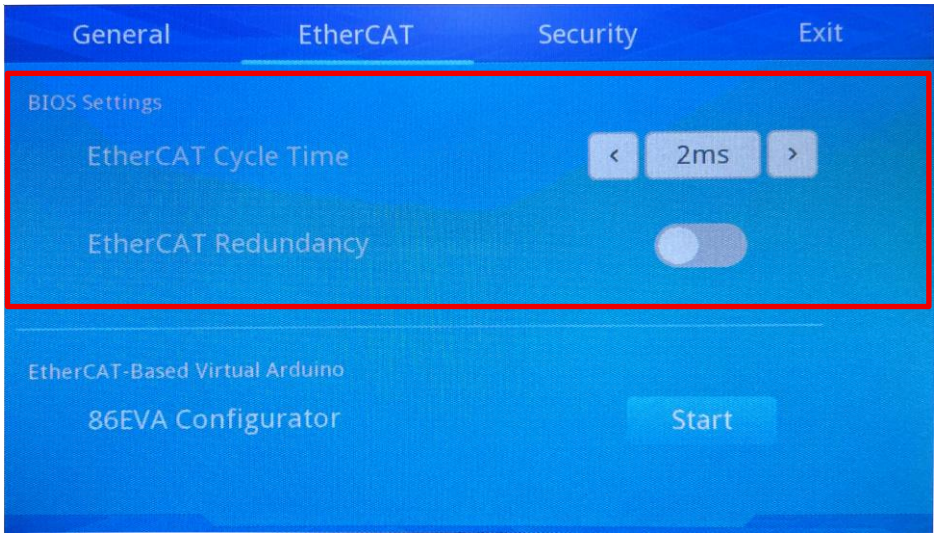
至此即完成 LCD 觸控校正。請點選「**Save**」，即可返回 Bootloader Menu。



4.6.3 EtherCAT 頁面

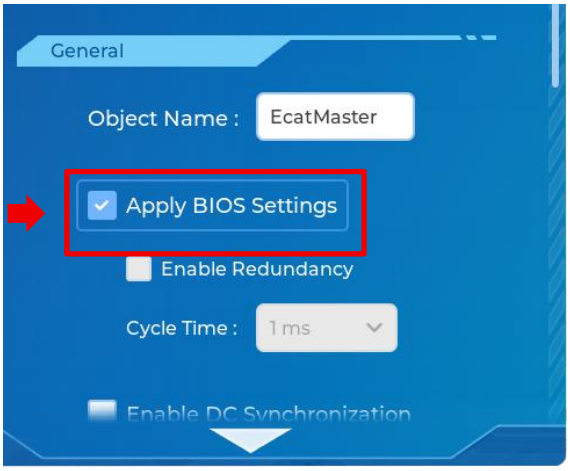
EtherCAT 頁面可供使用者設定 EtherCAT 與 86EVA Configurator 的相關項目，包括 Cycle Time 與 Redundancy，這些皆屬於 BIOS 預設值。

4.6.3.1 BIOS 設定



- **EtherCAT Cycle Time**：設定預設的 EtherCAT 通訊 cycle time，可選項目為 62.5μs、125μs、250μs、500μs、1ms 與 2ms。
- **EtherCAT Redundancy**： 啟用或停用預設的 EtherCAT redundancy 組態。

這些 BIOS 設定會直接影響 86EVA Configurator 中的 EtherCAT 組態，如 QEC MDevice 頁面所示。

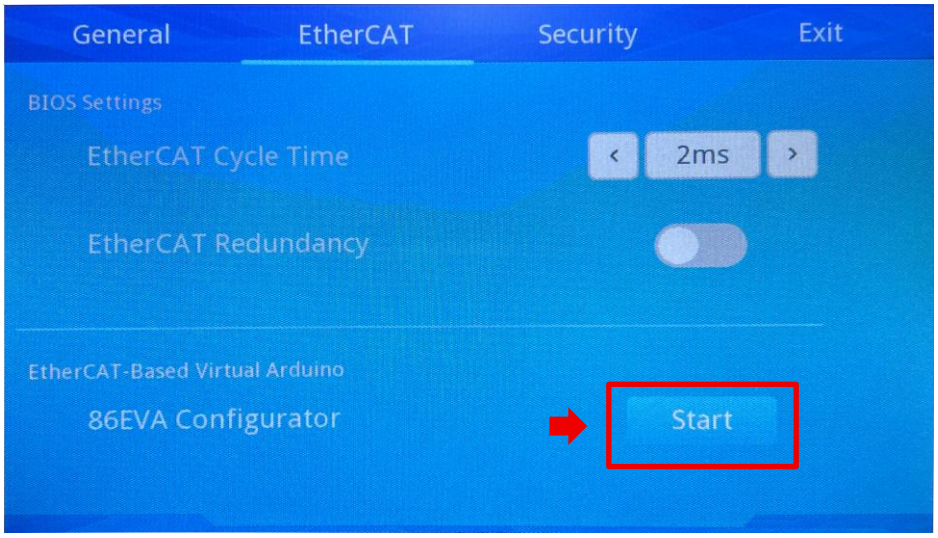


在 EtherCAT 頁面啟用 redundancy 後，86EVA Configurator 中的 Redundancy 勾選方塊會自動被勾選；而在 BIOS 中設定的 Cycle Time，也會成為 Configurator 中的預設值。

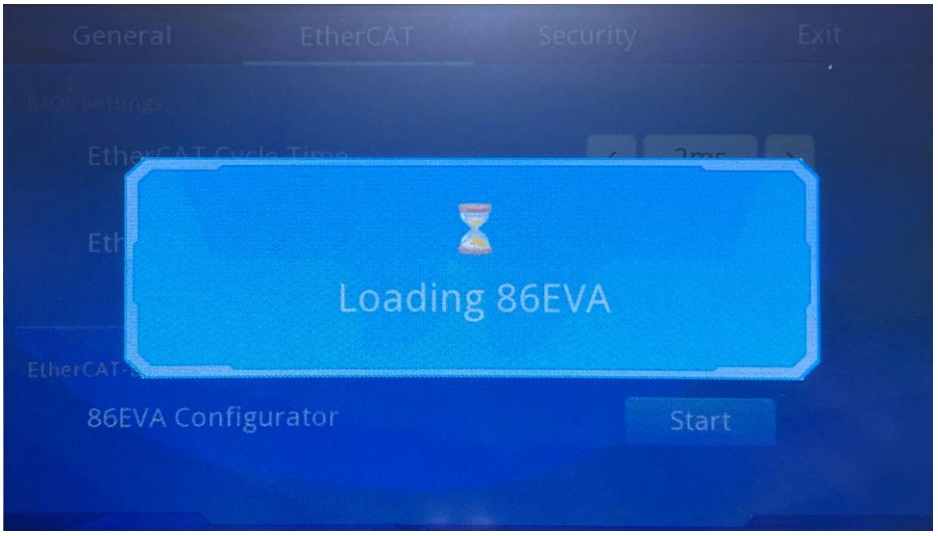
4.6.3.2 EtherCAT-Based Virtual Arduino

此功能可讓使用者透過 86EVA Configurator 掃描並設定 EtherCAT-Based Virtual Arduino 的對應關係。

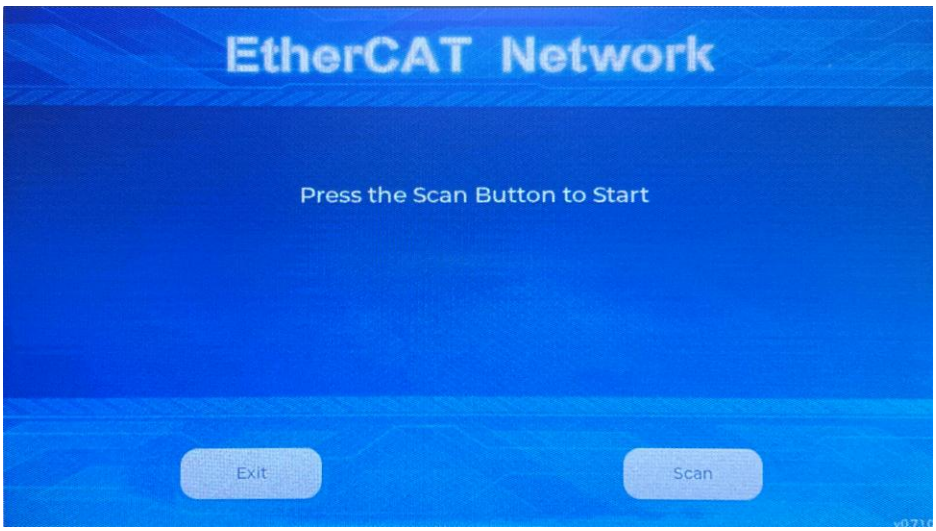
點選「**Start**」按鈕即可進入 86EVA 視窗。



點選「**Scan**」按鈕即可開始掃描 EtherCAT 網路。



QEC-PPC-M 系列 MDevice 上的 86EVA。

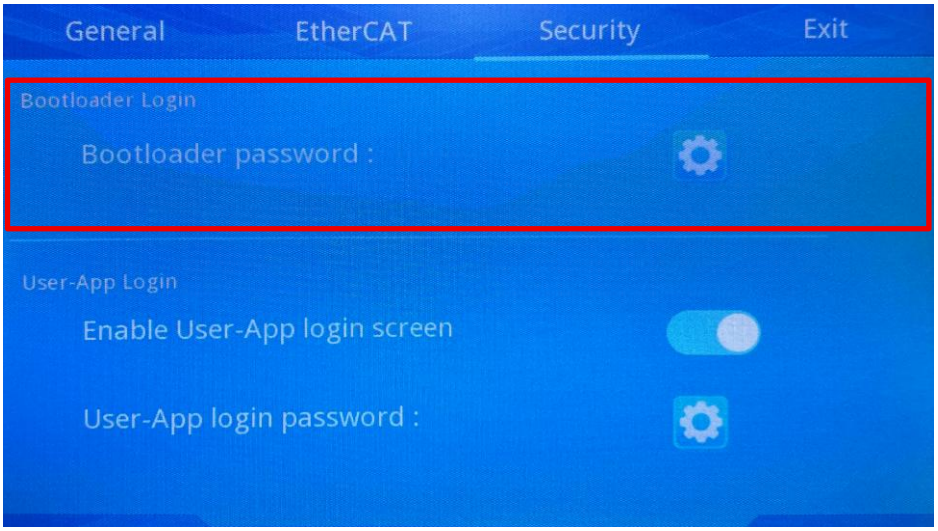


*注意：QEC-PPC-M 系列 MDevice 上的 86EVA Configurator 仍在開發中，目前不建議使用。

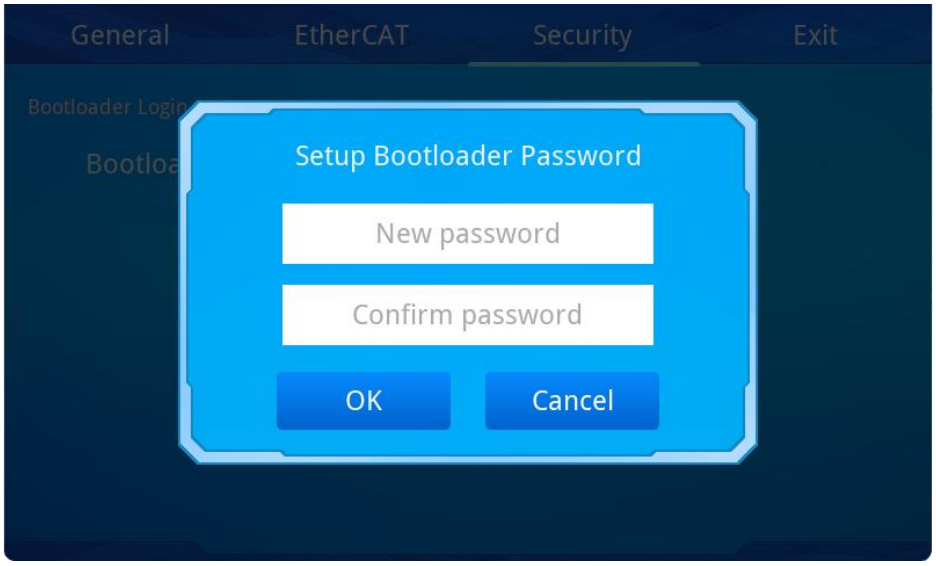
4.6.4 Security 頁面

Security 頁面可供使用者設定與管理 Bootloader 密碼，以保護程式與組態設定，確保只有經授權的使用者才能存取 Bootloader Menu。

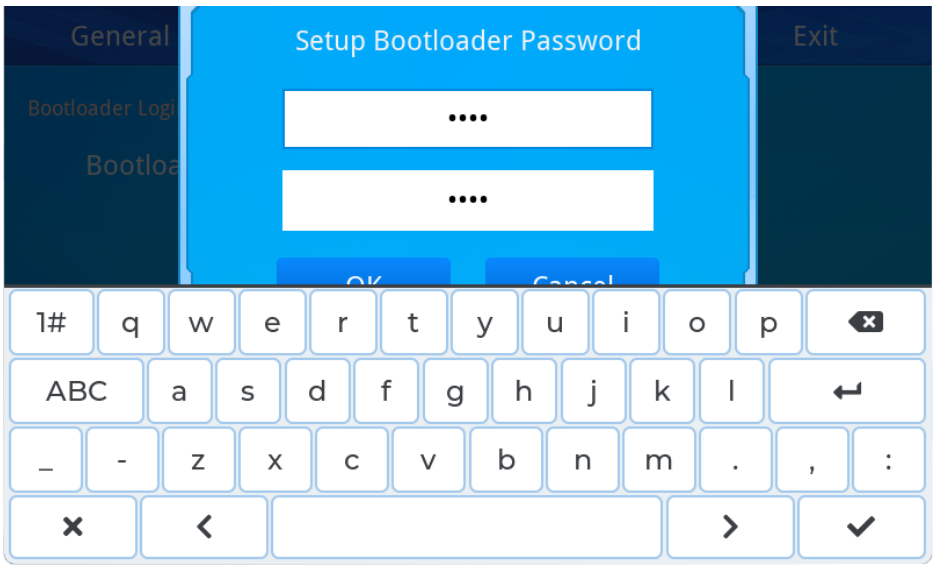
4.6.4.1 Bootloader 登入



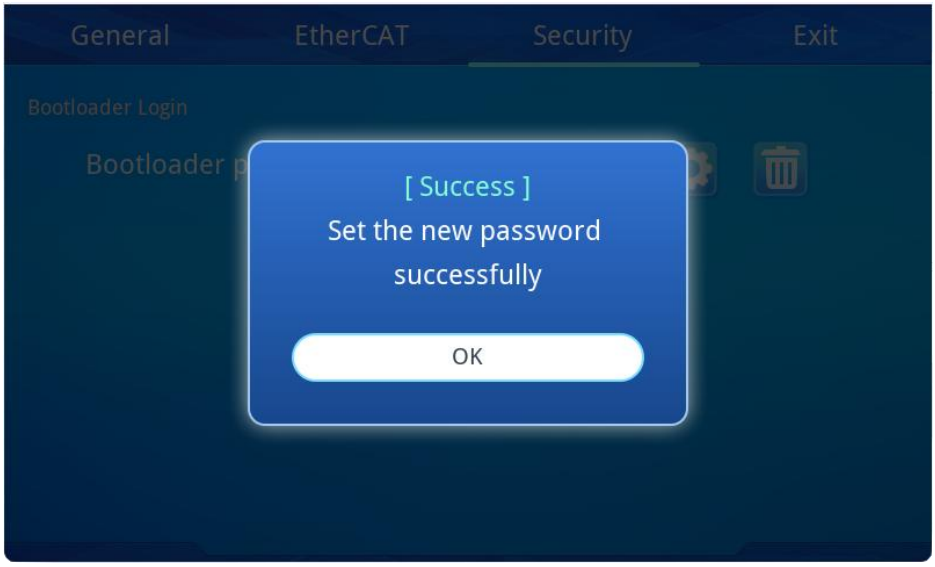
- **Bootloader password**：使用者可點選齒輪圖示 開啟「**Setup Bootloader Password**」視窗，如下圖所示。



