CANopen 僕端(Slave)設備

CAN-2000C 系列

使用手冊

保固條款

所有由泓格科技製造的產品，泓格科技皆提供對產品本身的一年保固，保固期由本公司交貨給原始訂購者的當天開始起算。

注意事項

泓格科技不對因使用本產品所引起的損害作任何的擔保，並保留在未公告的前提下，對本文件隨時進行修訂的權利。由泓格科技提供的這份文件被認定是正確且可信賴的，然而，泓格科技並不對這份文件的使用作任何的擔保，也不對因為使用這份文件所引起的違反專利或對第三方的侵權負任何責任。

版權

商標

目錄

1 介紹

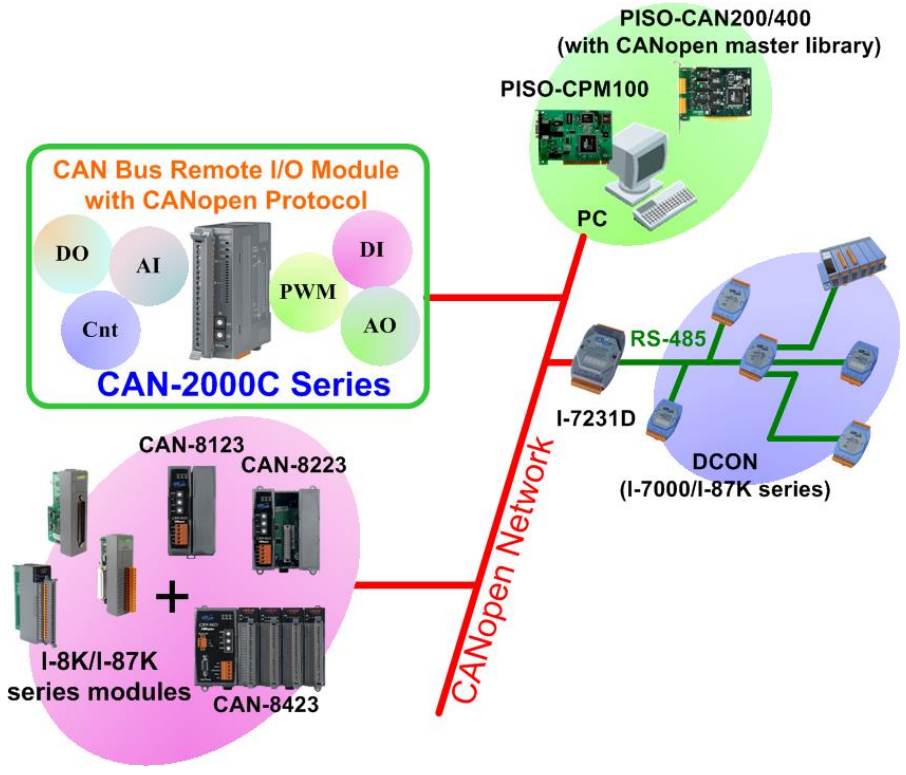
1.1 概述

CANopen，是一種基於智能領域匯流排(intelligent field bus，如CAN bus)之上的通訊協定。其被用來發展具備高度彈性組態能力的標準嵌入式網路。

CANopen為了因應不同需求，提供了多種標準化的通訊物件，像是適合用來傳輸即時(real-time)資料的Process Data Objects(PDO)、適合用來傳輸組態資料的Service Data Objects(SDO)、可進行網路管理的Network Management Objects(如NMT訊息與錯誤控制)，以及其他具有特殊功能的通訊物件(如Time Stamp、SYNC與EMCY訊息)等等

現今，CANopen被使用在各種不同的應用領域，像是醫療設備、工程車輛、航海電子、公眾傳輸與建築自動化等等。

CAN-2000C 系列模組是 CANopen 的僕端模組，遵循著 CiA DS-301 v4.02 及 DS-401 v2.1 的規範。同時支援大量的功能例如動態 PDO、EMCY 物件、錯誤輸出值、同步循環和同步非循環等…CAN-2000C 系列模組的應用架構如下:



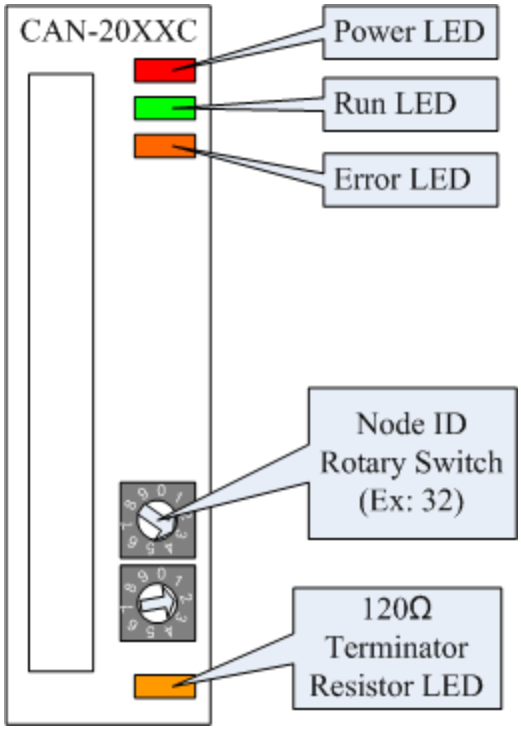
1.2 CAN-2000C 硬體特徵

- 內建看門狗(Watchdog)
- 電源指示燈(PWR LED)，運行指示燈(RUN LED)，錯誤指示燈(ERR LED)
- 終端電阻指示燈(Terminal resister LED)
- 可利用跳線器決定啟用與否的 120Ω 終端電阻
- CAN bus 介面: ISO 11898-2，5-pin 螺旋式終端(screw terminal)
- 電源輸入: +10V_{DC} ~ +30V_{DC}
- 操作溫度:-25°C ~ +75°C
- 儲存溫度:-30°C ~ +80°C
- 濕度: 10% ~ 90%

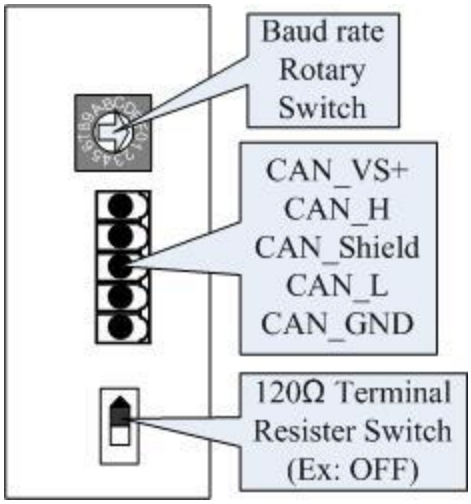
1.3 CAN-2000C 共同特色

- NMT: 支援做為 NMT 的僕端(Slave)
- 錯誤控制協定: 支援節點守衛和心跳事件協定
- 節點 ID:可透過旋鈕來調整 (1 ~ 99)
- SDO 數目: 1 server, 0 client
- PDO 數目: RxPDO 和 TxPDO 各支援 10 組
- PDO 模式: 事件觸發、遠端要求、循環與非循環 SYNC
- 支援動態 PDO 映射
- 緊急訊息：支援緊急訊息(EMCY message)的發送
- 支援“儲存(Save)”和“載入(Load)”命令，來儲存 I/O 的設定值或恢復 I/O 的預設值。
- CANopen 版本: DS-301 v4.02
- 裝置描述文件: DS-401 v2.1
- 支援鮑率：10K、20K、50K、125K、250K、500K、800K 以及 1M bps。
- 狀態指示燈: PWR LED、RUN LED 和 ERR LED

2 硬體規格



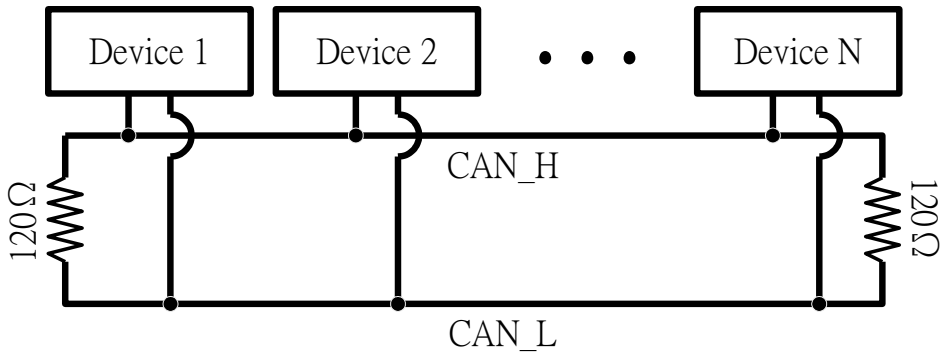
(頂視圖)



(底視圖)

2.1 連線

為了讓信號反射(reflection)在 CAN bus 上的影響降到最低，CAN bus 在網路的兩端必須要分別安裝一個終端電阻，如下圖所示: 根據 ISO 11898-2 的規範，在下圖中每一個終端電阻應為 120Ω (或介於 108Ω~132Ω)。接線本身的電阻值應小於 70 mΩ/m。在開始架設 CAN bus 前，使用者應先檢查匯流排上的電阻值是否正常。



此外，為了將長距離傳輸所引起的電壓損耗降到最低，實際使用的終端電阻值應比ISO 11898-2 所規範的再高一些。下表的數據可作為在架設CAN bus 時的參考。

匯流排長度 (meter)	匯流排纜線參數		終端電阻 (Ω)
	長度與電阻之關係 (mΩ/m)	橫截面 (mm^2)	
0~40	70	0.25(23AWG)~ 0.34mm^2(22AWG)	124 (0.1%)
40~300	< 60	0.34(22AWG)~ 0.6mm^2(20AWG)	127 (0.1%)
300~600	< 40	0.5~0.6mm^2 (20AWG)	150~300
600~1K	< 20	0.75~0.8mm^2 (18AWG)	150~300

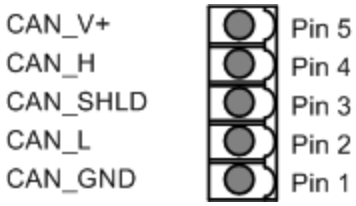
在CAN-2000系列模組內，本身便具有120Ω的終端電阻可供使用。下圖中的SW1可以用來啟動模組中的終端電阻。終端電阻啟用後，LED指示燈會亮起。

CAN bus上所能運行的鮑率受到匯流排長度的限制。下表指出不同匯流排長，所能運行的鮑率。

鮑率**(bit/s)**	最大匯流排長度**(m)**
1 M	25
800 K	50
500 K	100
250 K	250
125 K	500
50 K	1000
20 K	2500
10 K	5000