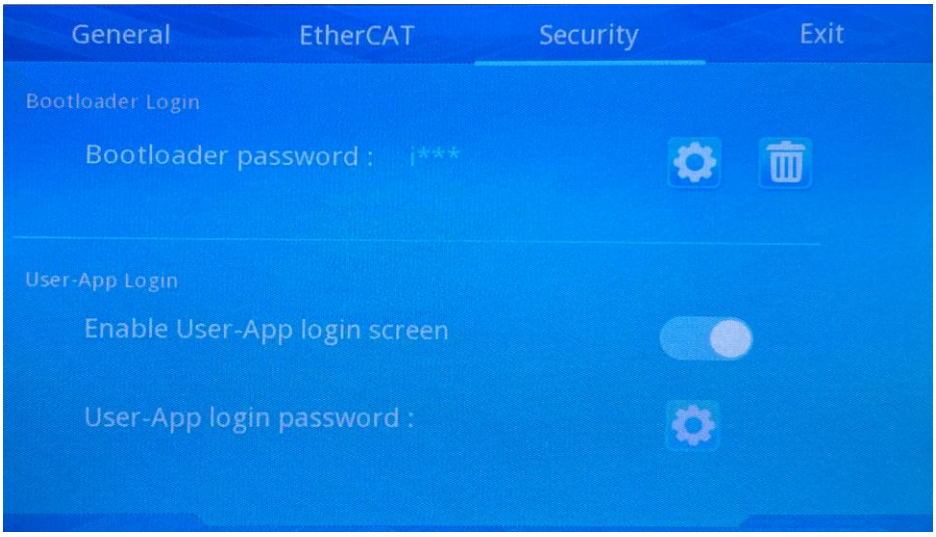
點選文字輸入區後，畫面上會出現鍵盤。



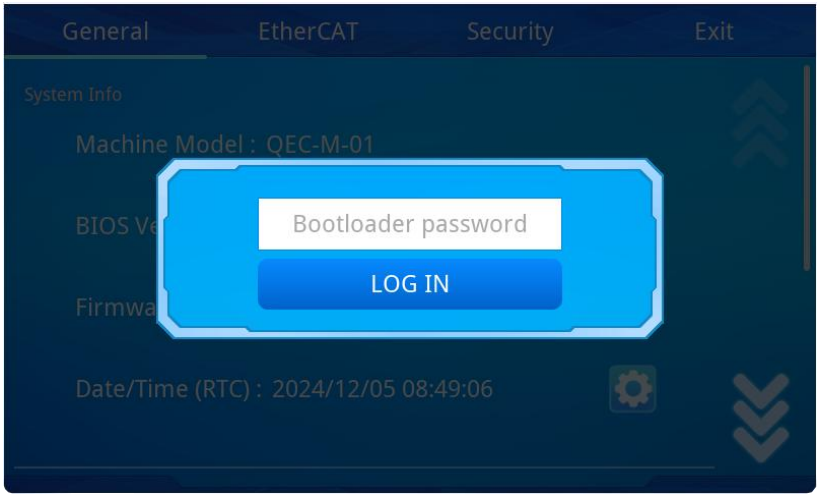
設定完 Bootloader 密碼後點選「**OK**」，畫面會出現【**Success**】視窗，表示新密碼已設定成功。



此時 Security 頁面會顯示如下。



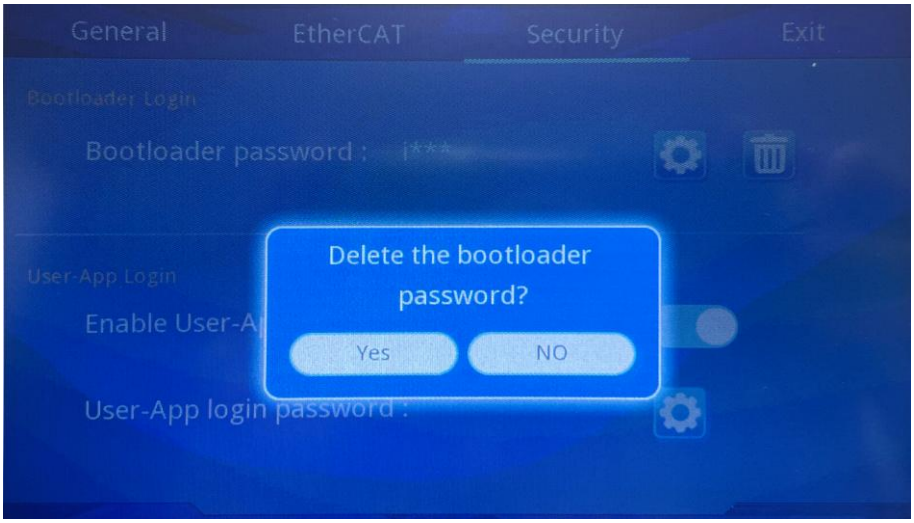
儲存 Bootloader Menu 的組態後，密碼即會一併儲存，下次開啟 Bootloader Menu 時便會要求輸入。



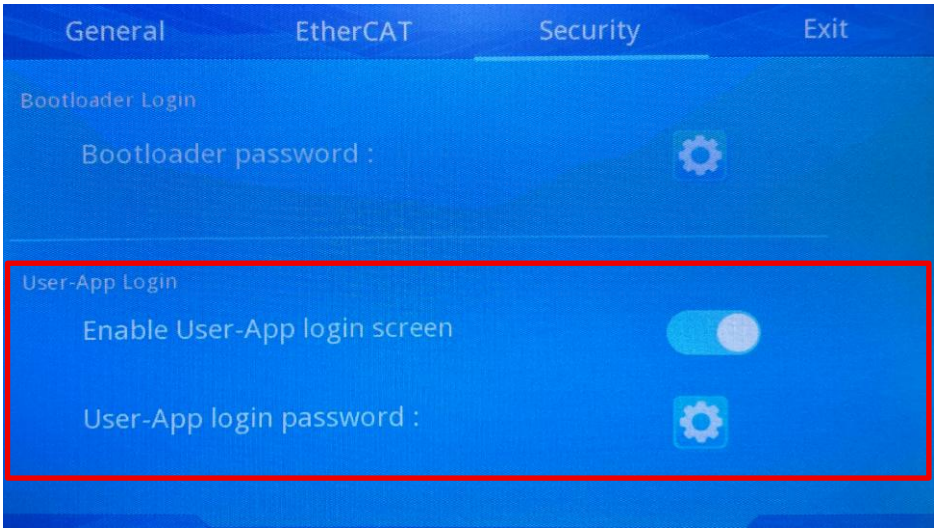
使用者可點選齒輪圖示變更密碼，或點選垃圾桶圖示刪除密碼。



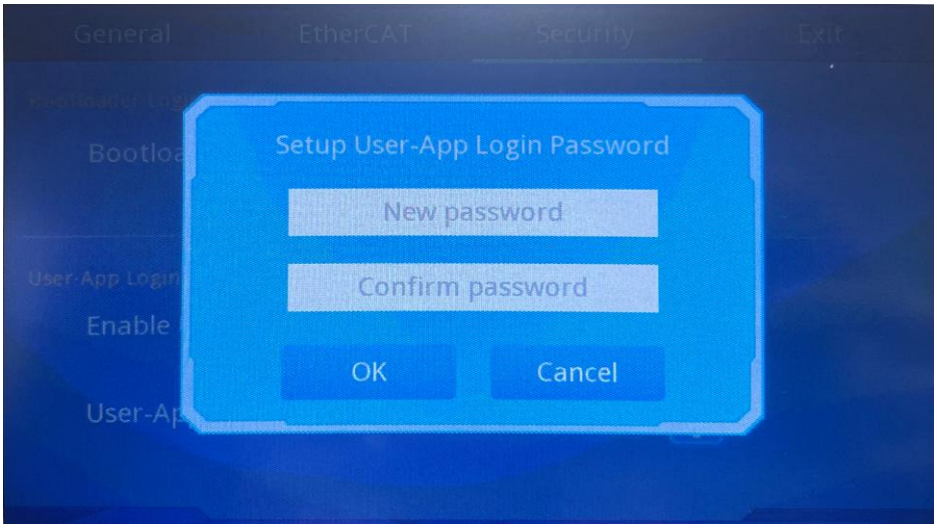
點選垃圾桶圖示刪除 Bootloader 密碼時，畫面會顯示下列確認視窗。



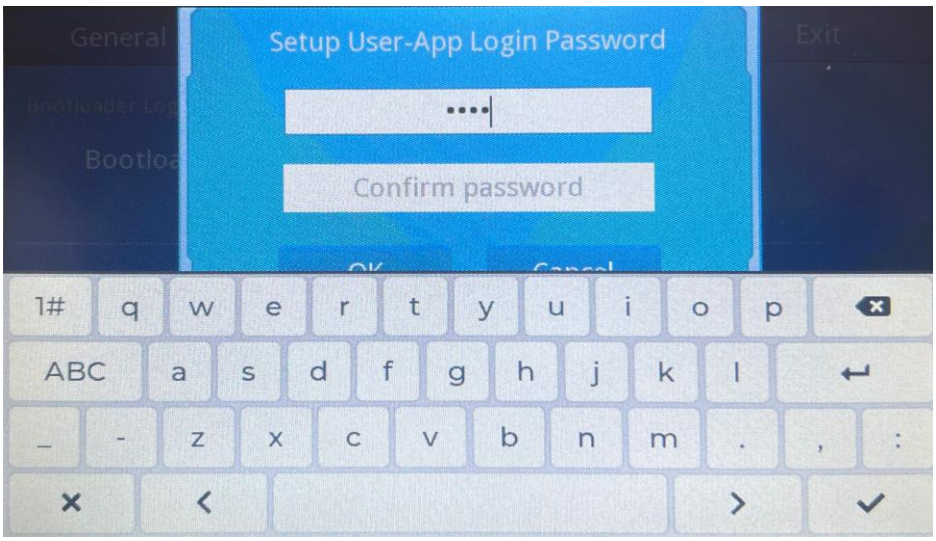
4.6.4.2 User-App 登入



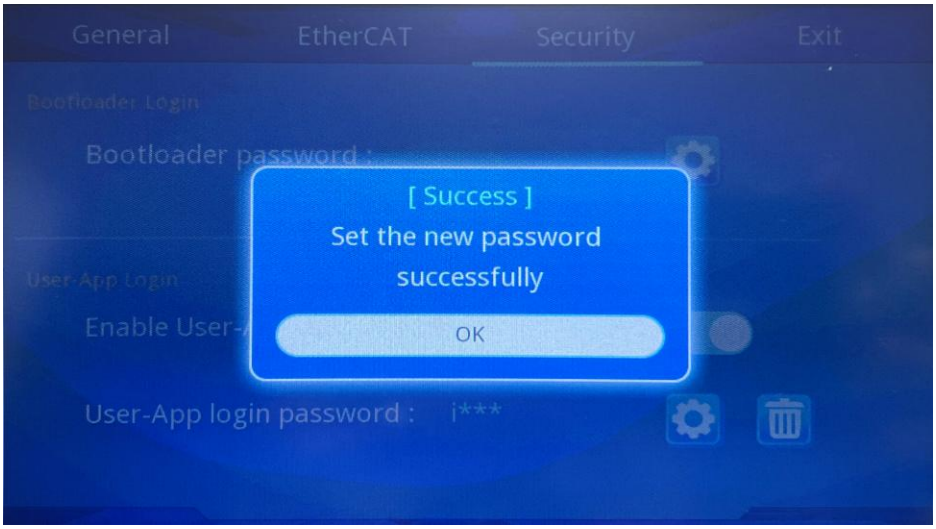
- **Enable User-App login screen**： 勾選此選項後，若已設定密碼，QEC MDevice 在執行使用者應用程式前會先顯示登入畫面。
- **User-App login password**： 使用者可點選齒輪圖示 開啟「**Setup User-App Login Password**」視窗，如下圖所示。



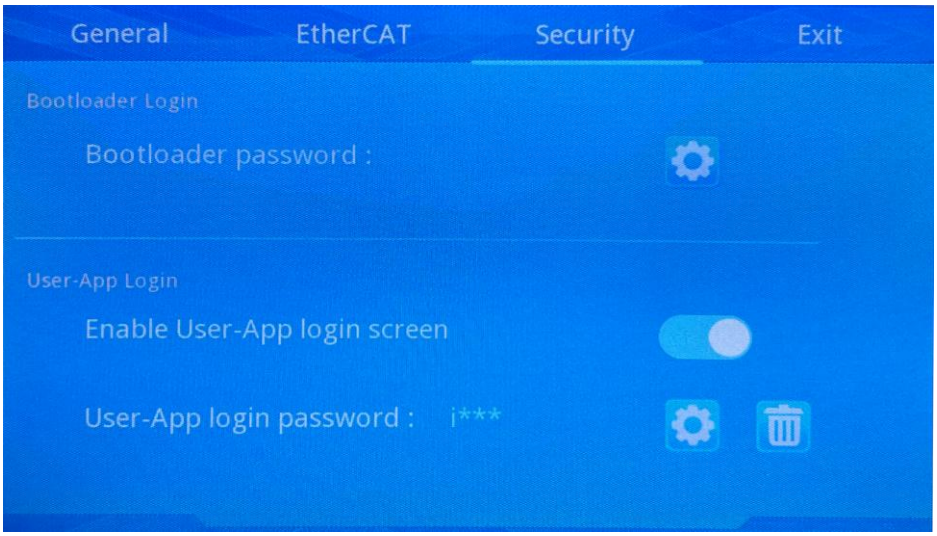
點選文字輸入區後，畫面上會出現鍵盤。



設定完 User-App 登入密碼後點選「**OK**」，畫面會出現 **[Success]** 視窗，表示新密碼已設定成功。

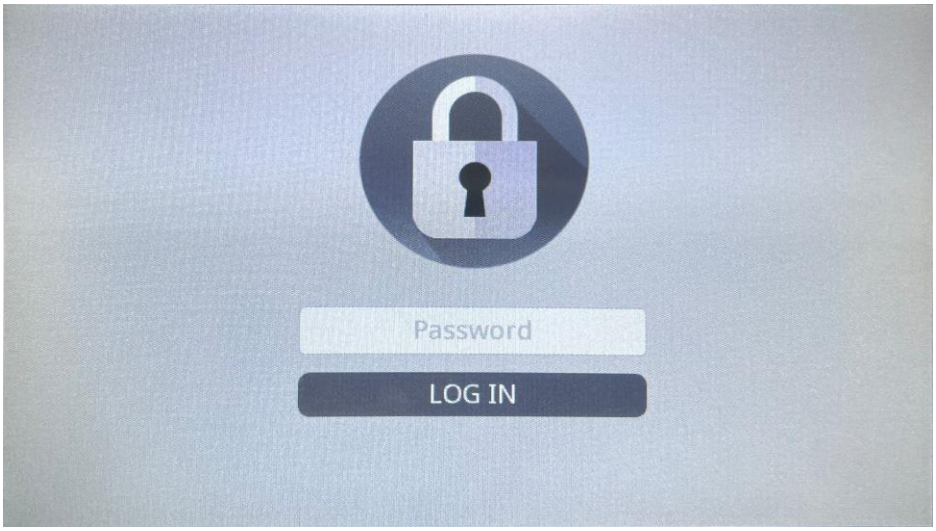


此時 Security 頁面會顯示如下。



儲存 Bootloader Menu 的組態後，密碼即會一併儲存，下次開啟 QEC MDevice 時便會顯示 User-App 登入畫面。

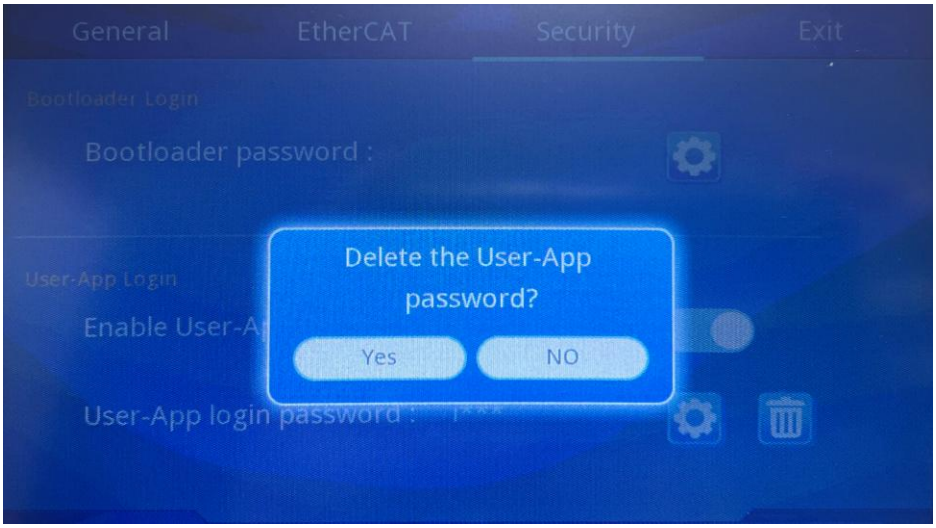
如下圖所示。



使用者可點選齒輪圖示變更密碼，或點選垃圾桶圖示刪除密碼。



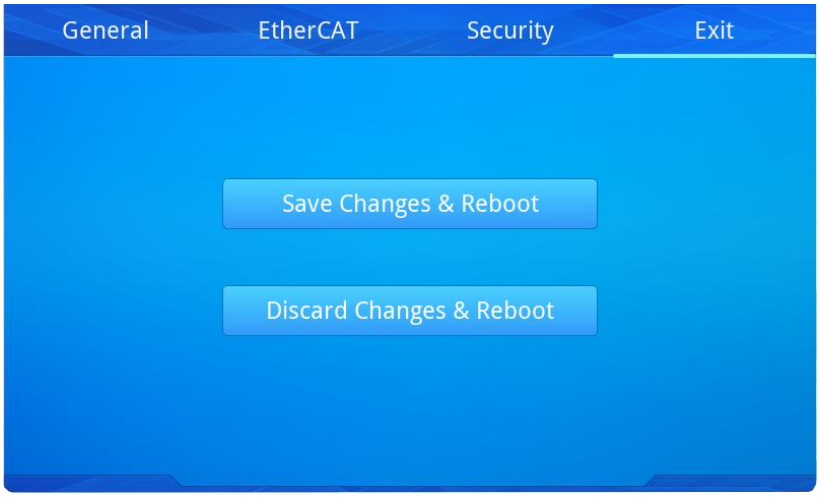
點選垃圾桶圖示刪除 User-App 登入密碼時，畫面會顯示下列確認視窗。



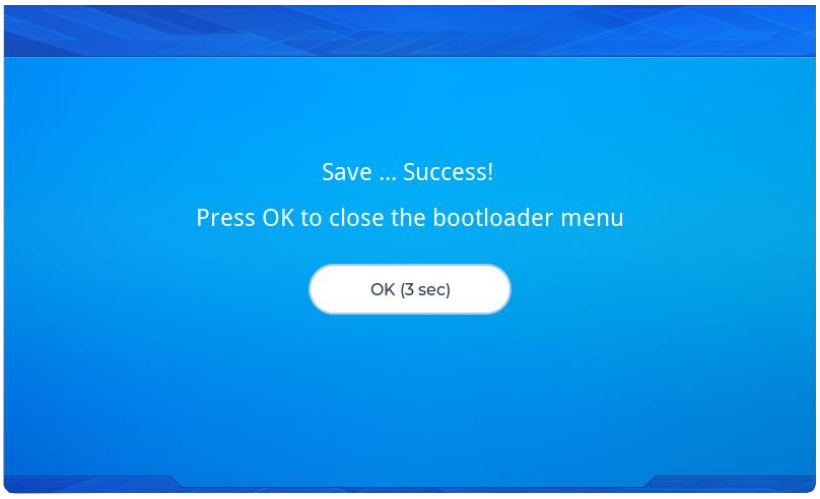
4.6.5 Exit 頁面

Exit 頁面可供使用者在重新啟動 QEC MDevice 前，選擇儲存或捨棄於 Bootloader Menu 中所做的變更，確保所有設定依使用者意願正確套用或還原。

Exit 頁面：



- **Save Changes & Reboot：** 選擇此項目時，於 Bootloader Menu 中所做的所有變更都會被儲存。Bootloader Menu 會顯示「Save... Success」確認訊息，表示組態已成功儲存。



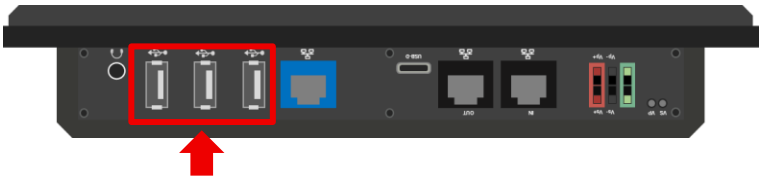
使用者可點選 OK 立即關閉 Bootloader Menu，或等待 5 秒後自動關閉。

- **Discard Changes & Reboot：** 選擇此項目會捨棄所有尚未儲存的變更，並以先前儲存的組態重新啟動 QEC MDevice。Bootloader Menu 會直接關閉而不進行儲存。

4.7 USB 裝置使用說明

本節說明如何正確使用 QEC-PPC-M 系列 MDevice 的 USB 裝置與檔案儲存功能。

QEC-PPC-M 系列 MDevice 具備 3 組支援熱插拔的標準 USB 2.0 埠，使用者可連接 USB 儲存裝置，用於檔案儲存、傳輸或存取組態資料。



使用者可利用 SD Library 讀寫 USB 資料夾，例如透過此 library 建立 .txt 檔案。在 QEC-PPC-M 系列上，SD library 可讀寫 USB 儲存裝置或 eMMC 儲存空間，並支援其上的 FAT16 與 FAT32 檔案系統。傳入 SD library 函式的檔名可包含以正斜線 / 分隔的路徑，例如「directory/filename.txt」。由於工作目錄永遠是所選儲存裝置的根目錄，檔名前是否加上斜線都指向同一個檔案（例如「/file.txt」等同於「file.txt」）。此 library 支援同時開啟多個檔案。
不過若要將檔案放入特定資料夾，則無法使用 SD Library，使用者必須改用標準 C 語言的 `fopen()` 函式手動指定檔案位置。

***警告：儲存槽使用建議**

QEC-PPC-M 系列 MDevice 具有 A、B、C、P 四個儲存槽。

- **A** 與 **B** 槽：保留給系統檔案使用。強烈建議使用者不要將檔案存放於此，否則可能造成系統損毀，需回復原廠設定或重新燒錄 32MB flash。
- **C** 槽：此槽對應 eMMC，建議用於安全存放使用者檔案。
- **P** 槽：此槽對應 USB 裝置，同樣適合存放使用者檔案。

4.7.1 範例 1：將 .txt 儲存至 USB 隨身碟

以下範例示範如何以 C 語言的 `fopen()` 建立檔案，並將 .txt 檔存放至**外接 USB 隨身碟**，說明如何指定檔案位置並確保檔案存放於適當的儲存槽。

範例程式碼如下：

```
char* strs[] = {"Hello, world", "Hello, world2", "Hello, world3"};

void setup() {
   FILE *fp;
   char str[256];

   Serial.begin(115200);

   // Write strings to the test.txt file in the P slot
   fp = fopen("P:\\test.txt", "w");
   for (int i=0; i<sizeof(strs)/sizeof(char*); i++)
      fprintf(fp, "%s\n", strs[i]);

   fclose(fp);

   // Read the strings and print them
   fp = fopen("P:\\test.txt", "r");
   while (fgets(str, sizeof(str), fp)!=NULL ) {
      Serial.print(str);
   }
   fclose(fp);
}

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

}
```

***注意：** 請在 QEC MDevice PPC 系列上使用外接 USB 隨身碟。

4.7.2 範例 2：將 .txt 儲存至 eMMC

以下範例示範如何以 C 語言的 `fopen()` 建立檔案，並將 .txt 檔存放至**內建 eMMC**，說明如何指定檔案位置並確保檔案存放於適當的儲存槽。

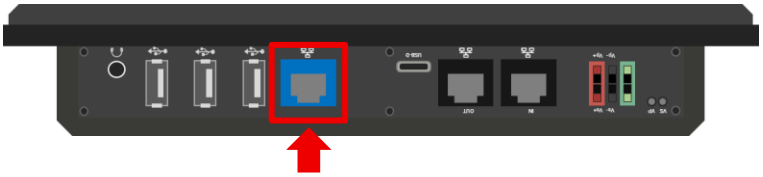
範例程式碼如下：

```
char* strs[] = {"Hello, world", "Hello, world2", "Hello, world3"};

void setup() {
   FILE *fp;
   char str[256];

   Serial.begin(115200);

   // Write strings to the test.txt file in the C:\\
   fp = fopen("C:\\test.txt", "w");
   for (int i=0; i<sizeof(strs)/sizeof(char*); i++)
      fprintf(fp, "%s\n", strs[i]);

   fclose(fp);

   // Read the strings and print them
   fp = fopen("C:\\test.txt", "r");
   while (fgets(str, sizeof(str), fp)!=NULL ) {
      Serial.print(str);
   }
   fclose(fp);
}

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

}
```

4.8 Giga LAN 設定

本節介紹 QEC MDevice 的 Giga LAN 設定與控制方式。

QEC-PPC-M 系列具備一組 Giga LAN 埠，專供一般網路用途的對外 Ethernet 通訊使用。



*注意： QEC-PPC-M MDevice 上的 Giga LAN 為藍色膠殼。

若要驅動 Giga LAN，可在 86Duino 環境中使用 Ethernet Library 或 Modbus Library。

Ethernet Library：

- 提供一般網路應用所需的工具，包括：
- 靜態與動態 IP 設定。
- TCP/UDP 通訊。
- 架設 Web Server。

更多細節請參閱 Ethernet Library Documentation。

Modbus Library：

- 可透過 Ethernet 與 Modbus TCP 裝置通訊。
- 適合工業自動化與監控應用。

更多資訊請參閱 Modbus Library Documentation。

4.8.1 Ethernet 通訊

本節介紹 QEC MDevice 的 Ethernet 設定與控制方式。

QEC-PPC-M 系列的 CPU 內建 10/100/1000Mbps LAN 介面，可透過 Ethernet library 存取，無須另外搭配 Arduino Ethernet Shield。此 library 可作為接受連線的 server，也可作為對外連線的 client。在 86Duino Coding 中，此 library 最多支援四個同時連線（傳入、傳出或兩者混合）；後續版本最多可支援 128 個同時連線。

4.8.1.1 範例 1：以 DHCP 取得並印出 IP

本範例使用 Ethernet library 的 DHCP 擴充功能，透過 DHCP 取得 IP 位址並印出。

範例程式碼如下：

```
#include <Ethernet.h>
byte mac[] = { 0x00, 0xAA, 0xBB, 0xCC, 0xDE, 0x02 };
EthernetClient client;

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

  // start the Ethernet connection:
  if (Ethernet.begin(mac) == 0)
    Serial.println("Failed to configure Ethernet using DHCP");

  // print your local IP address:
  Serial.print("My IP address: ");
  for (byte thisByte = 0; thisByte < 4; thisByte++) {
    // print the value of each byte of the IP address:
    Serial.print(Ethernet.localIP()[thisByte], DEC);
    Serial.print(".");
  }
  Serial.println();
}

void loop() {

}
```

4.8.1.2 範例 2：Web Server

一個簡單的 Web Server，用於顯示類比輸入腳位的數值。

範例程式碼如下：

```
#include <Ethernet.h>

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192, 168, 1, 177);
EthernetServer server(80);

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

  // start the Ethernet connection and the server:
  Ethernet.begin(mac, ip);
  server.begin();
  Serial.print("server is at ");
  Serial.println(Ethernet.localIP());
}

void loop() {
  // listen for incoming clients
  EthernetClient client = server.available();
  if (client) {
    Serial.println("new client");
    // an http request ends with a blank line
    boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        Serial.write(c);
        if (c == '\n' && currentLineIsBlank) {
          // send a standard http response header
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          client.println("Connection: close");
          client.println();
          client.println("<!DOCTYPE HTML>");
          client.println("<html>");
          // add a meta refresh tag, so the browser pulls again every 5
seconds:
          client.println("<meta http-equiv=\"refresh\" content=\"5\">");
          // output the value of each analog input pin
          for (int analogChannel = 0; analogChannel < 6; analogChannel++) {
            int sensorReading = analogRead(analogChannel);
            client.print("analog input ");
            client.print(analogChannel);
            client.print(" is ");
            client.print(sensorReading);
            client.println("<br />");
          }
```

```
          client.println("</html>");
          break;
        }
        if (c == '\n') {
          // you're starting a new line
          currentLineIsBlank = true;
        }
        else if (c != '\r') {
          // you've gotten a character on the current line
          currentLineIsBlank = false;
        }
      }
    }
    // give the web browser time to receive the data
    delay(1);
    // close the connection:
    client.stop();
    Serial.println("client disconnected");
  }
}
```

4.8.1.3 範例 3：DHCP Chat Server

一個簡單的伺服器，會將收到的訊息轉發給所有已連線的用戶端。您可以用 telnet 連到裝置的 IP 位址後輸入文字，同時也能在 Serial Monitor 中看到用戶端的輸入內容。

範例程式碼如下：

```
#include <Ethernet.h>

byte mac[] = { 0x00, 0xAA, 0xBB, 0xCC, 0xDE, 0x02 };

IPAddress ip(192,168,1, 177);
IPAddress dnsserver(192,168,1, 1);
IPAddress gateway(192,168,1, 1);
IPAddress subnet(255, 255, 0, 0);

// telnet defaults to port 23
EthernetServer server(23);
boolean gotAMessage = false; // whether or not you got a message from the
client yet

void setup() {
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect.
  }


  // start the Ethernet connection:
  Serial.println("Trying to get an IP address using DHCP");
  if (Ethernet.begin(mac) == 0) {
    Serial.println("Failed to configure Ethernet using DHCP");
    // initialize the ethernet device not using DHCP:
    Ethernet.begin(mac, ip, dnsserver, gateway, subnet);
  }
  // print your local IP address:
  Serial.print("My IP address: ");
  ip = Ethernet.localIP();
  for (byte thisByte = 0; thisByte < 4; thisByte++) {
    // print the value of each byte of the IP address:
    Serial.print(ip[thisByte], DEC);
    Serial.print(".");
  }
  Serial.println();
  // start listening for clients
  server.begin();

}

void loop() {
  // wait for a new client:
  EthernetClient client = server.available();
```

```
  // when the client sends the first byte, say hello:
  if (client) {
    if (!gotAMessage) {
      Serial.println("We have a new client");
      client.println("Hello, client!");
      gotAMessage = true;
    }

    // read the bytes incoming from the client:
    char thisChar = client.read();
    // echo the bytes back to the client:
    server.write(thisChar);
    // echo the bytes to the server as well:
    Serial.print(thisChar);
  }
}
```

4.8.1.4 範例 4：MySQL Connector

本範例示範如何由 QEC MDevice 的 Ethernet 埠連線至 MySQL 伺服器。

更多資訊與文件請參閱 wiki：https://github.com/ChuckBell/MySQL_Connector_Arduino/wiki。

範例程式碼如下：

```
#include <Ethernet.h>
#include <MySQL_Connection.h>

byte mac_addr[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };

IPAddress server_addr(10,0,1,35);  // IP of the MySQL *server* here
char user[] = "root";              // MySQL user login username
char password[] = "secret";        // MySQL user login password

EthernetClient client;
MySQL_Connection conn((Client *)&client);

void setup() {
  Serial.begin(115200);
  while (!Serial); // wait for serial port to connect
  Ethernet.begin(mac_addr);
  Serial.println("Connecting...");
  if (conn.connect(server_addr, 3306, user, password)) {
    delay(1000);
    // You would add your code here to run a query once on startup.
  }
  else
    Serial.println("Connection failed.");
  conn.close();
}

void loop() {

}
```

4.8.2 Modbus TCP 通訊

本節介紹 QEC MDevice 的 Modbus TCP 設定與控制方式。

4.8.2.1 範例 1：簡易 Modbus Master TCP

本範例使用 Ethernet 與 ModbusMaster library，透過 Ethernet 建立 Modbus TCP 通訊。程式會以靜態 IP 位址初始化 Ethernet 連線，並設定 Modbus TCP master 與 ID 為 11 的 slave 裝置互動。

範例程式碼如下：

```
#include <Ethernet.h>
#include <Modbus.h>

ModbusMaster bus;
ModbusMasterNode node;

byte mac[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
IPAddress localIp(192, 168, 1, 101);
IPAddress serverIp(192, 168, 1, 102);

int led = 0;
uint32_t value = 0;

void setup()
{
  Ethernet.begin(mac, localIp);

  /* Modbus TCP Mode via Ethernet. */
  bus.begin(MODBUS_TCP, serverIp);

  /* Slave node initialize. */
  node.attach(11, bus);
}

void loop()
{
  uint16_t reg[2];

  /* Write 1 coil to address 0 of the slave with ID 11. */
  node.writeSingleCoil(0, led);

  /* Write 2 word to holding registers address 16 of the slave with ID 11. */
  reg[0] = value & 0xFFFF;
  reg[1] = (value >> 16) & 0xFFFF;
  node.setTransmitBuffer(0, reg[0]);
  node.setTransmitBuffer(1, reg[1]);
  node.writeMultipleRegisters(16, 2);

  /* Read 2 word from holding registers address 16 of the slave with ID 11.
*/
  node.readHoldingRegisters(16, 2);
```

```
  Serial.print("From Node 1 Holding Register: ");
  value = node.getResponseBuffer(0) | (node.getResponseBuffer(1) << 16);
  Serial.println(value);

  /* Read 1 word from input registers address 2 of the slave with ID 11. */
  node.readInputRegisters(2, 1);
  Serial.print("             Input Register: ");
  Serial.print(node.getResponseBuffer(0));
  Serial.println();

  led = !led;
  value++;

  delay(1000);
}
```

4.9 HMI 設計

本節示範如何在 QEC-PPC-M 裝置（面板電腦系列）上建立基本的人機介面（HMI），可使用 LVGL 函式庫或 86Duino Coding IDE 501+ 內附的 86HMI Editor。