註：當匯流排的長度超過1公里時，就必需要在匯流排上，另外加裝橋接器或增益器。

CAN-2000C 上的匯流排連接器，其接腳所代表的意義，如下所示：



接腳編號	訊號	描述
1	CAN_GND	接地線(0V)
2	CAN_L	CAN_L 匯流排線(dominant low)
3	CAN_SHLD	選用的 CAN 遮蔽
4	CAN_H	CAN_H 匯流排線(dominant high)
5	CAN_V+	CAN 外部電源

2.2 電源指示燈

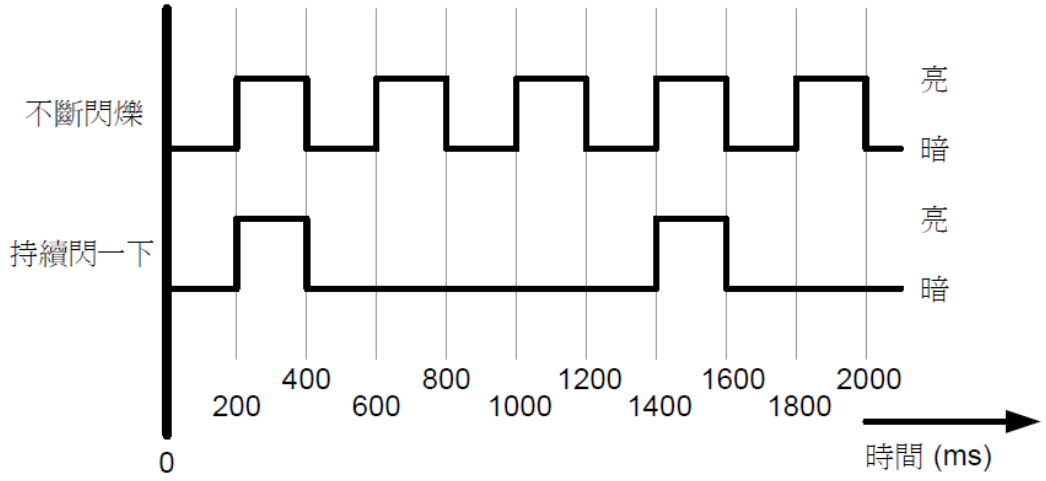
每一部CAN-2000C系列的裝置均需要10~30伏特的直流電壓作為電源輸入。若接線方式正確且供應的電力足夠，則紅色電源指示燈將會亮起。若供電後，電源指示燈無法亮起，使用者於此時可先檢查電源供應器是否正常作動，供電電壓是否正常。

2.3 CANopen 狀態指示燈

每個CAN-2000C系列模組均提供兩個CANopen 的指示燈，分別是錯誤指示燈(橘色)和運行指示燈(綠色)。錯誤指示燈和運行指示燈是在CANopen的規範內所定義的。這些指示燈會以不同的閃爍方式來對應不同的CANopen 通訊事件。於下將解釋不同的指示燈閃爍方式所代表的意義。

2.3.1 運行指示燈

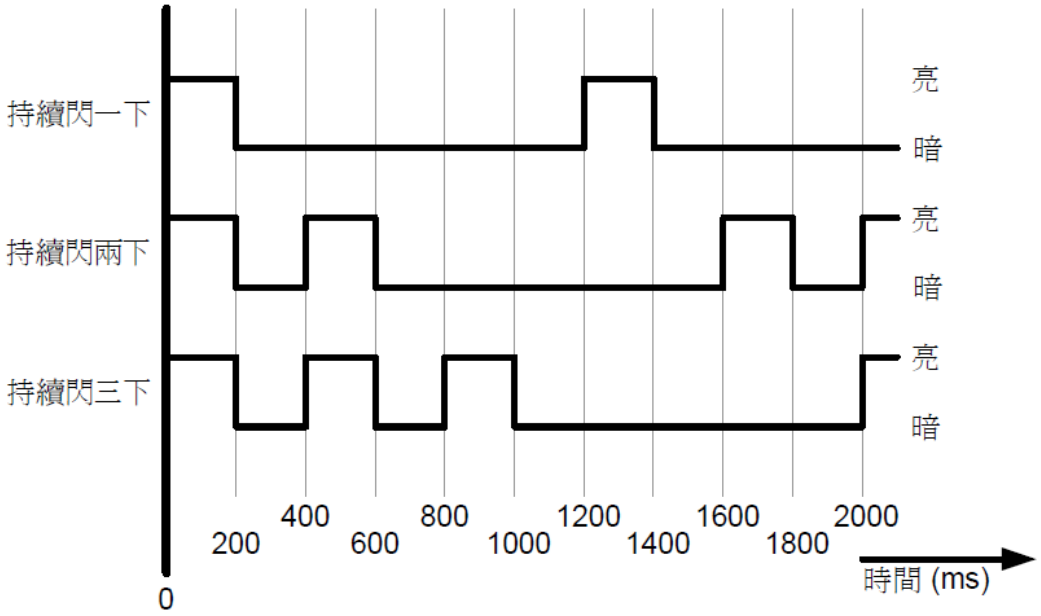
我們可以透過運行指示燈來了解裝置目前的網路狀態機制(network state mechanism)為何。關於CANopen 網路狀態機制的資訊，請參考3.5.1 節。不同的指示燈閃爍方式和其對應的意義，將分別呈現於下圖與下表：



編號	CAN 運行指示燈	狀態	描述
1	不亮	沒有運作	電源故障或接線有誤
2	持續閃一下	停止(stop)	裝置目前處於停止狀態
3	不斷閃爍	預操作(pre-operational)	裝置目前處於預操作狀態
4	恆亮	操作(operational)	裝置目前處於操作狀態

2.3.2 錯誤指示燈

以下幾種狀況可能改變錯誤指示燈的燈號，CAN 物理層(physical layer)發生錯誤，裝置沒有收到某些特定的訊息(如SYNC 訊息或NMT 守衛訊息等)所引起的錯誤。每一種錯誤事件均有其不同的閃爍頻率，不同的指示燈閃爍方式和其對應的意義，將分別呈現於下圖與下表：



編號	錯誤指示燈	狀態	描述
1	不亮	沒有錯誤	裝置目前處於工作狀態
2	持續閃一下	已到達警告限制	至少有一個CAN 控制器的錯誤計數器，已經到達或超過警告標準。(錯誤幀的數目太多)
3	持續閃兩下	錯誤控制事件	發生了守衛(guard)事件(NMT僕端或NMT 主端)
4	恆亮	匯流排關閉 (Bus Off)	CAN 控制器已經到達了匯流排關閉的條件。

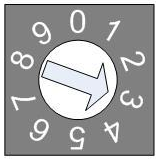
註：若同一時間內有多個錯誤出現，編號高者將會優先被顯示。舉例來說，若NMT錯誤(編號3)和Bus Off錯誤(編號4)同時發生，則Bus Off將會優先被顯示。

2.4 節點 ID 旋鈕與鮑率旋鈕

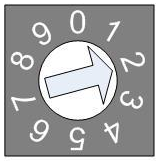
上面兩顆旋鈕是用來調整CAN-2000C模組的節點ID。其中上面的旋鈕代表節點ID十位數的部份，而下面的旋鈕則是代表節點ID個位數的部份。

以下圖為例，此CAN-2000C模組的節點ID就是32。

(x10)



(x1)



節點 ID 旋鈕

最下面的旋鈕(BAUD)是用來調整CAN-2000C模組的鮑率。鮑率旋鈕所指的數值與實際鮑率的對應關係請參照下表：

鮑率旋鈕

旋鈕的數值	鮑率 (K BPS)
0	10
1	20
2	50
3	125
4	250
5	500
6	800
7	1000

2.5 支援模組

CAN-2000C 系列模組包含了數位輸入、數位輸出、類比輸入、類比輸出等型別。詳細資訊可以參考以下網址.

網址: http://www.icpdas.com/products/Remote_IO/can_bus/can-2000.htm

3 CANopen 系統

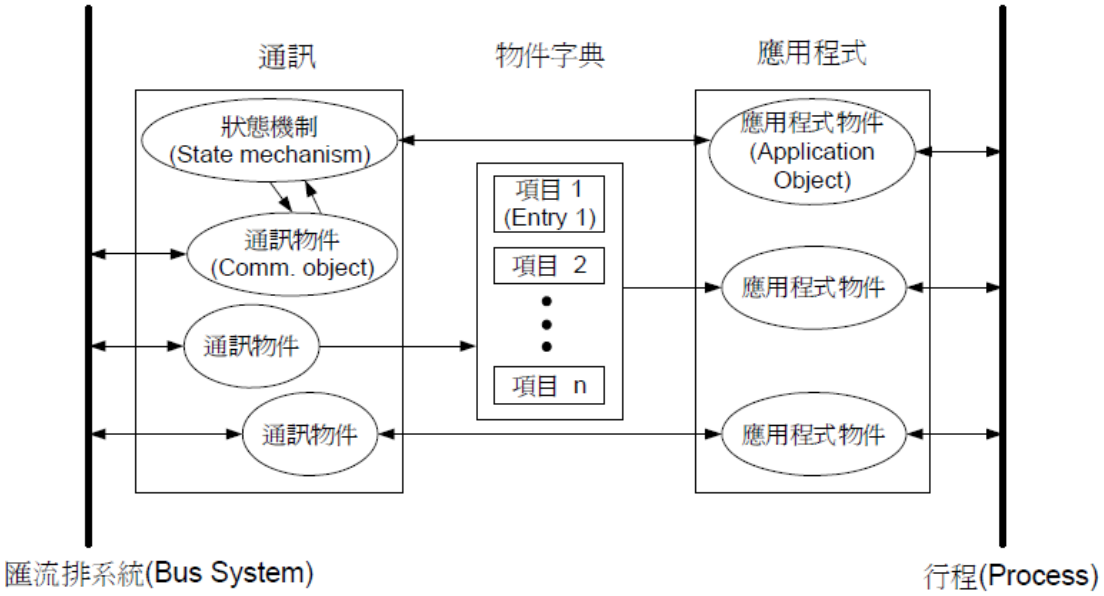
CANopen是一種基於CAN bus 之上的網路協定，且已經被使用在各種不同的應用中，像是交通工具，工業機械，自動化建築，醫療裝置，航海應用，飯店器具，實驗室設備與研究。

3.1 CANopen 介紹

CANopen內不只提供了訊息的廣播，同時也支援了節點間的點對點資料交換。CANopen內所規範的網路管理功能，可簡化了專案的設計。此外，使用者還可以透過CANopen規範內的網路啟動(network start-up)標準機制和錯誤管理(error management)標準機制，來對CANopen的網路進行實作與偵錯。