我們假設您已完成快速上手指南前面各節的步驟，包括包裝內容物確認、硬體配置、軟體驅動程式安裝，並已完成 QEC-PPC-M MDevice 的設定。

4.9.1 函式庫說明

LVGL（Light and Versatile Graphics Library）是一套功能強大的開源圖形函式庫，可為嵌入式系統打造美觀且具互動性的圖形使用者介面（GUI）。將 LVGL 函式庫與 86Duino IDE 搭配使用，開發者便能輕鬆且高效率地設計並實作 HMI。

您可運用按鈕、滑桿、圖表、影像等其他 LVGL 元件來擴充與客製化 HMI。LVGL 提供豐富的內建元件與客製化選項，可依您的應用需求打造精緻且具視覺吸引力的 HMI。



關於 LVGL 的更多細節，請參閱 https://docs.lvgl.io/7.11/index.html。

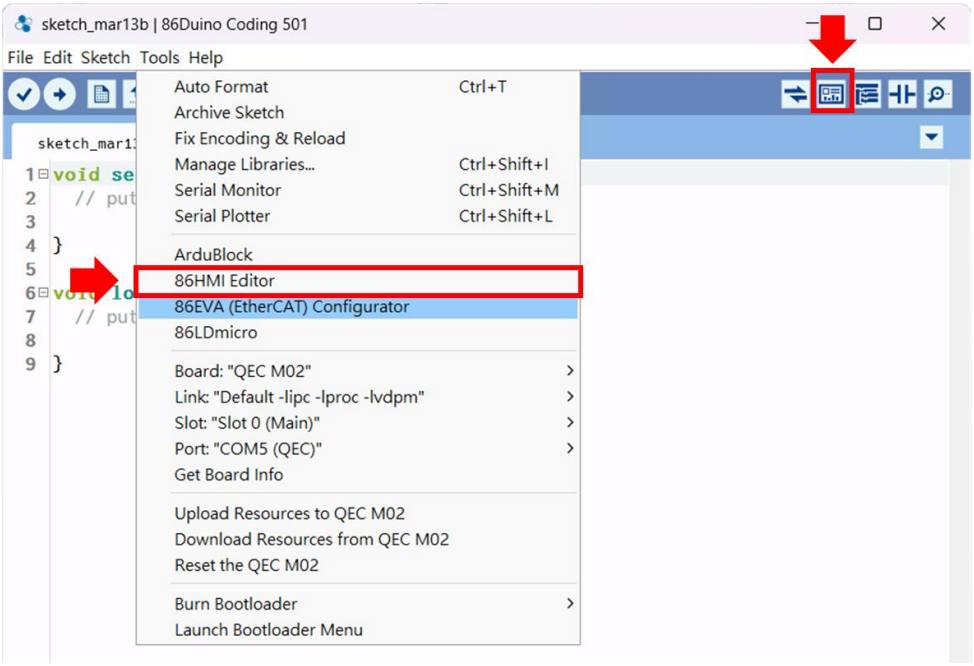
***注意：** 86Duino IDE 501+ 已完成 **LVGL 9.3 版**的測試，使用者可自行匯入 LVGL 函式庫。

4.9.2 使用圖形化 HMI 編輯器：86HMI Editor

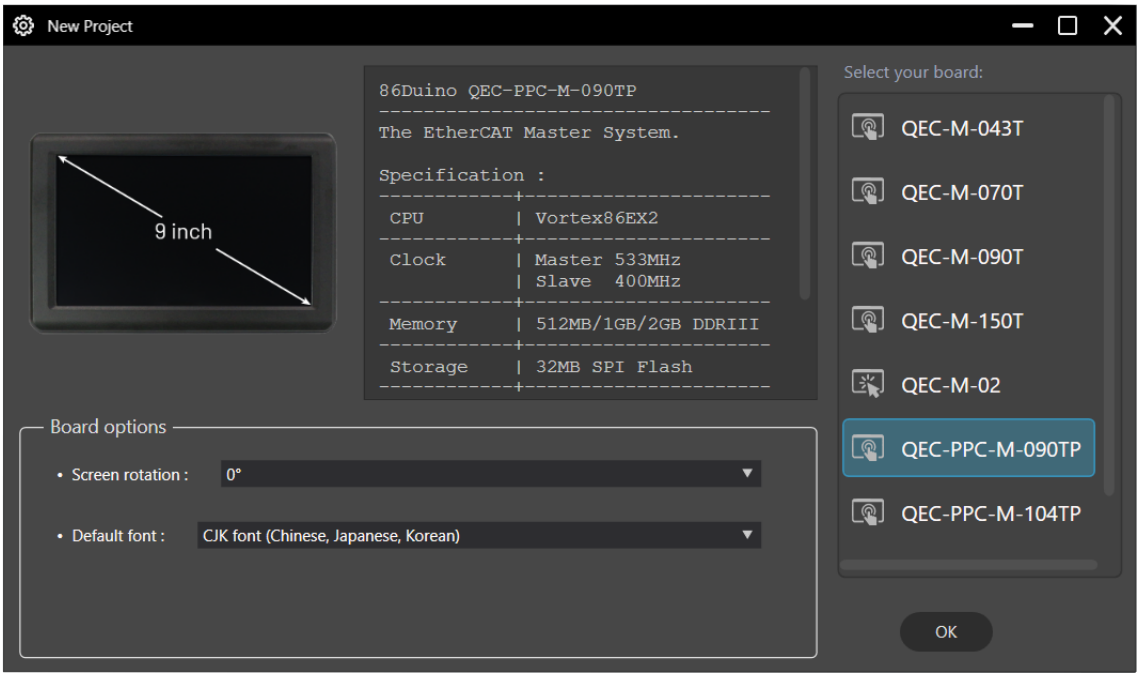
86HMI Editor 是一套簡單易用的 HMI 編輯器，以 LVGL 7.11 版為基礎，可快速建立客製化的 HMI。透過 86HMI Editor 的自動產生程式碼（Auto Code Generation）功能產生 HMI API（應用程式介面），即可達成不需撰寫程式也能建立 HMI 的效果。

以下是使用 86HMI 為 QEC MDevice 設計 UI 的完整步驟：

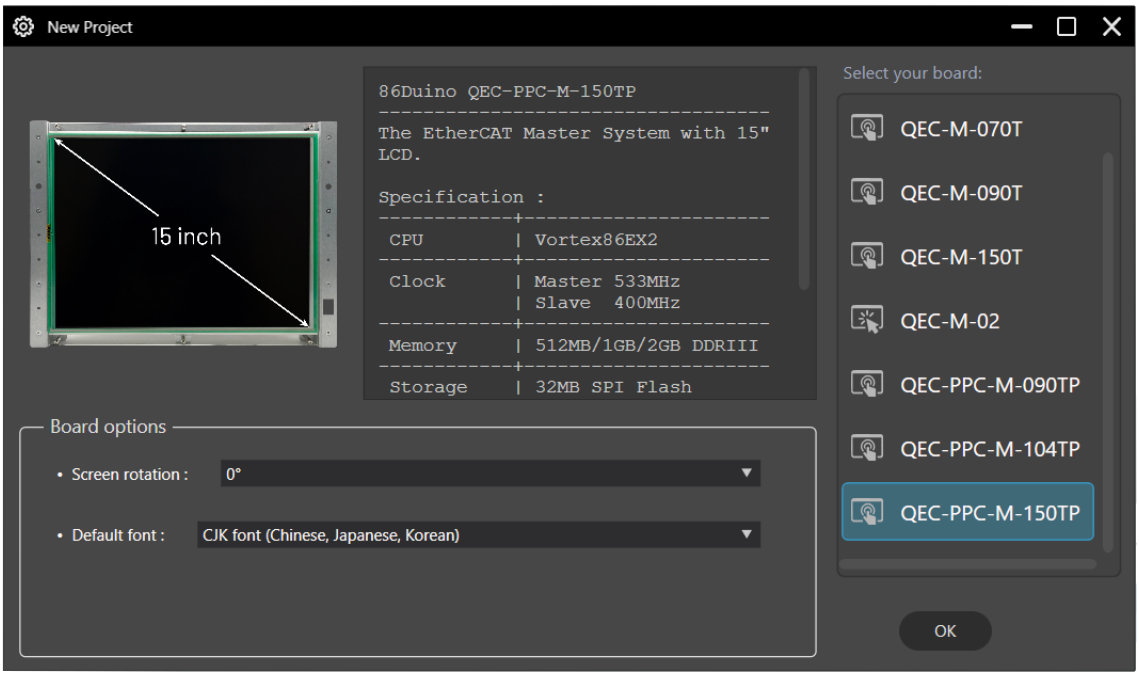
1. 透過 86Duino Coding IDE 501+ 開啟 86HMI 工具。



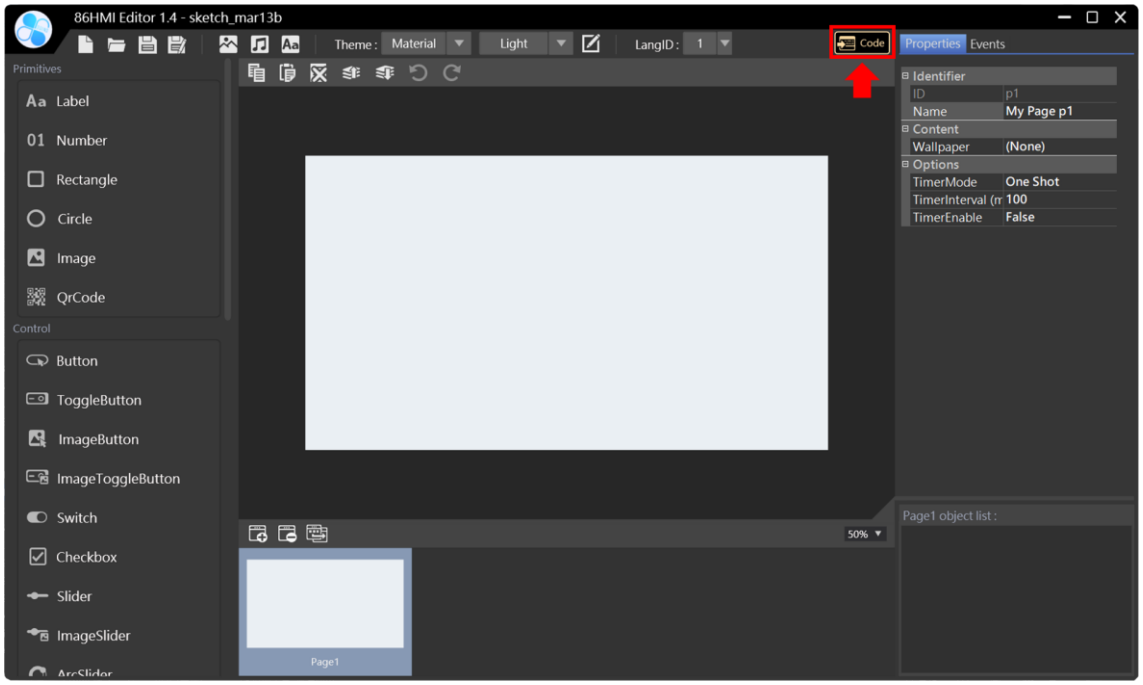
2. 選擇您的 QEC MDevice。QEC-PPC-M-090T：



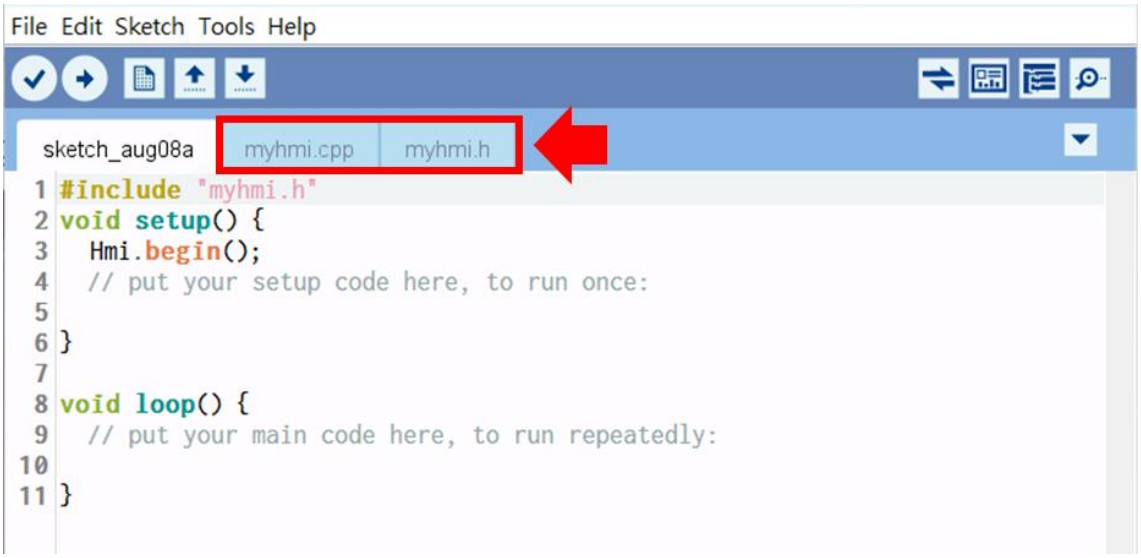
QEC-PPC-M-150T:



3. 接著即可使用左側選單開始設計 UI，並點選 「**Code**」按鈕，自動將原始碼產生回 86Duino IDE。



產生完成後，可在 86Duino IDE 中看到 **myhmi.h** 與 **myhmi.cpp**。



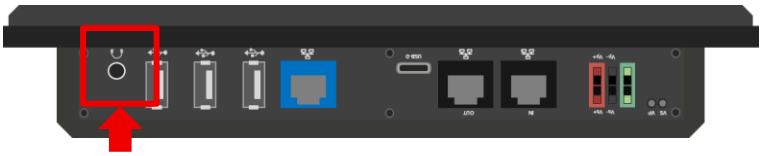
上傳完成後，即可在 QEC MDevice 上看到您剛才設計的使用者介面。

關於 86HMI Editor 的更多細節，請參閱 86HMI 使用手冊：
https://www.qec.tw/86duino/86hmi/。

4.10 音訊使用說明

本節介紹 QEC MDevice 的 HD Audio 控制方式。

QEC-PPC-M-090T 與 QEC-PPC-M-150T 各具備一組 HD Audio 埠，使用者可透過 Audio Library 播放音訊。



此 library 自 86Duino Coding 102 起提供，實作了 Arduino Due 的 Audio library API，可存取 QEC-PPC-M-090T/150T 的板載 HD Audio 介面。

4.10.1 範例 1：簡易音訊播放器

示範如何在 QEC-PPC-M-090T/150T 上使用 Audio library。

本範例示範如何在 QEC-PPC-M-090T/150T 上使用 Audio library，播放存放於 USB 儲存裝置中的 .wav 音訊檔。程式會由檔案讀取音訊資料，透過 Audio library 處理後即時輸出。

範例程式碼如下：

```
#include <SD.h>
#include <Audio.h>

void setup() {
  Serial.begin(9600);
  SD.setBank(USBDISK);
  Serial.print("Initializing SD card...");
  if (!SD.begin(4)) {
    Serial.println(" failed!");
    return;
  }
  Serial.println(" done.");
  Audio.begin(88200, 100);
}

void loop() {
  int count=0;
  File myFile = SD.open("test.wav");
  if (!myFile) {
    Serial.println("error opening test.wav");
    while (true);
  }

  const int S=1024;
  short buffer[S];
```

```
  Serial.print("Playing");

  while (myFile.available()) {
    myFile.read(buffer, sizeof(buffer));
    int volume = 1024;
    Audio.prepare(buffer, S, volume);
    Audio.write(buffer, S);
    count++;
    if (count == 100) {
      Serial.print(".");
      count = 0;
    }
  }
  myFile.close();

  Serial.println("End of file. Thank you for listening!");
  while (true) ;
}
```

4.11 疑難排解

4.11.1 無法順利上傳程式碼

當您無法順利上傳程式碼時，請開啟 86EVA 檢查 QEC EtherCAT MDevice 的環境是否異常。如下圖所示，請嘗試更新 QEC EtherCAT MDevice 的環境，更新內容包含下列三個項目：Bootloader、EtherCAT firmware 與 EtherCAT tool。