根據CANopen的裝置模型(device model)，任何CANopen的裝置均可有效率的存取I/O 的數值，或著查詢同一網路內其他裝置的節點狀態。一般來說，一個CANopen的裝置可以大略區分為三個部份。

- 通訊(Communication)
- 物件字典(Object Dictionary)
- 應用程式(Application)

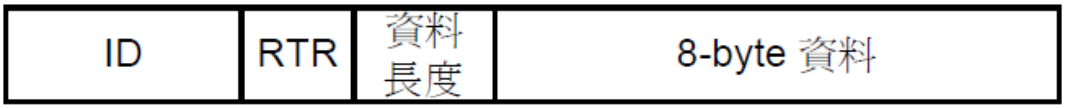


通訊

裝置模型中的通訊部份內含數個不同功能的通訊物件(COB，Communication Object)，裝置和裝置之間就是使用這些COB 來傳遞CANopen的訊息。這些COB分別為PDO(過程資料物件)、SDO(服務資料物件)、NMT(網路管理物件)以及SYNC(同步物件)等。每一通訊物件均有其不同的用途，在使用時也必須依循其通訊模型。

以PDO、SDO和NMT為例。SDO可用來存取裝置物件字典內的各個項目，其通訊模型為用戶端/伺服端(client/server)的架構(詳見3.2 節)。相較於SDO，PDO通訊物件的傳輸協定沒有header，也就是沒有額外的負擔(overhead)，速度較快，適合用來傳送與接收即時性的資料或I/O數值。PDO的通訊模型乃是依循了生產者與消費者(producer/consumer)的架構，這種架構同時也被稱為推挽式(Push/Pull)的模型(詳見3.3 節)。NMT的通訊物件則被用監控CANopen 網路中，各節點的狀態，其遵循了主從式(master/slave)的架構(詳見3.5 節)。

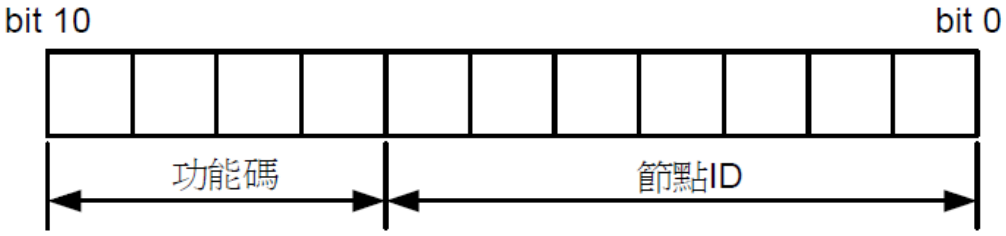
不管使用了哪一種的COB來進行訊息的傳輸，其格式都必須符合CAN 2.0 A所定義的資料幀(data frame)規範，一般來說，資料幀的格式會如同下圖：



ID 欄位共有 11 個 bit，作為匯流排上訊息優先權的仲裁機制之用。

RTR欄位則僅有一個bit，若某個訊息的RTR被設定為1，則這個訊息就是用作遠端傳送要求(RTR，Remote Transmit Requests)之用，此時，8 bytes 的資料欄位將不會被使用與傳送。資料長度欄位包含了4個bit的資料，其代表著在8 bytes的資料欄位內，有效的資料數。最後一個欄位，也就是資料欄位，則用來存放訊息資料。值得特別注意的是，根據CANopen的規範，裝置在傳輸資料時，資料欄位內每一筆獨立的數值，是以低字節先傳的方式來傳送的(Little Endian)。

各COB在透過data frame進行傳送時，所使用的Frame ID又被稱為COB-ID。根據CANopen 的規範，所有COB預設的COB-ID均是由4 bit的功能碼(Function Code)和7 bit的節點ID(Node ID)組合而成的，其格式如下所示：



COB-ID被定義用來識別訊息的來源與目的地，也被用來區別訊息的用途，另外也決定了網路中了每一點，在傳送訊息時的優先等級。根據CAN bus的仲裁機制，COB-ID數值較低的CAN訊息，有較高的優先等級可被傳送進入CAN bus。

另外，根據CANopen 的規範，有部分的COB-ID被保留給特定的通訊物件，或著留作更進一步的用途，使用者不能夠隨意的使用這些COB-ID。於下表列出被保留的COB-ID：

被保留的 **COB-ID (Hex)**	被使用的物件
0	網路管理(NMT)
1	保留
80	同步(SYNC)
81~FF	緊急事件(EMERGENCY)
100	時戳(TIME STAMP)
101~180	保留
581~5FF	預設用來傳輸 SDO
601~67F	預設用來接收 SDO
6E0	保留
701~77F	網路管理(NMT)錯誤控制
780~7FF	保留

除了之前提到被保留的COB-ID，使用者可以依據實際需求，使用剩下來的COB-ID。

(Bit10~Bit7) (功能碼)	(Bit6~Bit0)	通訊物件名稱
0000	0000000	網路管理(NMT)
0001	0000000	同步(SYNC)
0010	0000000	時戳(TIME STAMP)
0001	節點 ID	緊急事件(EMERGENCY)
0011/0101/0111/1001	節點 ID	TxPDO1/2/3/4
0100/0110/1000/1010	節點 ID	RxPDO1/2/3/4
1011	節點 ID	用來傳輸的 SDO (TxSDO)
1100	節點 ID	用來接收的 SDO (RxSDO)
1110	節點 ID	網路管理(NMT)錯誤控制

物件字典

物件字典內蒐集了許多重要的資訊。這些資訊對裝置的行為有重要的影響，像是I/O通道中的資料、通訊的參數與網路的狀態。物件字典實質上是一群物件，而一個物件內可能有一到多個項目，另外這些項目能夠透過一事先定義(pre-defined)的方法經由網路被存取。

物件字典內的每一個項目均有其各自的作用(如通訊參數、裝置描述文件(device profile)等)、資料型別(如8-bit的整數、8-bit的無號整數等)和存取類型(唯讀、唯寫等)。物件字典內的所有數值均以16進制來表示。

物件在物件字典中的位置，乃是透過一16-bit的主索引來定址，如果某個物件內有很多個項目，那這些項目還必須再透過一個8-bit的子索引來定址，如果某個物件內只有一個項目，那用主索引便可對這個項目定址。

標準物件字典(standard object dictionary)之中，所有描述文件的位置如下：

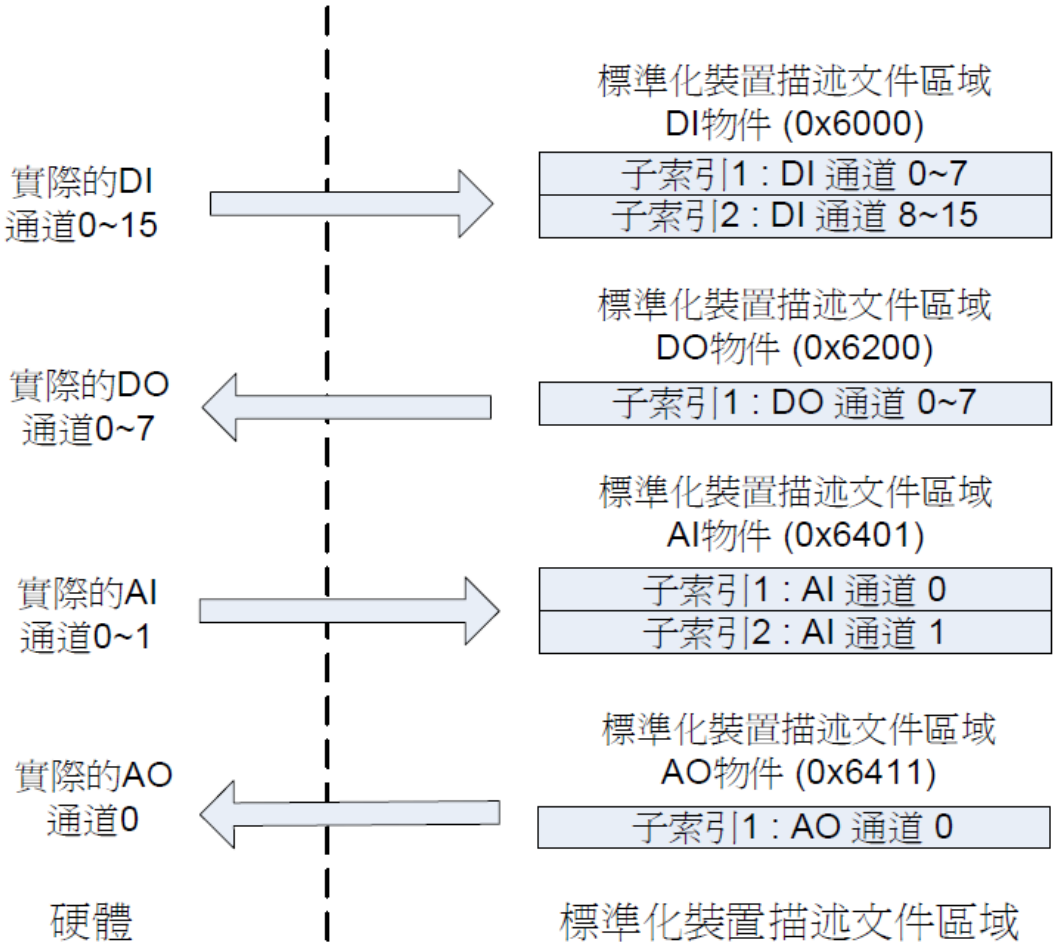
主索引(hex)	物件
0x0000	保留
0x0001 - 0x001F	靜態資料型別(Static Data Types)
0x0020 - 0x003F	複合資料型別(Complex Data Types)
0x0040 - 0x005F	製造商特定的複合資料型別(Manufacturer Specific Complex Data Types)
0x0060 - 0x007F	裝置描述文件特定的靜態資料型別
0x0080 - 0x009F	裝置描述文件特定的複合資料型別
0x00A0 - 0x0FFF	保留作更進一步的用途
0x1000 - 0x1FFF	通訊描述文件(Communication Profile)區域
0x2000 - 0x5FFF	製造商特定的描述文件區域 (Manufacturer Specific Profile Area)
0x6000 - 0x9FFF	標準化裝置描述文件區域 (Standardized Device Profile Area)
0xA000 - 0xBFFF	標準化介面描述文件區域 (Standardized Interface Profile Area)
0xC000 - 0xFFFF	保留作更進一步的用途

以標準化裝置描述文件區域為例，假設一個CANopen 的裝置有16個DI、八個DO、兩個AI以及一個AO 的通道(channel)，這些通道的數值均會分別被對應

到標準化裝置描述文件區域內的數個項目，如主索引0x6000、0x6200、0x6401和0x6411物件內的項目。

舉例來說，當CANopen的裝置抓取輸入通道的數值後，就會把這些數值存到主索引為0x6000和0x6401物件內的項目。此外，裝置可分別輸出存在主索引為0x6200和0x6411的物件內項目的數值到DO和AO通道上。