以下進一步說明更新的操作方式：

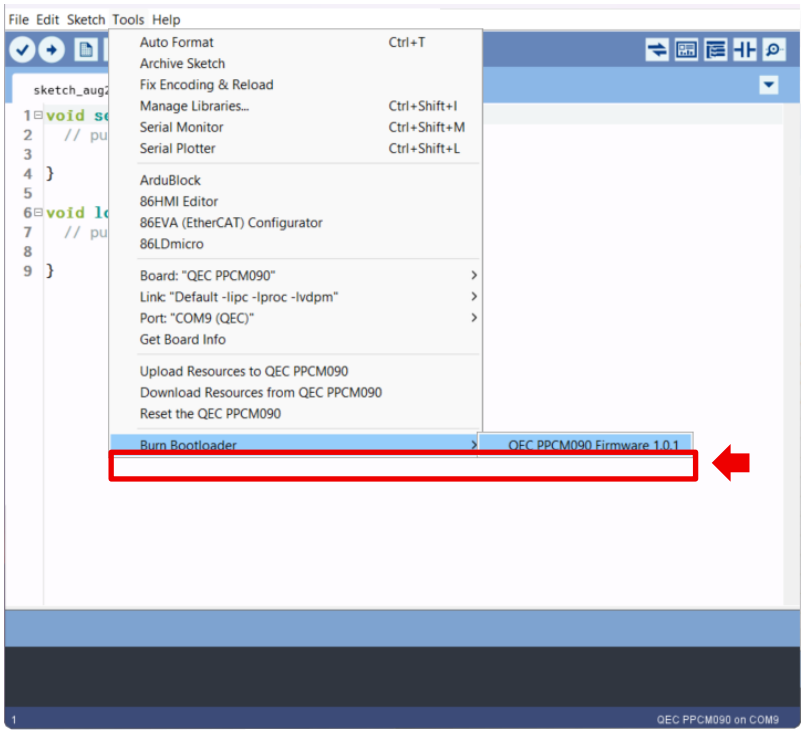
步驟 1：設定 QEC-M

1. 下載並安裝 86Duino IDE 501+（或更新版本）：可由 Software 頁面下載。
2. 連接 QEC MDevice：使用 USB 連接線將 QEC MDevice 連接至電腦。
3. 開啟 86Duino IDE：安裝完成後，開啟 86Duino IDE 軟體。
4. 選擇開發板：在 IDE 選單中選擇「**Tools**」→「**Board**」→「**QEC-PPC-M-090T**」(或您所使用的 QEC MDevice 型號)。
5. 選擇連接埠：在 IDE 選單中選擇「**Tools**」→「**Port**」，並選取 QEC MDevice 所連接的 USB 埠。

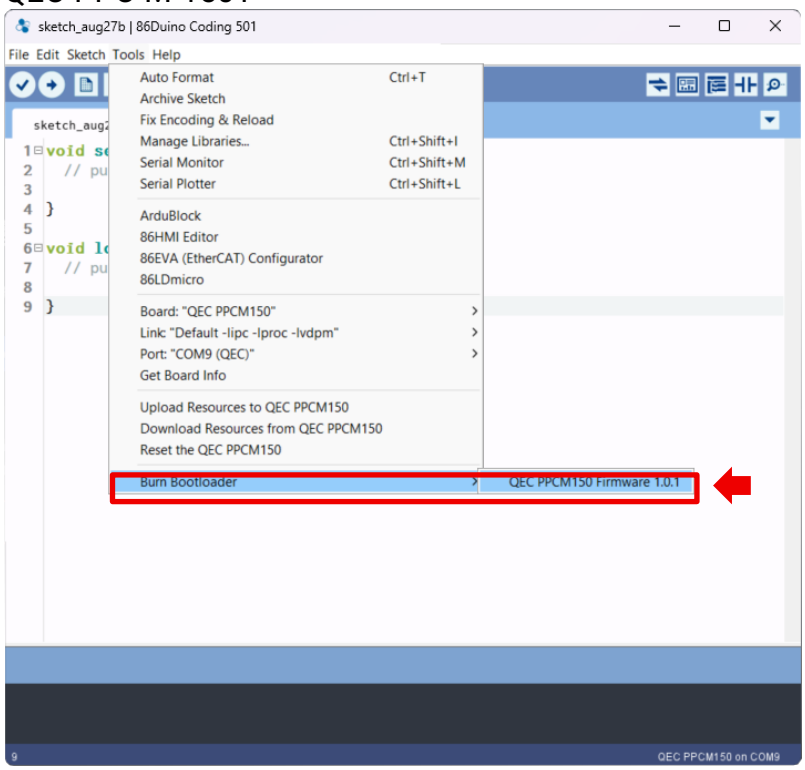
步驟 2：點選「Burn Bootloader」按鈕

連接至您的 QEC MDevice 後，選擇「**Tools**」→「**Burn Bootloader**」，畫面會顯示目前選取的 QEC MDevice 名稱。點選後即開始更新程序，約需 5 至 20 分鐘。

- QEC-PPC-M-090T:



- QEC-PPC-M-150T



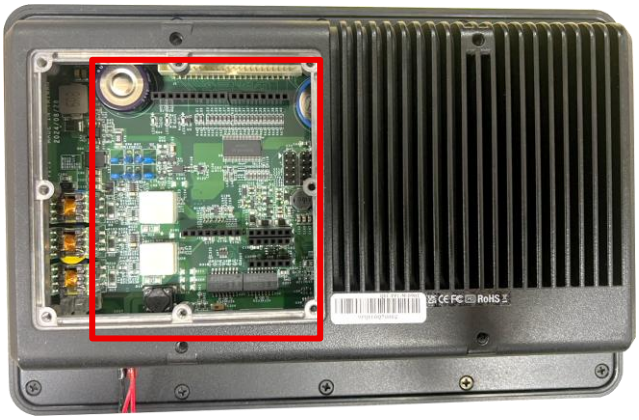
步驟 3：完成更新



完成上述步驟後，您的 QEC-M 便已成功更新至最新版本的開發環境。

4.12 Arduino 腳位使用說明

本節介紹 QEC MDevice 上的 GPIO 腳位配置，包含 Arduino 相容腳位，以及 CAN、SPI、PWM 等 QEC 專屬功能腳位。



QEC 系統採用 86Duino 架構，保留與 Arduino 程式模型的相容性，使用者可使用 `digitalWrite()`、`analogRead()`、`pinMode()` 等標準函式撰寫控制程式。

不過部分腳位提供 QEC 專屬的進階功能，包括硬體 CAN bus、專用 SPI 介面與擴充的 PWM 輸出，這些皆非標準 Arduino 開發板所具備。

為確保正確使用，使用者撰寫軟體時務必參照 86Duino 的腳位對應表。
請參考下方的 86Duino 腳位配置表，確認實體腳位位置與軟體腳位編號之間的正確對應關係。

表：86Duino 腳位配置

連接器	腳位	訊號	86Duino 腳位配置
J5	1	RXD1#	0 (Serial1)
	2	TXD1#	1 (Serial1)
	3	GP00	2
	4	MCM-3	3
	5	GP02	4
	6	MCM-5	5
	7	MCM-6	6
	8	GP05	7
J4	1	GP30	8
	2	MCM-9	9
	3	MCM-10	10
	4	MCM-11	11
	5	GP31	12
	6	MCM-13	13
	7	GND	-
	8	-	-
	9	I2C0_SDA	（線材）
	10	I2C0_SCL	（線材）
J13	6	GP40ADC	14 (A0)
	5	GP41ADC	15 (A1)
	4	GP42ADC	16 (A2)
	3	GP43ADC	17 (A3)
	2	GP56ADC	18 (A4)
	1	GP57ADC	19 (A5)
J14	1	-	-
	2	GP35	23
	3	GP36	24
	4	GP37	25
	5	GND	-
	6	VCC	-
J16	1	CAN1_L	(CAN1)
	2	CAN1_H	(CAN1)
	3	GND	-
	4	CAN0_L	(CAN)
	5	CAN0_H	(CAN)
	6	VCC3	-

連接器	腳位	訊號	86Duino 腳位配置
J7	1	SPI0_DI	(SPI)
	2	VCC	-
	3	SPI0_CLK	(SPI)
	4	SPI0_DO	(SPI)
	5	RESET-	-
	6	GND	-
	7	SPI0_CS	(SPI)
	8	-	-
	9	-	-
	10	-	-
J8	1	SPI1_DI	(SPI1)
	2	VCC	-
	3	SPI1_CLK	(SPI1)
	4	SPI1_DO	(SPI1)
	5	RESET-	-
	6	GND	-
	7	SPI1_CS	(SPI1)
	8	-	-
	9	RS485+	(Serial485 / SerialCOM)
	10	RS485-	(Serial485 / SerialCOM)

4.12.1 GPIO

QEC MDevice 上的 General Purpose Input/Output (GPIO) 腳位，可透過 86Duino Coding IDE 以熟悉的 Arduino 風格函式進行程式控制。

電氣特性：所有 GPxx (GPIO) 與 MCM 腳位的輸出為 **VOH = 3.3 V**（邏輯高準位）與 **VOL = 0 V**（邏輯低準位）。每支腳位的輸出驅動電流**預設為 8 mA，最大為 16 mA**。

您可參考下列 API 函式以進一步了解：

- pinMode()
- digitalWrite()
- digitalRead()

*__注意：__ 請使用 86Duino 腳位對應表中的正確腳位編號。

範例程式碼如下：

```
int ledPin = 13;                // LED connected to digital pin 13

void setup()
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin as output
}

void loop()
{
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(1000);                  // waits for a second
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(1000);                  // waits for a second
}
```

4.12.2 ADC

QEC MDevice 提供多組類比輸入（ADC）通道。預設情況下，`analogRead()` 會將 0 至 3.3 V 的輸入電壓對應為 0 至 1023 的整數值（10-bit 解析度）。呼叫 `analogReadResolution()` 可切換為 11-bit 解析度，對應為 0 至 2047 的數值。在預設的 10-bit 解析度下，每一單位約為 3.3 V / 1024 ≈ 3.2 mV。

您可參考下列 API 函式以進一步了解：

- analogRead()
- analogReference()

*注意： 請使用 86Duino 腳位對應表中的正確腳位編號。

範例程式碼如下：

```
int analogPin = 3;     // potentiometer wiper (middle terminal) connected to
analog pin 3
int val = 0;           // variable to store the value read

void setup()
{
  Serial.begin(9600);          //  setup serial
}

void loop()
{
  val = analogRead(analogPin);    // read the input pin
  Serial.println(val);           // debug value
}
```

4.12.3 TX/RX

QEC MDevice 支援一組 UART 介面 (一對 TX/RX)，遵循標準 UART 協定，可透過 86Duino Coding IDE 中的 `Serial1` 物件存取。

您可參考下列 API 函式以進一步了解：

- Serial

*注意： 請使用 86Duino 腳位對應表中的正確腳位編號。

範例程式碼如下：

```
int incomingByte = 0;   // for incoming serial data

void setup() {
    Serial1.begin(9600);     // opens serial port, sets data rate to 9600 bps
}

void loop() {
    // send data only when you receive data:
    if (Serial1.available() > 0) {
            // read the incoming byte:
            incomingByte = Serial1.read();

            // say what you got:
            Serial1.print("I received: ");
            Serial1.println(incomingByte, DEC);
    }
}
```

4.12.4 RS-485

QEC MDevice 支援一組 RS-485 介面，遵循標準 RS-485 協定，可透過 86Duino Coding IDE 中的 `Serial485` 或 `SerialCOM` 物件存取。

您可參考下列 API 函式以進一步了解：

- Serial
- Modbus Library

*注意： 請使用 86Duino 腳位對應表中的正確腳位編號。

4.12.4.1 序列通訊

使用者可透過 `SerialCOM` 設定 QEC MDevice PPC 系列上的序列埠。

範例程式碼如下：

```
void setup() {
  SerialCOM.begin(115200);
  Serial.begin(115200);
  Serial.println("Start");
}

void loop() {
  SerialCOM.write("hello Serial");
  delay(10);

  for (int i = 0; i < 11; i++) {
    while (SerialCOM.available() < 1);
    Serial.print(SerialCOM.read());
    Serial.print(",");
  }
  Serial.println();
}
```

主要函式：

- `SerialCOM.begin(baud_rate)` ：以指定的鮑率初始化通訊。
- `SerialCOM.read()` ：由序列埠讀取傳入的資料。
- `SerialCOM.write(data)` ：透過序列埠送出資料。

4.12.4.2 Modbus RTU 通訊

若需使用更進階的通訊協定，RS485 介面可搭配 Modbus Library 使用。

86Duino Coding IDE 支援 Modbus 通訊協定。Modbus 是 1979 年發布的工業通訊標準，用於可程式邏輯控制器（PLC）等自動化電子裝置之間的通訊。

Modbus 是以 master/slave 為基礎的協定，由一個 master 節點與網路上的多個 slave 節點通訊，每個節點各有唯一位址。當 master 節點對指定位址送出封包時，只有對應位址的 slave 節點會接收並解析該封包，並依封包內容執行命令與回應。

86Duino 的 Modbus library 具備下列特性：

- 支援 Modbus RTU、TCP 與 ASCII 子協定。
- 可同時作為 Modbus master 與 Modbus slave 節點運作。
- 支援 Modbus gateway 功能。

詳細用法請參閱 86Duino Modbus Library Documentation。

4.12.4.2.1 範例 1：以 Modbus Master RTU 輸出 8-bit 資料

- 將主機端 RS485 的鮑率設為 115200。
- 將 Modbus master 設定為 RTU 模式，並連接至 QEC MDevice 上的主機端 RS485。
- 主迴圈程式：Modbus slave 的 ID 為 30，具 8 個 Coil。先將奇數編號的 Coil 寫入 1、偶數編號的 Coil 寫入 0；間隔 1 秒後，再將奇數編號的 Coil 寫入 0、偶數編號的 Coil 寫入 1，如此持續循環。

範例程式碼如下：

```
#include "Modbus.h"

ModbusMaster bus;

void setup() {
  Serial.begin(3000000);
  SerialCOM.begin(115200);    // Set the Serial baud rate
  bus.begin(MODBUS_RTU, SerialCOM);    // Initialize Modbus RTU
}

void loop() {
  static bool state = false;
  uint16_t coils = state ? 0xAAAA : 0x5555;    // Odd coils 1, even coils 0
and vice versa
  bus.writeMultipleCoils(30, 0, 8, &coils);
  state = !state;
  delay(1000);        // Wait for 1 second
}
```

4.12.4.2.2 範例 2：以 Modbus Master RTU 讀取 8-bit 資料

- 將主機端 RS485 的鮑率設為 115200。
- 將 Modbus master 設定為 RTU 模式，並連接至主機端 RS485。
- 主迴圈程式：Modbus slave 的 ID 為 30。持續讀取第 4 個 Holding Register 的數值，並每隔 1 秒以 `Serial.print` 印出，如此循環。

範例程式碼如下：

```
#include "Modbus.h"

ModbusMaster bus;

void setup() {
  Serial.begin(3000000);
  SerialCOM.begin(115200);    // Set the serial baud rate
  bus.begin(MODBUS_RTU, SerialCOM);    // Initialize Modbus RTU
}

void loop() {
  static uint32_t lastTime = 0;
  uint32_t currentTime = millis();
    // Check if one second has passed
    if (currentTime - lastTime >= 1000) {
      uint16_t data[1];
      uint8_t result = bus.readHoldingRegisters(30, 4, 1, data);
      if (result == 0) {
        Serial.print("Holding Register 4 Value: ");
        Serial.println(data[0]);
      } else {
        // Handle error
      }
      lastTime = currentTime;
      // Update the last time a write/read was attempted
    }
}
```

4.12.5 CAN

QEC MDevice 配備兩組 CAN Bus 介面，可直接與各式工業裝置通訊。這兩組 CAN Bus 可透過 86Duino Coding IDE 中的 `CAN` 與 `CAN1` 物件存取。

透過 86Duino CANBus Library，使用者可用熟悉的 Arduino 風格函式，經由 CAN 物件輕鬆收送 CAN frame。

86Duino 平台所內含的 CAN bus 具備下列特性：

- 支援 CAN 2.0A 與 2.0B 標準
- 支援最高 1 Mbps 的資料傳輸速率
- 支援標準（11-bit）與延伸（29-bit）的 data/remote frame 傳輸
- 提供 3 組 frame buffer
- 完善的錯誤偵測與處理機制

為降低學習與上手的門檻，本 library 的 API 設計高度貼近 Wire library 與 Seeed Studio 的 CAN bus library。以下為本 library 各 API 的功能與用法說明。

您可參考下列 API 函式以進一步了解：

- CANBus Library

*__注意：__ 請使用 86Duino 腳位對應表中的正確腳位編號。

範例程式碼如下：

```
#include <CANBus.h>
unsigned char buf[8] = {0, 1, 2, 3, 4, 5, 6, 7};

void setup() {
    Serial.begin(115200);
    CAN.begin(CAN_500KBPS); // Configure CAN bus transmission speed to
500KBPS
}

void loop() {
    CAN.beginTransmission(0x00, CAN_STDID);    // Set the CAN device ID to
0x00, using standard data frame.
    CAN.write(buf, 8);                        // Sent 8 bytes of data
    CAN.endTransmission();                    // End transmission
    delay(10);
}
```