大致上的概念如下圖所示：

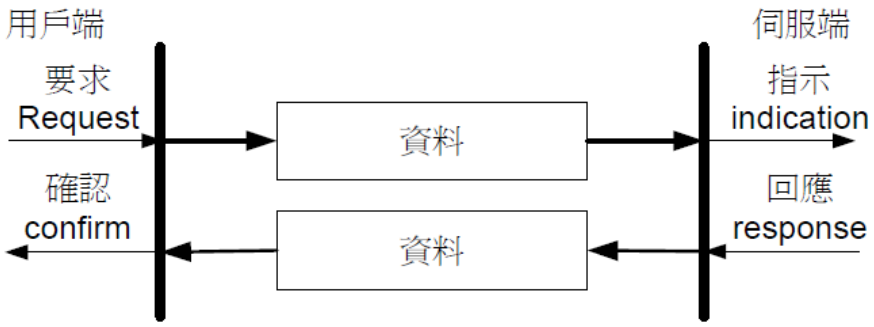


應用程式

應用程式物件包含了裝置上，裝置與行程環境(process enviroment)之間互動的所有功能，其可被視為是物件字典和實際行程(如DI/DO/AI/AO)之間的橋樑。

3.2 SDO 介紹

SDO(Service Data Object，服務資料物件)的作用為存取裝置物件字典的項目。藉由SDO的通訊方式，來建立兩個裝置之間點對點通訊的橋樑。SDO的通訊模型依循著用戶端-伺服端(Client-Server)的關係，如下圖所示：



SDO可分為兩種，分別為RxSDO(Receive SDO)以及TxSDO(Transmit SDO)，它們的COB-ID是分開的。這兩種SDO的區分方式，是從CANopen 裝置的角度來看的。

舉例來說，若使用者欲傳送一SDO訊息給CAN-2000C模組，則此SDO對裝置而言就是RxSDO。因此，這個SDO訊息的COB-ID就必須是裝置的RxSDO COB-ID。若CAN-2000C模組希望對外傳送一個SDO訊息，則此SDO對裝置而言就是TxSDO。這個訊息的COB-ID就必須是裝置的TxSDO COB-ID。

此外，所有SDO的傳輸，均需由SDO用戶端主動發起。而在SDO用戶端開始進行SDO傳輸之前，需先針對其需求選擇適當的SDO傳輸協定。

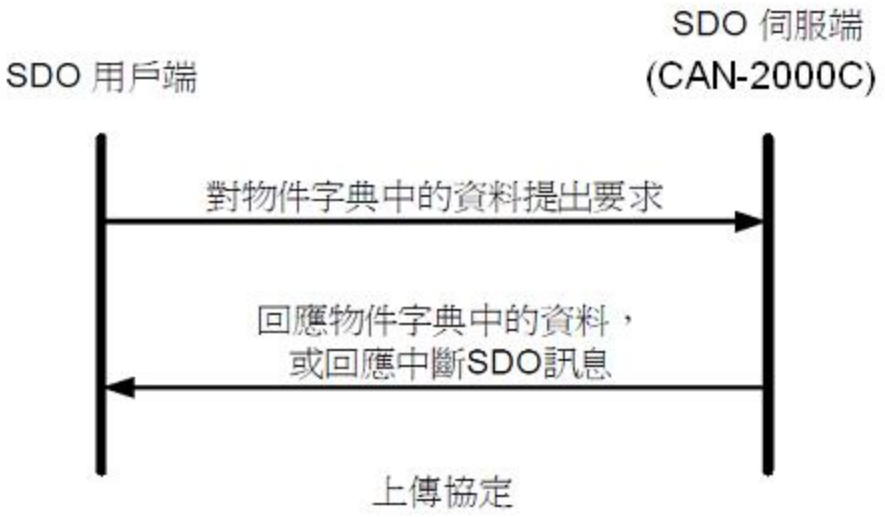
如果SDO用戶端必須要由SDO伺服端上的物件字典中獲取資訊，則需要使用上傳SDO協定(upload SDO protocol)或SDO區塊上傳協定(SDO block upload protocol)。前者適合用來傳輸較少量的資料，後者則適合用來傳輸較大量的資料。

當SDO用戶端想要更改SDO伺服端的物件字典內容時，則可以透過下載SDO協定(download SDO protocol)或SDO區塊下載協定(SDO block download protocol。這兩者之間的差異就如同於上傳SDO 協定與SDO 區塊上傳協定之間的差異。

此外，因為物件字典中，不同的項目會有不同的存取類型，並不是所有物件字典的內容都能透過SDO的傳輸來進行存取。如果SDO用戶端試圖更動SDO伺服端物件字典中，不被允許存取的項目，這時SDO伺服端就會啟動異常中止SDO

傳輸協定(Abort SDO Transfer Protocol)，以終止SDO的傳輸。

CAN-2000C模組僅支援作為SDO伺服端。因此，它只能被動等待SDO用戶端發起SDO的傳輸。有關CAN-8123/CAN-8223/CAN-2000C模組的上傳協定和下載協定，其大致上的概念可參考下圖：



3.3 PDO 介紹

PDO(Process Data Object)常被用來傳輸即時性的資料，像是DI、DO、AI、AO等等。由於CAN bus的傳輸資料格式之限制，因此，透過PDO來傳輸資料，每一個PDO內最多僅能放入8 bytes的資料。另外，因為PDO的訊息在傳輸時沒有傳輸協定上的額外負擔(overhead)，在資料的傳輸上比起其它CANopen的通訊物件都更有效率。

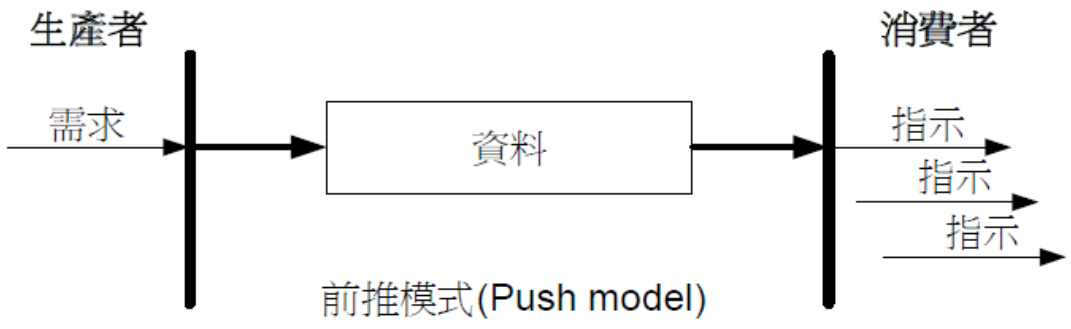
PDO 的通訊模式

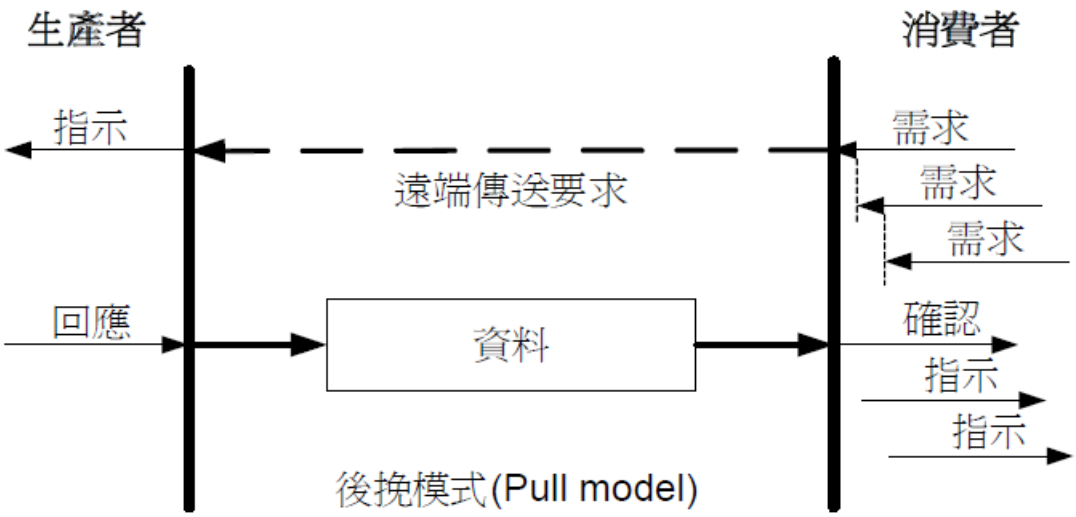
PDO的傳送與接收通訊模型，乃是依據生產者/消費者(producer/consumer)的架構所實作的，這種通訊模式也可稱為推挽式(push/pull)模型。

當PDO前推模式(push mode)開始進行時，必需有一個CANopen的裝置扮演PDO生產者的角色，而PDO消費者則可有可無。在PDO生產者送出PDO訊息之前，這一個PDO訊息必需要先在CAN bus上的仲裁中獲勝。接著，CAN bus上的每一個PDO的消費者都會收到這個PDO訊息，並決定接受或著丟棄這個PDO訊息。

在PDO後挽模式(pull mode)中，會由一個PDO消費者先傳送一個遠端傳送要求(remote transmit request, RTR)給PDO生產者，然後PDO生產者根據這個遠端傳送要求的訊息，它會回覆一個具有相同COB-ID的PDO訊息給在CAN bus上的每一個PDO消費者。

PDO 的通訊架構圖如下所示：





PDO和SDO一樣，也可分為兩種，分別為RxPDO以及TxPDO，它們的COB-ID一樣是分開的。這兩種PDO的區分方式，也和SDO一樣，同樣是從CANopen裝置的角度來出發的。

裝置用來對外傳輸資料的PDO就是TxPDO，其常被用來傳輸裝置上DI/AI通道的數值。而在裝置用來接收資料的PDO，就是RxPDO，其常被拿來改變裝置上DO/AO通道的數值。

以CAN-2000C模組為例，如果一個PDO生產者傳送了一個PDO訊息給CAN-2000C模組，這個訊息的COB-ID必需為裝置的RxPDO COB-ID，這是因為從CAN-2000C模組的角度來看，這動作是屬於PDO的接收。相對地，當PDO的消費者傳送遠端傳送要求(RTR)訊息給CAN-2000C模組，要求其對外傳送TxPDO時，此一訊息的COB-ID就必需使用CAN-2000C模組的TxPDO COB-ID，這是因為對於CAN-2000C模組來說，回覆遠端傳送要求的動作是屬於PDO的傳送。