4.12.6 SPI

QEC MDevice 配備兩組硬體 SPI 介面，可透過 86Duino Coding IDE 中的 `SPI` 與 `SPI1` 物件存取。這些介面可與 ADC、DAC、flash 記憶體、顯示模組等外部周邊進行高速全雙工通訊。

您可參考下列 API 函式以進一步了解：

- SPI library

*注意： 請使用 86Duino 腳位對應表中的正確腳位編號。

範例程式碼如下：

```
// include the SPI library:
#include <SPI.h>

void setup() {
  // initialize SPI:
  SPI.begin();
}

void loop() {
  // go through the six channels of the digital pot:
  for (int channel = 0; channel < 6; channel++) {
    // change the resistance on this channel from min to max:
    for (int level = 0; level < 255; level++) {
      digitalPotWrite(channel, level);
      delay(10);
    }
    // wait a second at the top:
    delay(100);
    // change the resistance on this channel from max to min:
    for (int level = 0; level < 255; level++) {
      digitalPotWrite(channel, 255 - level);
      delay(10);
    }
  }
}

void digitalPotWrite(int address, int value) {
  // take SPI0_CS low to select the chip:
  SPI.SPICS(0);
  delay(100);
  //  send in the address and value via SPI:
  SPI.transfer(address);
  SPI.transfer(value);
  delay(100);
  // take SPI0_CS high to de-select the chip:
  SPI.SPICS(1);
}
```

4.12.6.1 SPI Chip Select (SPI0_CS / SPI1_CS)

在 QEC MDevice 上，chip-select 訊號線 SPI0_CS 與 SPI1_CS 並非以 `digitalWrite()` 控制。

請改用 SPI (SPI0) 與 SPI1 物件的 `SPICS()` 方法來致能與解除 chip-select 訊號：

- `SPICS(0)` — 將 chip-select 訊號線拉為 LOW，以選取目標裝置。
- `SPICS(1)` — 將 chip-select 訊號線拉為 HIGH，以取消選取裝置。

***注意：** 請使用 86Duino 腳位對應表中的正確腳位編號。

以下範例驅動兩顆數位電位器，分別接在 SPI0 與 SPI1 上，並讓各通道的電阻值在最小與最大之間來回循環。

範例程式碼如下：

```
#include <SPI.h>

void setup() {
  // initialize SPI:
  SPI.begin();
  SPI1.begin();
}

void loop() {
  // go through the six channels of the digital pot:
  for (int channel = 0; channel < 6; channel++) {
    for (int level = 0; level < 255; level++) {      // min -> max
      digitalPotWrite(channel, level);
      delay(10);
    }
    delay(100);
    for (int level = 0; level < 255; level++) {      // max -> min
      digitalPotWrite(channel, 255 - level);
      delay(10);
    }
  }
}

void digitalPotWrite(int address, int value) {
  SPI.SPICS(0);            // take SPI0_CS low to select
  SPI.transfer(address);
  SPI.transfer(value);
  SPI.SPICS(1);            // take SPI0_CS high to de-select

  SPI1.SPICS(0);           // take SPI1_CS low to select
  SPI1.transfer(address);
  SPI1.transfer(value);
  SPI1.SPICS(1);           // take SPI1_CS high to de-select
}
```

***注意：** 請使用 86Duino 腳位對應表中的正確腳位編號。SPI0_CS 位於 J7 Pin 7，SPI1_CS 位於 J8 Pin 7。

4.12.7 I2C

QEC MDevice 透過 Wire 物件提供 I2C（Inter-Integrated Circuit）介面，可與感測器、EEPROM、RTC 模組、I/O 擴充器等裝置進行雙線通訊。此 library 可讓您透過 SDA（資料線）與 SCL（時脈線）腳位與 I2C / TWI 裝置通訊。

您可參考下列 API 函式以進一步了解：

- Wire library

*注意： 請使用 86Duino 腳位對應表中的正確腳位編號。

範例程式碼如下：

```
#include <Wire.h>

byte val = 0;

void setup()
{
  Wire.begin(); // join i2c bus
}

void loop()
{
  Wire.beginTransmission(44); // transmit to device #44 (0x2c)
                              // device address is specified in datasheet
  Wire.write(val);              // sends value byte
  Wire.endTransmission();     // stop transmitting

  val++;        // increment value
  if(val == 64) // if reached 64th position (max)
  {
    val = 0;    // start over from lowest value
  }
  delay(500);
}
```

4.12.8 MCM

Multi-Channel PWM Module (MCM) 是 QEC MDevice 的重要硬體特性，可高效且彈性地產生多路 **PWM（Pulse Width Modulation）**訊號，也是 86Duino Coding IDE 中 `analogWrite()` 函式的基礎。

MCM 特別適用於下列應用：

- 馬達轉速控制
- LED 調光
- 伺服訊號產生
- 一般類比輸出模擬

在我們以 1000Hz PWM 頻率實作的 `analogWrite()` 函式中，寫入解析度最高可設為 13。將寫入解析度設為 13 後，即可使用 `analogWrite()` 以 0 至 8191 之間的數值設定 PWM 訊號而不會溢位。

您可參考下列 API 函式以進一步了解：

- analogWrite()
- analogWriteResolution()

*__注意：__ 請使用 86Duino 腳位對應表中的正確腳位編號。

範例程式碼如下：

```
void setup(){
  // open a serial connection
  Serial.begin(9600);
  // make our digital pin an output
  pinMode(11, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(13, OUTPUT);
}

void loop(){
  // read the input on A0 and map it to a PWM pin
  // with an attached LED
  int sensorVal = analogRead(A0);
  Serial.print("Analog Read: ");
  Serial.print(sensorVal);

  // the default PWM resolution
  analogWriteResolution(8);
  analogWrite(11, map(sensorVal, 0, 1023, 0 ,255));
  Serial.print(" , 8-bit PWM value : ");
  Serial.print(map(sensorVal, 0, 1023, 0 ,255));

  // change the PWM resolution to 12 bits
  analogWriteResolution(12);
  analogWrite(12, map(sensorVal, 0, 1023, 0, 4095));
  Serial.print(" , 12-bit PWM value : ");
```

```
  Serial.print(map(sensorVal, 0, 1023, 0, 4095));

  // change the PWM resolution to 4 bits
  analogWriteResolution(4);
  analogWrite(13, map(sensorVal, 0, 1023, 0, 15));
  Serial.print(", 4-bit PWM value : ");
  Serial.println(map(sensorVal, 0, 1023, 0, 15));

  delay(5);
}
```

***注意：** 若您將 `analogWriteResolution()` 設為超過硬體允許範圍的數值，我們的實作會*捨棄*多餘的位元。例如：在 86Duino 上使用 `analogWriteResolution(16)`，傳入 `analogWrite()` 的數值僅會採用前 13 個 bit，最後 3 個 bit 會被捨棄。

若您將 `analogWriteResolution()` 設為低於硬體允許範圍的數值，不足的位元會補零以湊足所需長度。例如：在 86Duino 上使用 `analogWriteResolution(8)`，86Duino 會在 `analogWrite()` 所使用的 8-bit 數值後補上 5 個零位元，以取得所需的 13 bit。

Ch. 5

軟體功能

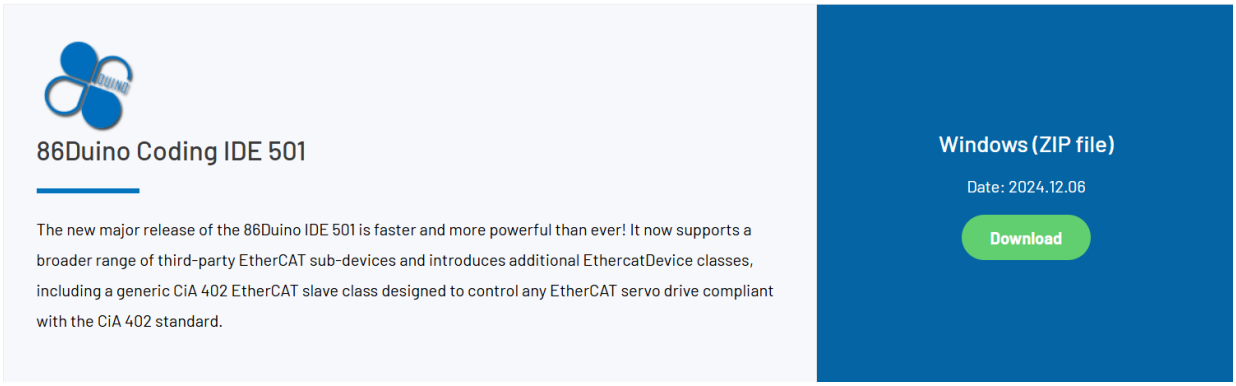
5.1 軟體說明

86Duino Coding IDE 501+ 由 QEC 團隊開發，專為工業控制系統設計，透過開源的類 Arduino 生態系，將簡潔而強大的功能帶入工業應用。



86Duino 整合開發環境（IDE）讓您能輕鬆撰寫程式碼並上傳至 86Duino 開發板，可運行於 Windows、Mac OS X 與 Linux。此開發環境以 Java 撰寫，並以 Arduino IDE、Processing、DJGPP 等開源軟體為基礎。

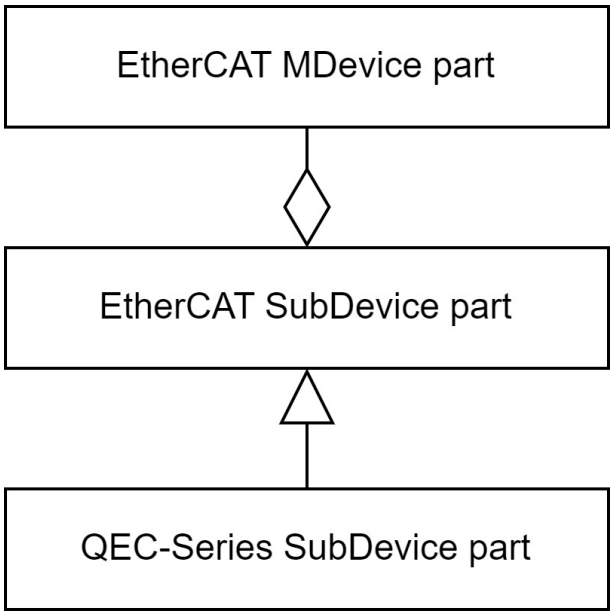
86Duino Coding IDE 501+ 的詳細資訊請造訪 qec.tw。



下載連結：https://www.qec.tw/software/。

5.2 EtherCAT 函式列表

QEC-MDevice 是以 C/C++ 實作的 EtherCAT MDevice library，內含 MDevice、generic SubDevice、CiA 402 SubDevice 等類別，以及專為 QEC 系列 SubDevice 設計的專用類別。這些類別不僅職責劃分明確，也已考量日後的擴充性。



這些類別可分為下列三個部分：

- EtherCAT MDevice
  *EtherCAT MDevice 部分*不僅提供多樣且彈性的 MDevice 組態與操作函式，也提供多種 EtherCAT SubDevice 操作函式，供 EtherCAT SubDevice 部分呼叫使用。
- EtherCAT SubDevice
  *EtherCAT SubDevice 部分*提供通用的 EtherCAT SubDevice 類別，可操作 PDO、CoE、FoE 等功能，並包含 CiA 402 SubDevice 的通用類別。
- QEC-Series SubDevice
  *QEC-Series SubDevice 部分*提供 ICOP QEC 系列 SubDevice 的專用函式，讓使用者能以更友善、更精簡的方式撰寫程式。

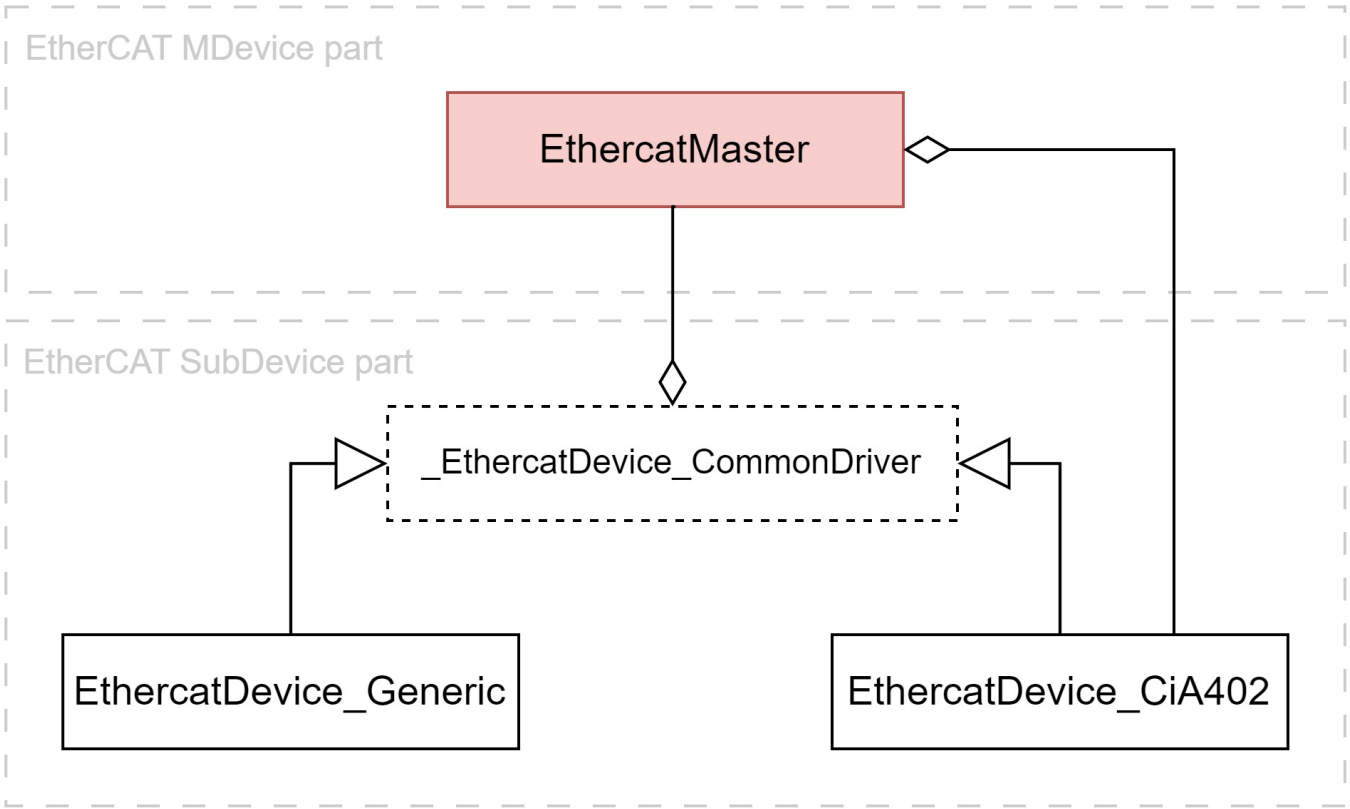
下列清單介紹 QEC MDevice 的 EtherCAT Library API 函式。
API 函式的詳細說明請參閱 EtherCAT Library API User Manual。

5.2.1 EtherCAT MDevice

EtherCAT MDevice 部分不僅提供多樣且彈性的 MDevice 組態與操作函式，也提供多種 EtherCAT SubDevice 操作函式，供 EtherCAT SubDevice 部分呼叫使用。

EthercatMaster 是 EtherCAT MDevice 部分中唯一的類別，是與 EtherCAT firmware 溝通的關鍵橋樑。在雙系統通訊方面，它負責通訊介面初始化、週期性交換 process data、處理非週期性傳輸介面，以及管理中斷事件；在 API 方面，它提供 MDevice 初始化、MDevice 控制，以及存取 SubDevice 資訊的相關函式。

EtherCAT MDevice 部分與 EtherCAT SubDevice 部分之間的主要類別關係為關聯（association），其中 EtherCAT SubDevice 部分相依於 EtherCAT MDevice 部分。EthercatMaster 的類別關係如下圖所示：



- EthercatMaster 與 _EthercatDevice_CommonDriver 之間為關聯關係，其中 _EthercatDevice_CommonDriver 相依於 EthercatMaster。
- EthercatMaster 與 EthercatDevice_CiA402 之間為關聯關係，其中 EthercatMaster 相依於 EthercatDevice_CiA402。

函式：

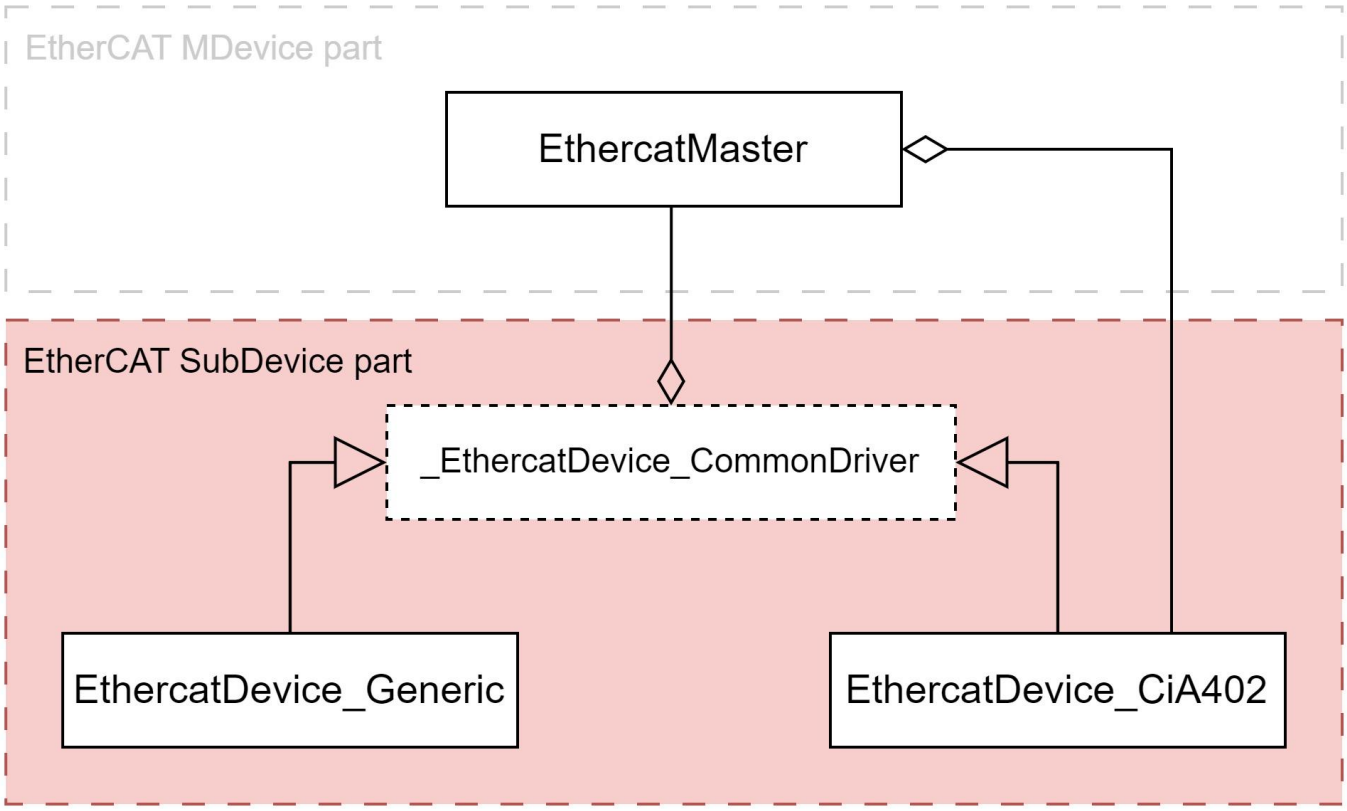
函式名稱	說明	可於 Callback 中呼叫
初始化相關函式		
begin()	初始化 EtherCAT MDevice。	
end()	解除初始化 EtherCAT MDevice。	
isRedundancy()	檢查 EtherCAT MDevice 是否已啟用 cable redundancy。	O
libraryVersion()	取得 EtherCAT MDevice library 版本。	O
firmwareVersion()	取得 EtherCAT firmware 版本。	O
readSettings()	讀取目前的 EtherCAT MDevice 設定。	
saveSettings()	儲存 EtherCAT MDevice 設定。	
控制相關函式		
start()	啟動 EtherCAT MDevice。	
stop()	停止 EtherCAT MDevice。	
update()	更新 process data 並處理非週期性命令。	O
setShiftTime()	設定 DC-Synchronous 模式的 Global Shift Time。	
getShiftTime()	取得 DC-Synchronous 模式的 Global Shift Time。	O
getSystemTime()	取得目前週期的系統時間。	O
getWorkingCounter()	取得目前週期的 working counter。	O
getExpectedWorkingCounter()	取得預期的 working counter。	O
Callback 相關函式		
attachCyclicCallback()	註冊 cyclic callback。	
detachCyclicCallback()	取消註冊 cyclic callback。	
attachErrorCallback()	註冊 error callback。	
detachErrorCallback()	取消註冊 error callback。	
attachEventCallback()	註冊 event callback。	
detachEventCallback()	取消註冊 event callback。	
errGetCableBrokenLocation1()	於 error callback 中取得斷線位置 1。	O[1]
errGetCableBrokenLocation2()	於 error callback 中取得斷線位置 2。	O[1]
evtGetMasterState()	於 event callback 中取得 EtherCAT MDevice 狀態。	O[2]
SubDevice 資訊相關函式		
getSlaveCount()	取得網路上的 SubDevice 數量。	O
getVendorID()	取得指定 SubDevice 的 vendor ID。	O
getProductCode()	取得指定 SubDevice 的 product code。	O
getRevisionNumber()	取得指定 SubDevice 的 revision number。	O
getSerialNumber()	取得指定 SubDevice 的 serial number。	O
getAliasAddress()	取得指定 SubDevice 的 alias address。	O
getSlaveNo()	在網路上尋找符合條件之 EtherCAT SubDevice 的序號。	O

- 註 1：此函式僅能在 error callback 中呼叫。
- 註 2：此函式僅能在 event callback 中呼叫。

5.2.2 EtherCAT SubDevice

EtherCAT SubDevice 部分提供通用的 EtherCAT SubDevice 類別，可操作 PDO、CoE、FoE 等功能，並包含 CiA 402 SubDevice 的通用類別。

EtherCAT SubDevice 部分與 EtherCAT MDevice 部分之間的主要類別關係為關聯（association），其中 EtherCAT SubDevice 部分相依於 EtherCAT MDevice 部分。如下圖所示，_EthercatDevice_CommonDriver 與 EthercatMaster 之間存在關聯關係。



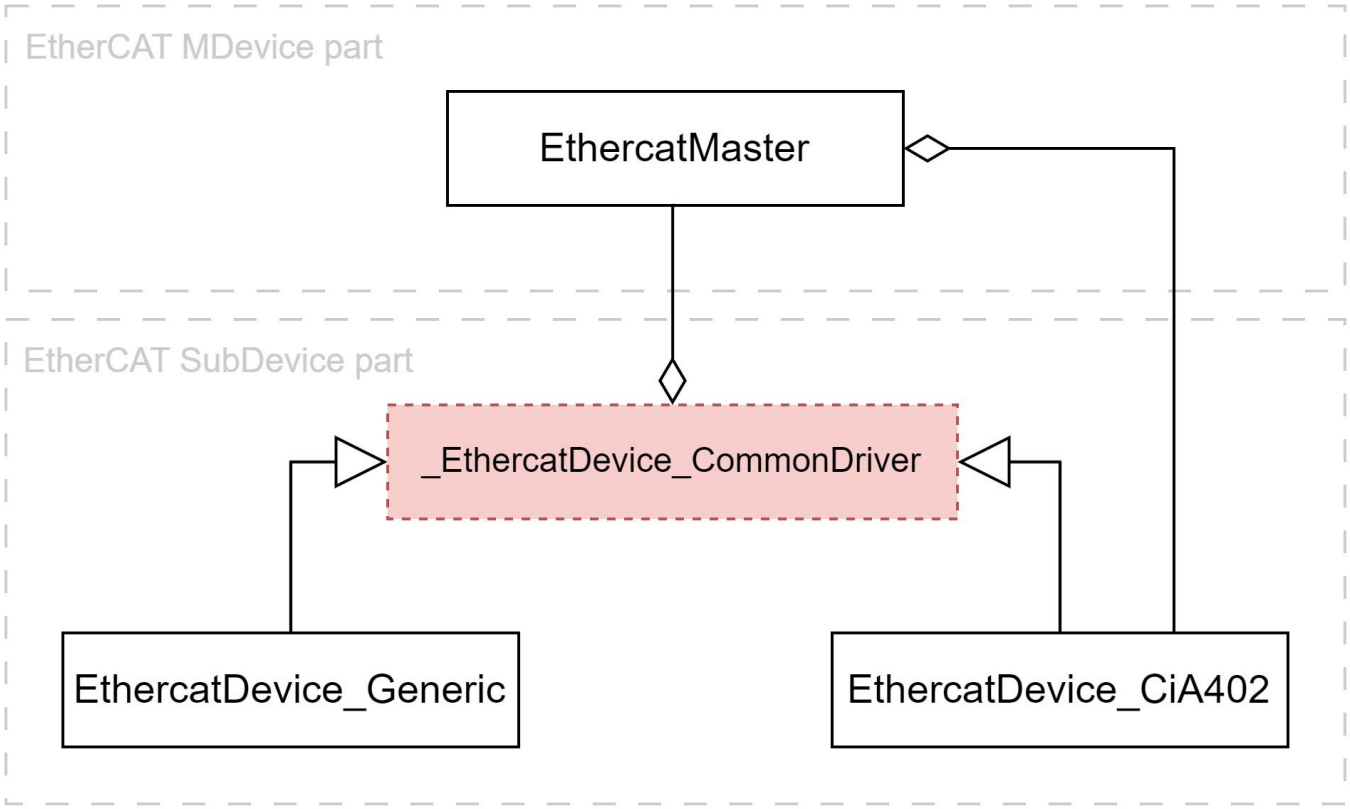
類別：

- _EthercatDevice_CommonDriver
- EthercatDevice_Generic
- EthercatDevice_CiA402

5.2.2.1 _EthercatDevice_CommonDriver

_EthercatDevice_CommonDriver 是一個抽象類別，除了具備存取 SubDevice 資訊的函式外，也提供 PDO、SII、CoE、FoE、DC 等多種 EtherCAT 功能的存取方式。所有 EtherCAT SubDevice 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 之間為關聯關係，其中 _EthercatDevice_CommonDriver 相依於 EthercatMaster。
- 所有其他 EtherCAT SubDevice 類別皆繼承自 _EthercatDevice_CommonDriver。

警告： *禁止使用此類別宣告物件。*

函式：

函式名稱	說明	可於 Callback 中呼叫
SubDevice 資訊相關函式		
getVendorID()	取得 vendor ID。	O
getProductCode()	取得 product code。	O
getRevisionNumber()	取得 revision number。	O
getSerialNumber()	取得 serial number。	O
getAliasAddress()	取得 alias address。	O
getSlaveNo()	取得在 EtherCAT 網路上的序號 ID。	O
getDeviceName()	取得裝置名稱。	O
getMailboxProtocol()	取得支援的 mailbox 協定類型。	O

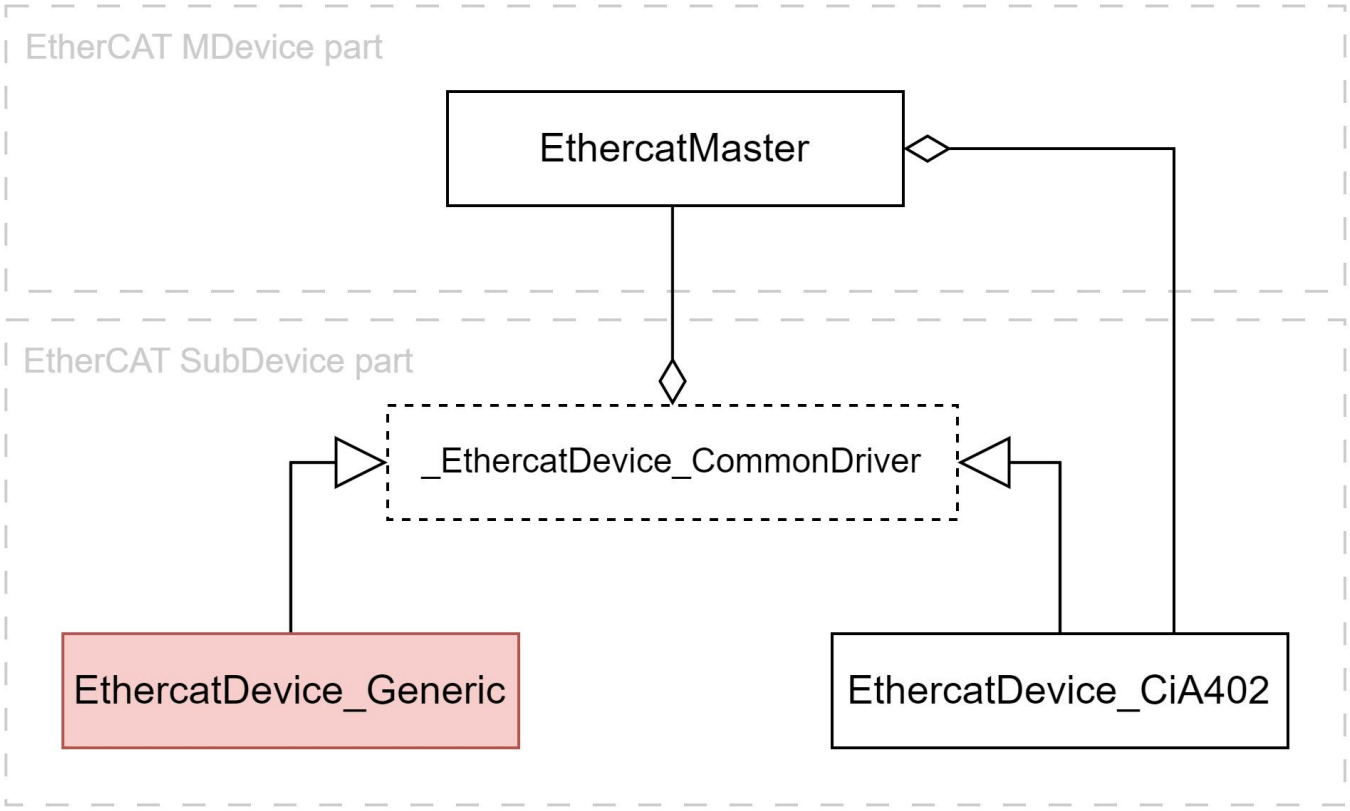
getCoEDetails()	取得 CoE 支援的詳細資訊。	O
getFoEDetails()	取得 FoE 支援的詳細資訊。	O
getEoEDetails()	取得 EoE 支援的詳細資訊。	O
getSoEChannels()	取得支援的 SoE 通道數量。	O
isSupportDC()	檢查 EtherCAT SubDevice 是否支援 DC。	O
PDO 存取函式		
pdoBitWrite()	寫入 1-bit 輸出 process data。	O
pdoBitRead()	讀取 1-bit 輸入 process data。	O
pdoGetOutputBuffer()	取得輸出 process data 的記憶體指標。	O
pdoGetInputBuffer()	取得輸入 process data 的記憶體指標。	O
pdoWrite()	寫入多個 byte 的輸出 process data。	O
pdoWrite8()	寫入 8-bit 輸出 process data。	O
pdoWrite16()	寫入 16-bit 輸出 process data。	O
pdoWrite32()	寫入 32-bit 輸出 process data。	O
pdoWrite64()	寫入 64-bit 輸出 process data。	O
pdoRead()	讀取多個 byte 的輸入 process data。	O
pdoRead8()	讀取 8-bit 輸入 process data。	O
pdoRead16()	讀取 16-bit 輸入 process data。	O
pdoRead32()	讀取 32-bit 輸入 process data。	O
pdoRead64()	讀取 64-bit 輸入 process data。	O
CoE 通訊函式		
sdoDownload()	將多個 byte 的資料寫入物件。	
sdoDownload8()	將 8-bit 數值寫入物件。	
sdoDownload16()	將 16-bit 數值寫入物件。	
sdoDownload32()	將 32-bit 數值寫入物件。	
sdoDownload64()	將 64-bit 數值寫入物件。	
sdoUpload()	由物件讀取多個 byte 的資料。	
sdoUpload8()	由物件讀取 8-bit 數值。	
sdoUpload16()	由物件讀取 16-bit 數值。	
sdoUpload32()	由物件讀取 32-bit 數值。	
sdoUpload64()	由物件讀取 64-bit 數值。	
getODlist()	取得 object dictionary 中既有物件的清單。	
getObjectDescription()	取得物件的 object description。	
getEntryDescription()	取得物件的 entry description。	
FoE 通訊函式		
readFoE()	由 EtherCAT SubDevice 讀取檔案。	
writeFoE()	將檔案寫入 EtherCAT SubDevice。	
DC 組態函式		
setDc()	設定 DC 參數。	
SII EEPROM 存取函式		
writeSII()	將多個 byte 的資料寫入 SII EEPROM。	
writeSII8()	將 8-bit 數值寫入 SII EEPROM。	
writeSII16()	將 16-bit 數值寫入 SII EEPROM。	

writeSII32()	將 32-bit 數值寫入 SII EEPROM。	
readSII()	由 SII EEPROM 讀取多個 byte 的資料。	
readSII8()	由 SII EEPROM 讀取 8-bit 數值。	
readSII16()	由 SII EEPROM 讀取 16-bit 數值。	
readSII32()	由 SII EEPROM 讀取 32-bit 數值。	
初始化相關函式		
attach()	初始化此 EtherCAT SubDevice 類別的物件。	
detach()	解除初始化此 EtherCAT SubDevice 類別的物件。	