PDO 的觸發模式

對PDO生產者而言，PDO訊息傳輸的觸發條件可以分為三種。分別為事件驅動(event driven)、時間驅動(timer driven)以及遠端要求(remote request)條件。以下將對這三種條件做描述。

事件驅動 (Event Driven)

PDO 的收送可以被一物件特定事件(object specific event)的發生所觸發。

以循環同步傳輸型態的PDO為例，當裝置接收到某一特定數量的SYNC訊息時，便會觸發PDO的收送。

對非同步傳輸型態的PDO而言，裝置特定事件是由CANopen的規範，DS-401 v2.1所定義的。根據規範，當被映射到DI通道的TxPDO，其傳輸型態被設定為非同步時，則只要DI通道數值發生變化，便會觸發裝置對外傳送此TxPDO。

時間驅動 (Timer Driven)

PDO的傳輸除了可以被裝置特定事件的發生所觸發之外，也可以設定成，若沒有事件發生的狀態持續一段特定時間後，便讓裝置對外傳送PDO。

舉例來說，使用者可以設定TxPDO通訊參數內的事件計時器，則當經過了事件計時器內所記錄的特定時間間隔，且在這段時間內均未發生其他的事件，CAN-2000C模組就會對外傳送此TxPDO。

遠端要求 (Remote Request)

如果PDO的傳輸型態被設定唯遠端傳送要求、非同步或非循環同步，則只有在接收到其他PDO消費者所傳輸的遠端傳送要求(RTR)之時，PDO傳輸才會被觸發。

PDO 傳輸型態(Transmission Types)

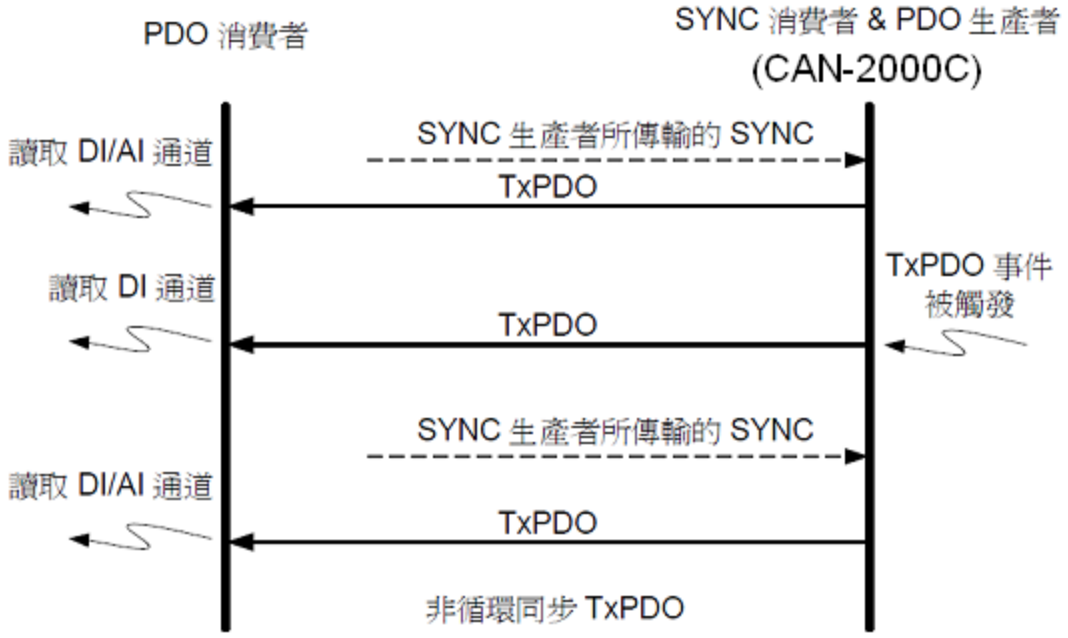
一般來說，PDO的傳輸型態可大略分為兩種，分別為同步和非同步。在同步傳輸型態下，必須要藉由SYNC訊息的接收，才會觸發PDO的收送。在非同步型態下，PDO傳輸的觸發是與SYNC物件相互獨立的。

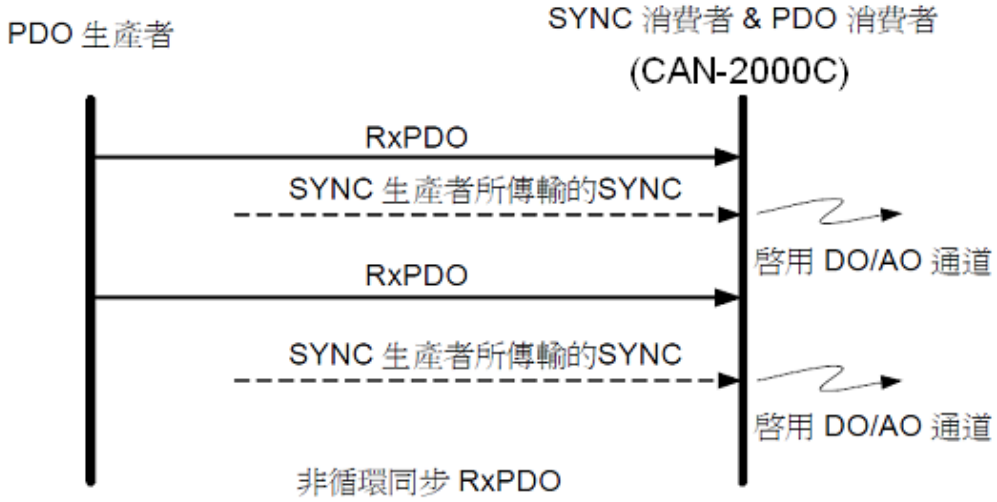
同步傳輸型態可再細分為三種，分別為非循環同步、循環同步以及唯遠端傳送要求同步。而非同步傳輸型態可再細分為兩種，分別為唯遠端傳送要求非同步和非同步。

非循環同步 (Acyclic synchronous)

若某TxPDO的傳輸型態被設定為非循環同步，則裝置上若先發生物件特定事件，接著又收到由SYNC生產者傳送的SYNC物件，此時裝置就會對外傳送此TxPDO訊息給PDO的消費者。

或著裝置收到此TxPDO的RTR訊息，此時裝置也會對外傳送此TxPDO。若某RxPDO的傳輸型態被設定為非循環同步，則裝置必需要在接收到SYNC物件之後，才會啟用在接收到此SYNC物件之前所收到且尚未被啟用的RxPDO物件。每一次收到SYNC物件，裝置都會對尚未被啟用的RxPDO物件進行啟用。關於非循環同步的PDO傳輸型態，使用者可參考下圖：



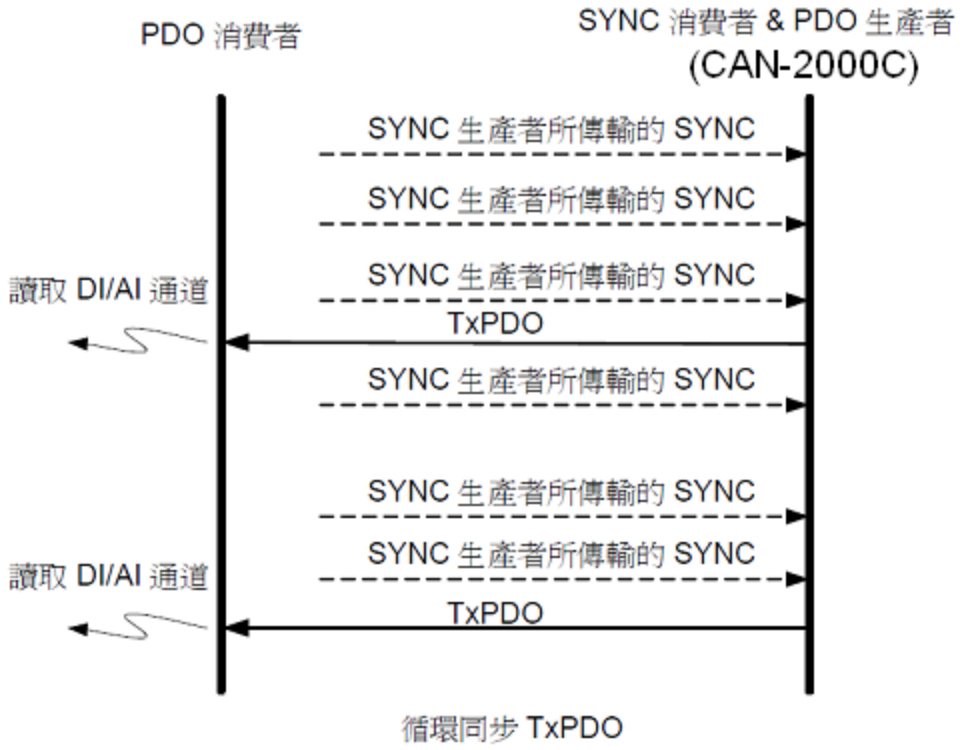


循環同步 (Cyclic synchronous)

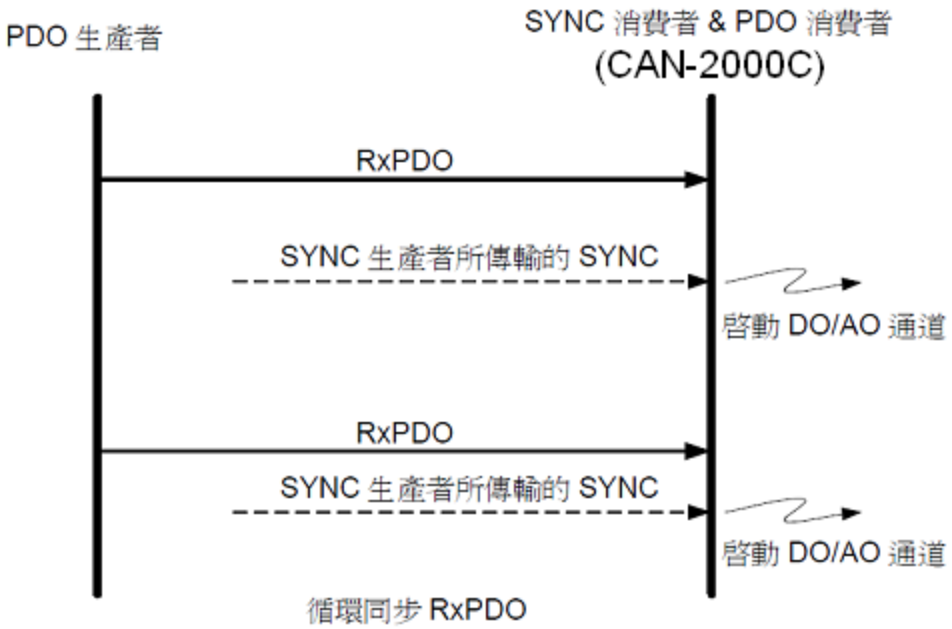
若某TxPDO的傳輸型態被設定為循環同步，則裝置必須要接收到特定數量的SYNC物件之後，才會觸發此TxPDO的傳輸，SYNC物件的特定數量最大可設定為240。