5.2.2.2 EthercatDevice_Generic

EthercatDevice_Generic 是通用的 EtherCAT SubDevice 類別，可用於控制所有 EtherCAT SubDevice，包括存取 SubDevice 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。

父類別

- _EthercatDevice_CommonDriver

函式

函式名稱	說明	可於 Callback 中呼叫
初始化相關函式		
attach()	初始化此 EtherCAT SubDevice 類別的物件。	
detach()	解除初始化此 EtherCAT SubDevice 類別的物件。	

5.2.2.3 EthercatDevice_CiA402

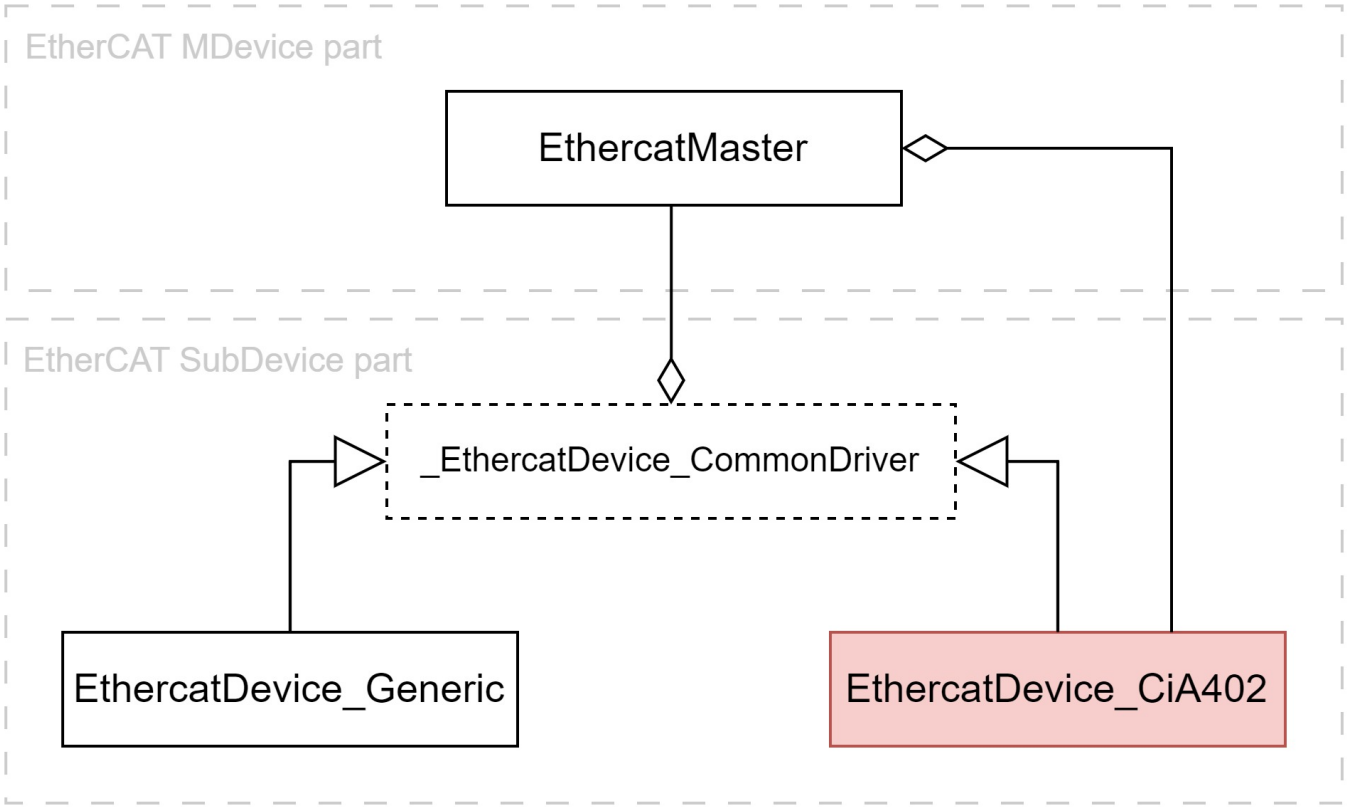
EthercatDevice_CiA402 是通用的 CiA 402 EtherCAT SubDevice 類別，可用於控制任何支援 CiA 402 標準的 EtherCAT 伺服驅動器。它提供常用 CiA 402 物件的存取函式，以及數種 CiA 402 操作模式與功能群組的操作函式，包括：

- **操作模式**
  - 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**
  - Touch Probe

關於 CiA 402 的詳細資訊，請參閱下列文件：

- CiA Draft Standard 402: CANopen device profile drives and motion control
- ETG.6010 Implementation Directive for CiA402 Drive Profile
- 目前所使用之 CiA 402 驅動器裝置的使用手冊

EthercatDevice_CiA402 的類別關係如下圖所示：



- *EthercatDevice_CiA402* 繼承自 *_EthercatDevice_CommonDriver*。

父類別

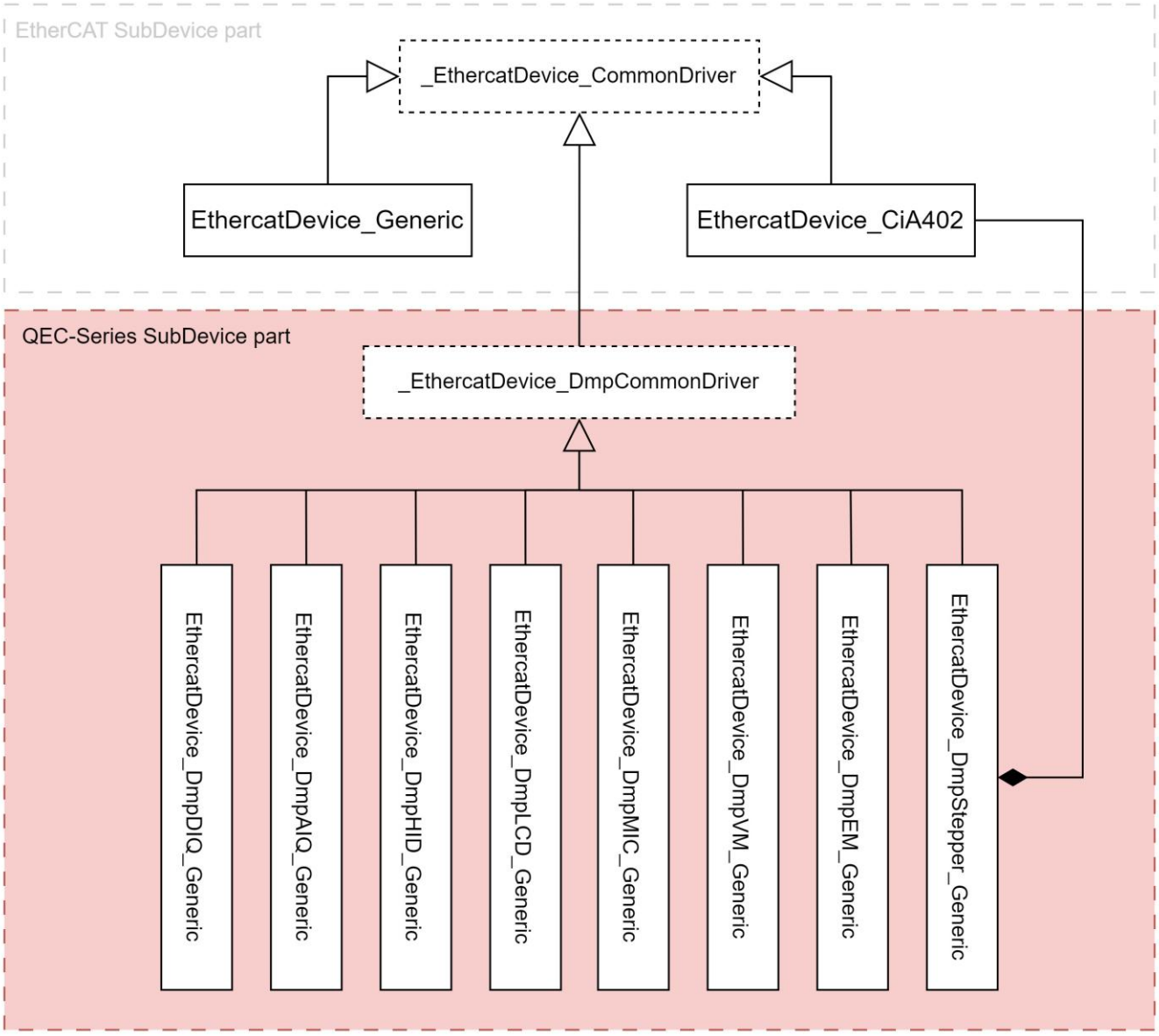
- _EthercatDevice_CommonDriver

關於 CiA 402 API 的詳細資訊，請參閱 EtherCAT Library API User Manual - QEC。

5.2.3 QEC 系列 SubDevice

QEC-Series SubDevice 部分提供 ICOP QEC 系列 SubDevice 的專用函式，讓使用者能以更友善、更精簡的方式撰寫程式。

QEC-Series SubDevice 部分與 EtherCAT SubDevice 部分之間的主要類別關係為關聯（association），其中 QEC-Series SubDevice 部分相依於 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_DmpHID_Generic
- EthercatDevice_DmpLCD_Generic
- EthercatDevice_DmpStepper_Generic

5.3 延伸資源

若您想進一步了解 86Duino IDE 提供的函式庫，或深入研究 86Duino 程式語言的細節，請參閱下列連結：

- **EtherCAT Library API User Manual**： QEC MDevice 是相容於 86Duino Coding IDE 501+ 的 EtherCAT MDevice，可在 EtherCAT MDevice 與 EtherCAT SubDevice 之間提供即時通訊。詳細資訊請參閱 https://www.qec.tw/ethercat/api/ethercat-library-api-user-manual/。

- **86Duino IDE Libraries**： 可於 https://www.qec.tw/86duino/libraries/ 查閱 86Duino IDE 支援的完整函式庫清單，以及詳細文件與範例。

- **86Duino Language Reference**： 可於官方的 86Duino Language Reference（https://www.qec.tw/86duino/86duino-language-reference/）了解 86Duino 程式語言的語法、函式與使用方式。

無論是初學者或資深開發者，這些資源都能為使用 86Duino 平台提供實用的參考。善用這些連結，您將能充分發揮 86Duino IDE 的潛力，打造具創新性的專案。

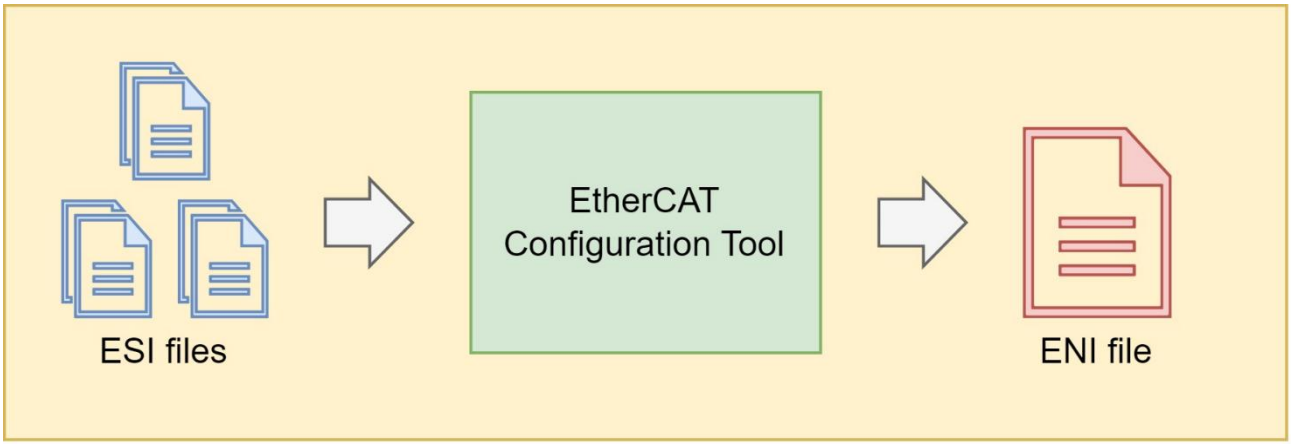
祝您在 86Duino IDE 上開發愉快！

86Duino reference 的文字內容修改自 Arduino reference，並依 Creative Commons Attribution-ShareAlike 3.0 License 授權。reference 中的程式碼範例則釋出至公共領域。

附錄

A1. 關於 86Duino IDE 的 ENI 組態

EtherCAT Network Information (ENI) 包含設定 EtherCAT 網路所需的必要資訊。這份以 XML 為基礎的檔案，記錄了 EtherCAT MDevice 的一般資訊，以及所有連接至該 EtherCAT MDevice 之 SubDevice 的組態。EtherCAT 組態工具會讀取 ESI 檔案，或對網路進行線上掃描以取得所有 SubDevice，接著使用者可設定 PDO mapping、啟用 DC 等相關 EtherCAT 設定，並匯出 ENI 檔案。



EtherCAT Technology Group 規定，EtherCAT MDevice 軟體在 Network Configuration 項目中至少須支援下列其中一種方式：線上掃描（Online Scanning）或讀取 ENI（Reading ENI）。本 library 兩者皆支援。不過在讀取 ENI 時，目前僅會由 ENI 檔案中擷取部分資訊用於網路組態。

擷取的資訊包括下列項目：

EtherCATConfig : Config : SubDevice : Info

- 元素
  - VendorId
  - ProductCode
- 屬性
  - Identification : Value
- 用途

  用於檢查網路上的 EtherCAT SubDevice 是否與 ENI 檔案所指定的 SubDevice 相符，檢查規則如下：
  - 檢查 ENI 檔案中的 SubDevice 數量是否與網路上的 SubDevice 數量相符。
  - 對於 ENI 檔案中具有 Identification: Value 屬性的 SubDevice，檢查網路上是否存在 Alias Address 與 Identification: Value 屬性相符，且 Vendor ID 與 Product Code 亦相符的 SubDevice。若存在，即表示比對成功。
  - 對於具有 Identification: Value 屬性但比對失敗的 SubDevice，或未具備此屬性的 SubDevice，則檢查網路上相同序號之 SubDevice 的 Vendor ID 與 Product Code 是否相符。

EtherCATConfig : Config : SubDevice : Mailbox

- 元素
  - Send : MailboxSendInfoType : Start
  - Recv : MailboxRecvInfoType : Start
- 用途

  用於設定 EtherCAT SubDevice 的 mailbox Physical Start Address。

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 EtherCAT state machine 切換至 Pre-Operational 狀態後，於 `EthercatMaster::begin()` 中執行 EtherCAT SubDevice 的 CoE 初始化命令。

EtherCATConfig : Config : SubDevice : ProcessData

- 元素
  - Recv : BitLength
  - Send : BitLength
- 用途

  提供 EtherCAT SubDevice 輸出與輸入 process data 的 bit 長度給 firmware，作為相關組態之用。

EtherCATConfig : Config : SubDevice : ProcessData : Sm

- 元素
  - SyncManagerSettings : StartAddress
  - SyncManagerSettings : ControlByte
  - SyncManagerSettings : Enable
- 用途

  用於設定 EtherCAT SubDevice 之 process data 的 Sync Manager 暫存器。

EtherCATConfig : Config : SubDevice : DC

- 元素
  - CycleTime0
  - CycleTime1
  - ShiftTime
- 用途

  用於設定 EtherCAT SubDevice 的 DC 參數。

保固

本產品自購買日起保固一年，保證於此期間內維持良好運作狀態。若本產品於保固期間內發生功能異常，本公司將依下列條款，選擇免費更換或維修。本保固不適用於因不當使用、擅自改裝、意外或天災所造成之損壞。對於因使用、誤用本產品或無法使用本產品所導致的任何損害、利潤損失、儲蓄損失，或其他附帶性、衍生性損害，供應商概不負責；供應商亦不對任何其他相關方所提出之請求負責。退回商品前，必須先取得供應商的退貨授權；您可致電或傳真供應商，申請退貨授權編號（RMA number）。退回商品時，請務必附上清楚的問題描述。

本手冊中出現的所有商標，均為其各自所有人之註冊商標。所有規格如有變更，恕不另行通知。
©ICOP Technology Inc. 2026