舉例來說，若某TxPDO 被設定為當接收到3個SYNC 物件後便進行觸發，則CAN-8123/CAN-8223/CAN-2000C模組將會在收到3個SYNC 物件後，進行此TxPDO的傳送。或著裝置收到此TxPDO的RTR訊息。此時裝置也會對外傳送此TxPDO訊息給PDO的消費者。

非循環同步與循環同步傳輸型態的RxPDO觸發方式相同，均是藉由接收SYNC物件來啟用RxPDO，與收到的SYNC物件數目沒有關係。關於循環同步的PDO傳輸型態，使用者可參考下圖：



循環同步 TxPDO


PDO 生產者
SYNC 消費者 & PDO 消費者
(CAN-2000C)
RxPDO
SYNC 生產者所傳輸的 SYNC
啓動 DO/AO 通道
RxPDO
SYNC 生產者所傳輸的 SYNC
啓動 DO/AO 通道
循環同步 RxPDO

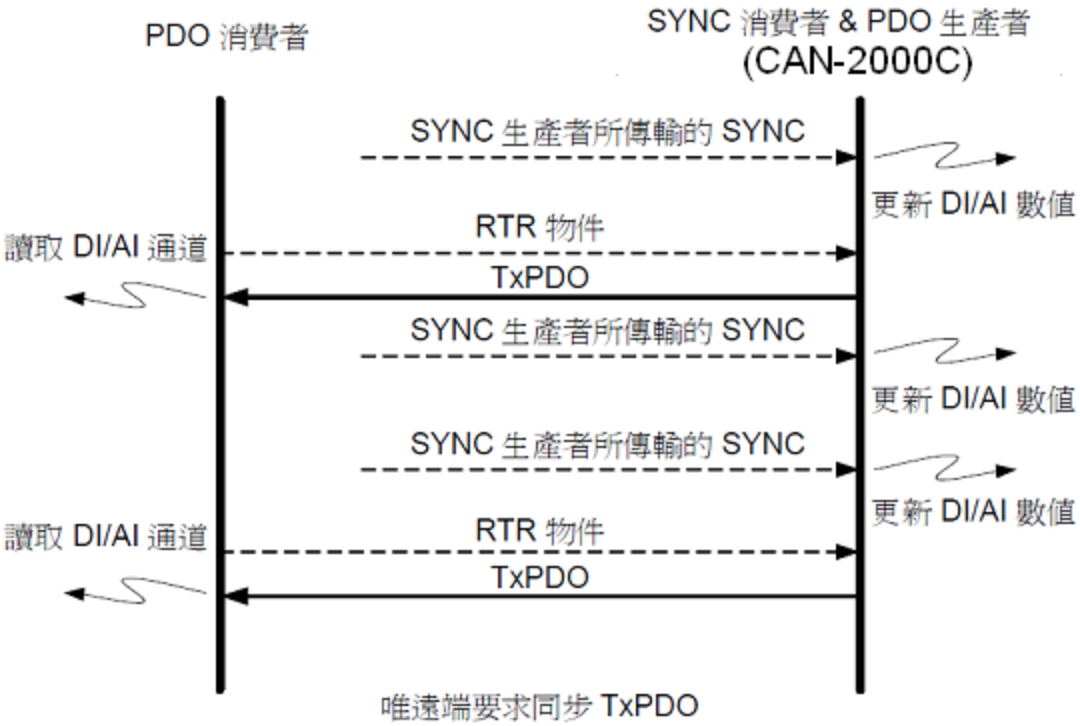
唯遠端傳送要求同步 (RTR-only synchronous)

若某TxPDO的傳輸型態被設定為唯遠端傳送要求同步，和其他傳輸型態不一樣的是，當裝置接收到RTR之後，裝置並不會去更新TxPDO裡面的數值，會直接就對外傳送此TxPDO。此時TxPDO內記錄的有可能是過時的資料。

若使用者想要獲得有最新數值的TxPDO，需先傳送SYNC訊息給裝置，讓裝置更新TxPDO內記錄的數值，再傳送此TxPDO的RTR訊息給裝置，則時，裝置對外傳送的TxPDO就會有最新的數值。

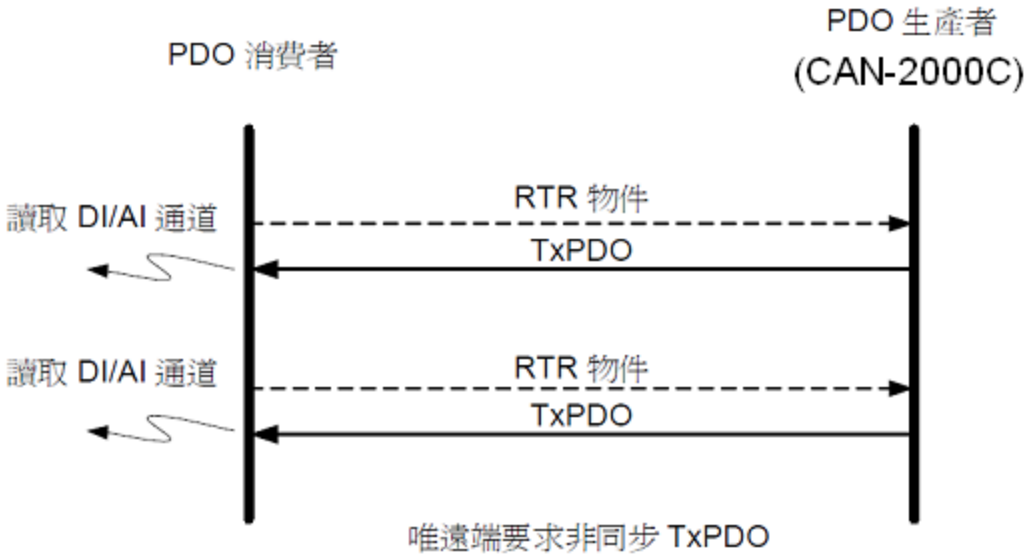
舉例來說，若這個TxPDO內的內容映射到DI/AI，則只有在裝置接收到SYNC物件後，裝置才會抓取最新的DI/AI數值到TxPDO之中。此時再傳送此TxPDO的RTR訊息給裝置，則裝置就會對外傳送最新數值的TxPDO。

只有TxPDO能被設定成這種傳輸型態。關於唯遠端傳送要求同步的PDO傳輸型態，使用者可參考下圖：



唯遠端傳送要求非同步 (RTR-only asynchronous)

只有TxPDO的傳輸型態能被設定為唯遠端傳送要求非同步，對此傳輸型態的TxPDO來說，只有在裝置接收到此TxPDO的RTR訊息之後，裝置才會對外傳輸此TxPDO。使用者可參考下圖：

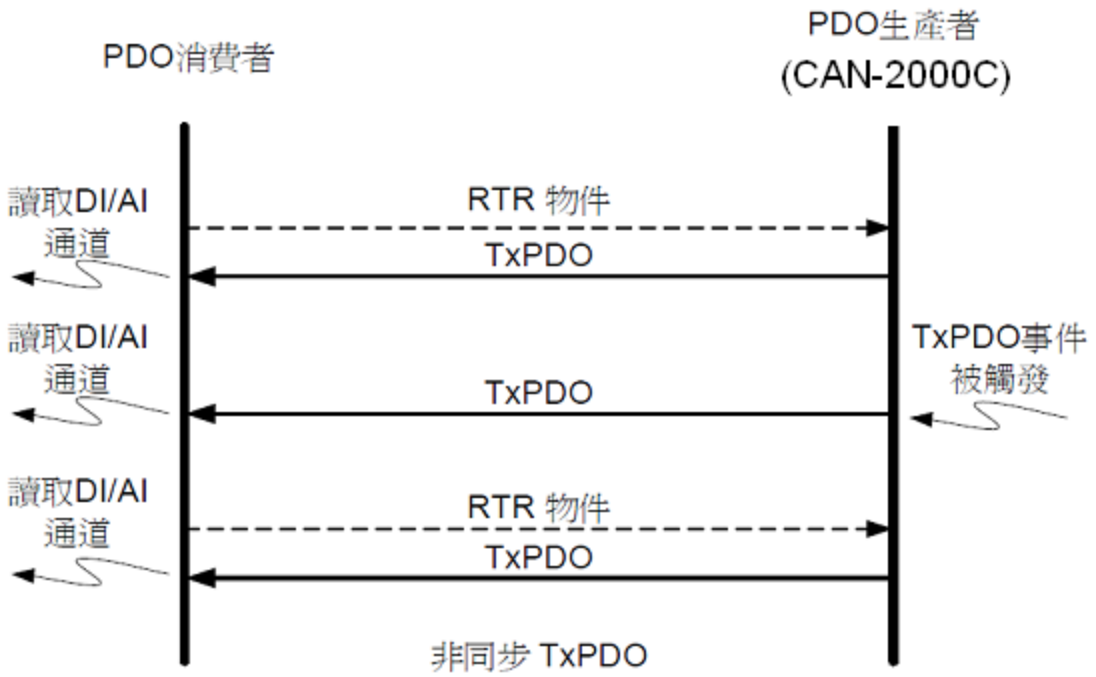


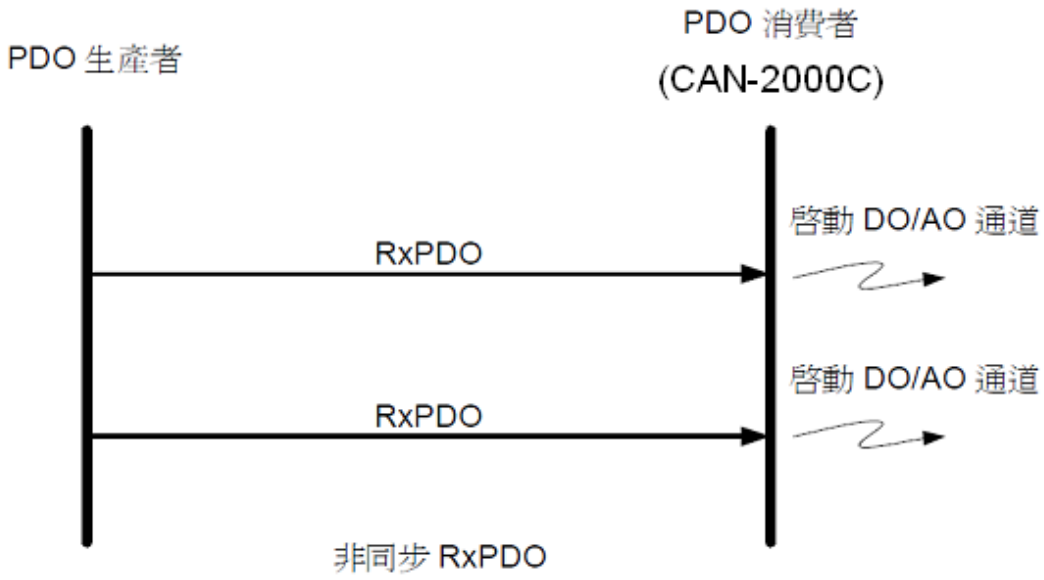
非同步 (Asynchronous)

若某TxPDO的傳輸型態被設定為非同步，則TxPDO訊息的觸發可以透過以下兩種方式來達成，一為裝置收到此TxPDO的RTR訊息，一為先前提到過，裝置上特定事件的發生。

另外，若RxPDO的傳輸型態被設定為非同步，則當裝置接收到RxPDO時，裝置便會直接啟用RxPDO，不需要接收SYNC訊息。

而CAN-2000C模組在開機之後，不管是TxPDO還是RxPDO，其預設的傳輸型態就是非同步。有關非同步傳輸型態使用者可參考下圖：




PDO 生產者
PDO 消費者
(CAN-2000C)
RxPDO
啓動 DO/AO 通道
RxPDO
啓動 DO/AO 通道
非同步 RxPDO

抑制時間 (Inhibit Time)

由於CAN bus的仲裁機制，在CANopen中，COB-ID較小者，有著較高的傳輸優先權。

舉例來說，如果CAN bus上面有兩個節點，其中一個想要傳送COB-ID為0x181的訊息，另外一個想要傳送COB-ID為0x182的訊息。當這兩個節點同時傳送訊息到CAN bus上時，只有COB-ID為0x181的訊息能夠被成功傳送，這是因為它有著較高的傳輸優先權。COB-ID為0x182的訊息必須等到匯流排再次空閒(bus idle)，才能被傳送。這樣的仲裁機制可以擔保在訊息傳輸過程當中有衝突發生時，至少能夠有一個節點成功的傳輸訊息。

然而，如果COB-ID為0x181的訊息不斷的被傳送，那COB-ID為0x182的訊息將永遠沒有機會被傳送。換言之，如果較高優先權的訊息連續的被傳送，那較低優先權的CAN訊息就不可能會傳輸成功，這就是這種仲裁機制的缺點。

為了避免這種狀況的發生，CANopen的規範就對每一個PDO的物件定義了所謂的抑制時間(單位為100us)，當同一個PDO的物件在連續傳送的狀態下，PDO訊息與PDO訊息之間的時間間隔，必須大於此PDO的抑制時間，如此就可避免上述情況的發生。

事件計時器 (Event Timer)

此參數只對TxPDO有作用。若某TxPDO的事件計時器不為0，且其為非同步傳輸模式，則計時器便會開始倒數，當計時器倒數到0時，就會被視為發生一次事件，並且此事件會觸發TxPDO的傳輸。事件計時器所記錄的數值，其時間單位為1ms。

PDO 映射參數 (PDO Mapping Objects)

CANopen裝置上的PDO映射參數，提供了PDO訊息與實際I/O資料之間的連結。而PDO訊息中每個byte所代表的意義(即PDO映射參數的內容)，又可以透過SDO訊息來加以改變。所有的PDO映射參數皆存放在通訊描述文件區域之中(Communication Profile Area，物件字典中的1000-1FFF)。根據CANopen的規範CiA DS-401，RxPDO和TxPDO預設的映射參數具有下面的特性：

- 預設的 PDO 映射中應分別支援 4 組的 TxPDO 和 4 組的 RxPDO。

- 第 1 組 RxPDO 和 TxPDO 的映射，分別被對應到 DO 和 DI。

- 第 2、3、4 組的 RxPDO 映射被對應到 AO，第 2、3、4 組的 TxPDO 被對應到 AI。

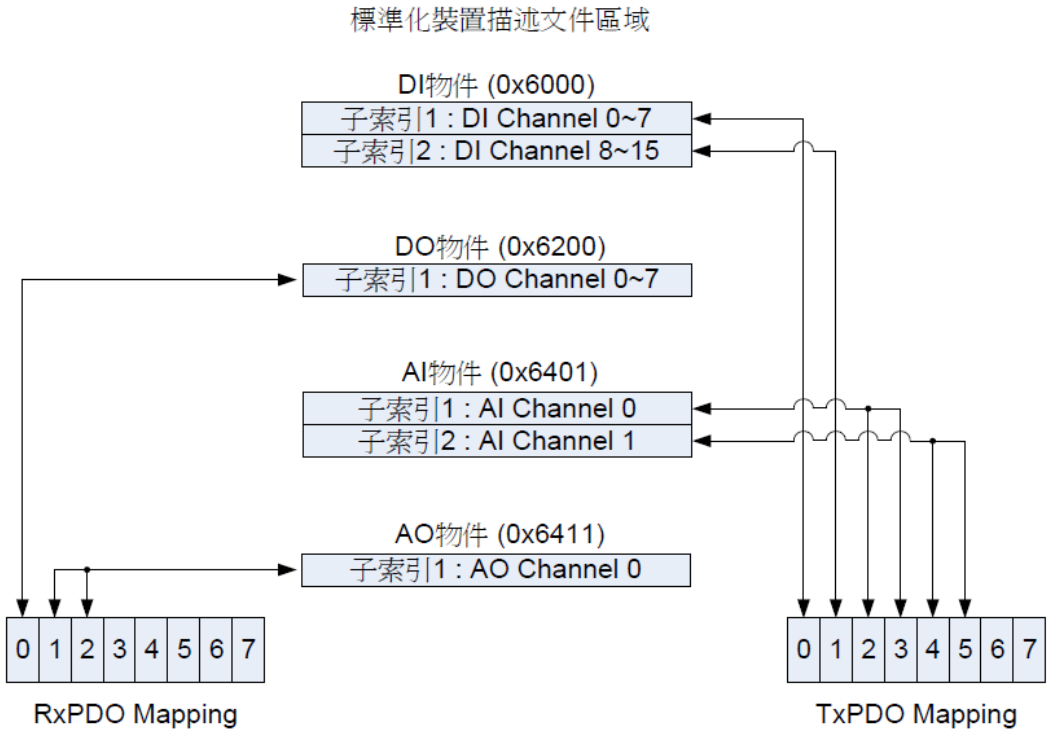
- 在預設的狀況下，前8個DO通道的物件會被映射到第1個RxPDO，而前8個DI的物件會被映射到第1個TxPDO(一個DO/DI物件的大小為1 byte，其內至多可含8個DO/DI的通道，而一個RxPDO/TxPDO內的資料欄位最多可容納8 bytes，也就是8個DO/DI的物件)。若一個裝置所支援的DO/DI物件超過8個，則裝置就會從第5個RxPDO/TxPDO開始映射第8個以後的DO/DI物件，因為第2個到第4個RxPDO/TxPDO預設是保留給AO/AI物件使用的。而在預設的情況下，前12個AO的物件會依序被映射到第2、3、4個RxPDO，而前12個AI的物件會依序被映射到第2、3、4個TxPDO (一個AO/AI物件的大小為2 byte，內含1個AO/AI的通道，而一個RxPDO/TxPDO內的資料欄位最多可容納8 bytes，也就是4個AO/AI的物件)。若一個裝置所支援的AO/AI 物件數量超過12個，則必須要從第1個未被映射的RxPDO/TxPDO開始映射第12個以後的AO/AI物件。

於下以例子來進一步說明，若在CAN-8123/CAN-8223/CAN-2000C模組 的物件字典之中，有11個DO和13個AO的物件。則在預設的情況下，前8個DO的物件會被映射到第1個RxPDO，後3個DO的物件會被映射到第5個RxPDO。前4個AO的物件會被映射到第2個RxPDO，接下來的8個AO的物件將會被分別映射到第3個RxPDO和第4個RxPDO，另外因為第5個RxPDO已經用來映射DO了，故最後一個AO的物件會被映射到第6個RxPDO。

在使用PDO進行通訊之前，針對此PDO，生產者和消費者必須要有相同的PDO映射參數。一方面，PDO生產者需要PDO的映射參數用來決定要傳送的PDO該填入哪些資料。

另一方面，PDO消費者需要PDO的映射參數才知道接收到的PDO訊息之中，每一個byte所代表的實際意義。除了預設的PDO映射之外，使用者也可以根據實際使用的需求，自行定義PDO映射參數。

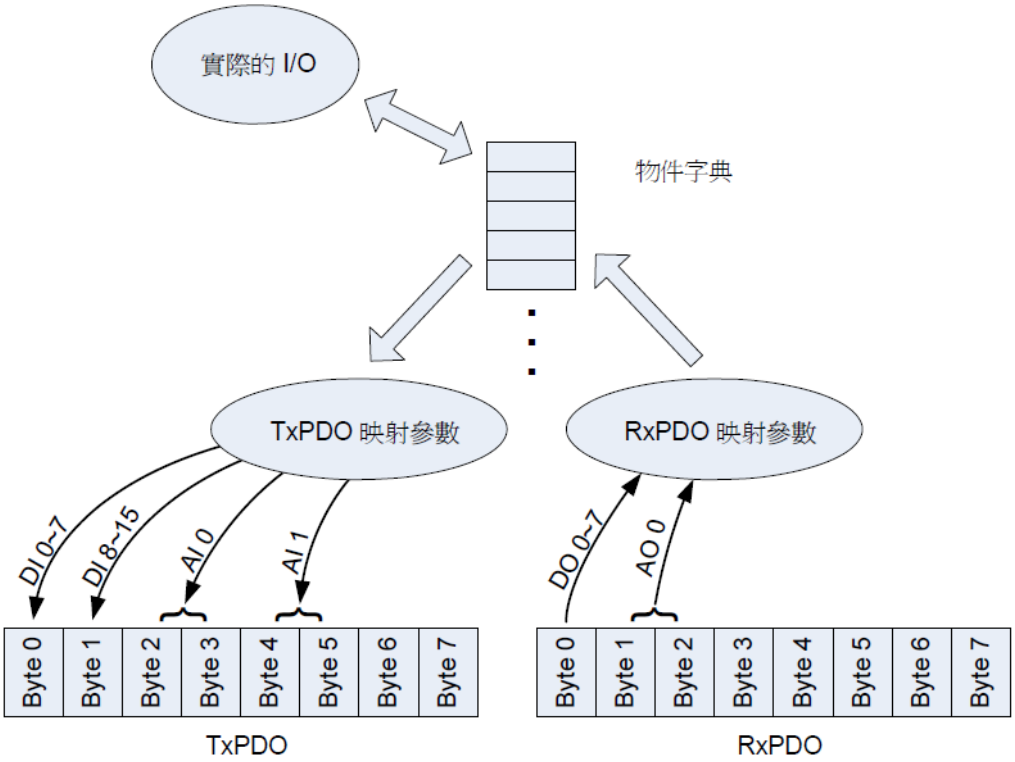
若一個CANopen的裝置有16個DI、8個DO、2個AI和1個AO通道，且這些輸入和輸出通道的數值已經分別被存入物件字典內的數個不同項目。除了預設的PDO映射之外，使用者也可以自行定義PDO映射參數，將這些通道的數值映射給使用者自定義的PDO。於此處我們把這些通道的數值分別映射到1組RxPDO和1組TxPDO上，如下圖所示：



若某CANopen的裝置收到此RxPDO訊息，根據上圖所描述的RxPDO映射關係，第1個byte會被解譯成DO通道0~7 的輸出值，其他兩個byte會被解譯成AO通道0的輸出值。在解譯完RXPDO訊息的資料後，裝置會輸出解譯後的資料到DO和AO的通道上。

而TxPDO訊息的傳送和RxPDO訊息的接收剛好相反，當CANopen裝置上發生觸發傳送此TxPDO的事件之後，裝置會依照上圖的TxPDO映射關係，把DI和AI通道的數值填到TxPDO之中，然後裝置再傳送TxPDO訊息給PDO的消費者。於此處，裝置會把DI通道0~7 和DI通道8~15 的數值填到TxPDO的前2 byte，把AO通道0的數值填到第3~4 byte，把AO通道1的數值填到第5~6 byte。

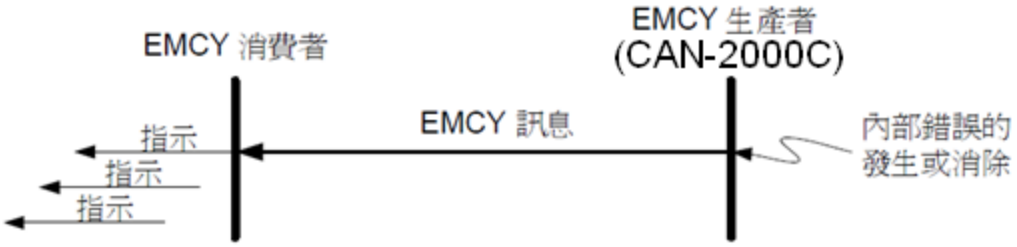
物件字典、PDO 映射參數和 PDO 訊息三者之間的關係，如下所示：



3.4 EMCY 介紹

若裝置的錯誤狀態發生改變(發生內部錯誤，或錯誤被排除)，此時裝置就會對外傳送EMCY訊息。EMCY的通訊模型為生產者/消費者架構，在CANopen的裝置偵測到錯誤狀態發生改變之後，就會被傳送EMCY訊息給EMCY消費者(一錯誤事件，對應一EMCY 訊息)。如果裝置的內部錯誤狀態沒有改變，那就裝置就不會傳送EMCY訊息。同一個EMCY訊息可多個EMCY消費者所接收。

CAN-2000C模組僅支援作為EMCY訊息的生產者。有關EMCY訊息的傳輸行為，使用者可參考下圖：



一個EMCY訊息包含8 bytes的資料，其被稱之為緊急物件資料(emergency object data)，其結構如下表所示。

Byte	0	1	2	3	4	5	6	7
內容	Emergency 錯誤碼 (Emergency Error Code)		Error 暫存器 (Error register)	製造商特定的錯誤區域 (Manufacturer specific Error Field)				

我們將於 5.3 節對緊急物件資料中的各個欄位詳細進行說明。

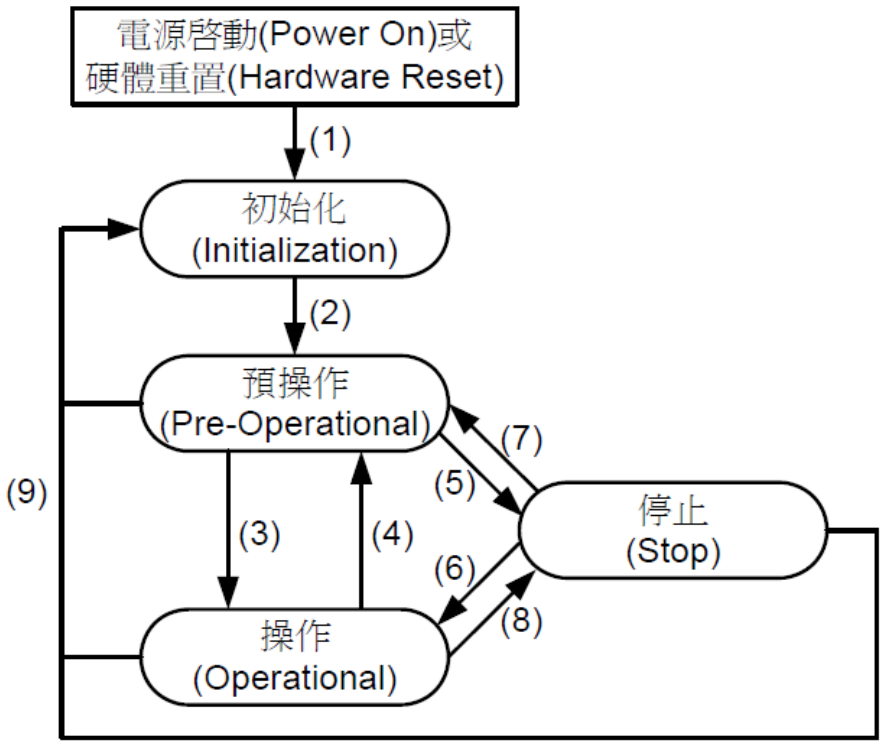
3.5 NMT 介紹

在CANopen內，所謂的NMT通訊物件，其通訊模型乃依循著節點導向(node-oriented)以及主從式(master-slave)的架構。在一個CAN bus的網路中，只有一個CANopen的裝置能夠作為NMT主端，其他的節點則作為NMT僕端。每一個NMT僕端都是獨一無二的，可藉由其節點ID(1~127)來對其進行識別。

NMT因應不同的用途支援了兩種協定，一為模組控制協定(module control protocol)，一為錯誤控制協定(error control protocol)。透過NMT模組控制協定，可以控制節點進入不同的狀態，像是起始(installing)、預操作(pre-operational)、操作(operational)、以及停止(stopped)等狀態。而錯誤控制協定則讓使用者能夠對網路中的遠端錯誤(remote error)進行偵測，可以用來確認節點的是否存活。

3.5.1 模組控制協定

在介紹模組控制協定(Module Control Protocol)之前，讓我們先將焦點放在NMT狀態機制(NMT state mechanism)的架構上。下圖列出了各NMT狀(NMT state)之間的關係以及各NMT從端(NMT slave)其NMT狀態的轉換機制。



狀態機制(State Mechanism)圖

(1)	電源啟動或硬體重置後，裝置即自動進入初始化
(2)	初始化結束後即自動進入預操作狀態
(3),(6)	“啟動遠端節點(Start Remote node)”指示(indication)
(4),(7)	“進入預操作狀態”指示
(5),(8)	“停止遠端節點(Stop Remote node)”指示
(9)	“節點重置”或“通訊重置”(Reset Communication)指示

當初始化結束之後，裝置即進入預操作狀態。此時，裝置可以根據收到的模組控制協定訊息，切換到不同的NMT狀態。在不同的NMT狀態下，裝置可使用的通訊物件會有所差異。舉例來說，裝置僅能在操作狀態下進行PDO訊息的傳送和接收。在下表中將列出在不同NMT狀態下，CANopen裝置可以使用的通訊物件。

	起始 (Installing)	預操作 (Pre-operational)	操作 (Operational)	停止 (Stopped)
PDO			**O**	
SDO		**O**	**O**	
SYNC Object		**O**	**O**	
Time Stamp Object		**O**	**O**	
EMCY Object		**O**	**O**	
Boot-Up Object	**O**			
NMT		**O**	**O**	**O**

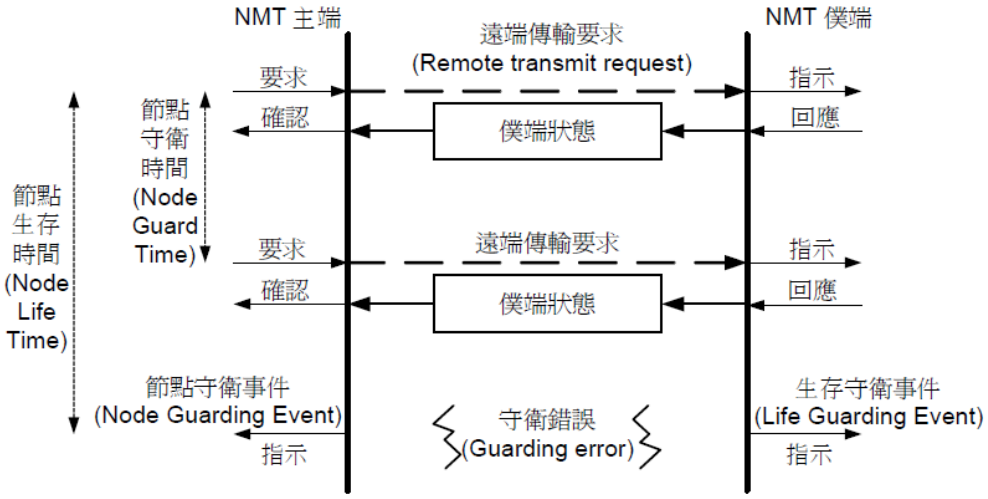
3.5.2 錯誤控制協定

CAN-8123/CAN-8223/CAN-2000C 模組所支援的錯誤控制協定(Error Control Protocols)共有兩種，分別為節點守衛協定(Node Guarding Protocol)與心跳產生者協定(Heartbeat Producer Protocol)。根據CANopen的規範，CANopen的裝置在同一時間內僅能使用一種錯誤控制協定，同時使用兩種錯誤控制協定是不被允許的，使用者可以在實際的應用中，於CAN-2000C模組上啟用這項功能。

下面將針對此兩種錯誤控制協定做進一步介紹。

節點守衛協定(Node Guarding Protocol)

節點守衛協定的通訊模型依循著主僕(Master/Slave)的關係，使用者可以透過這個協定監測網路中CANopen節點的狀態。其通訊協定如下圖所示：



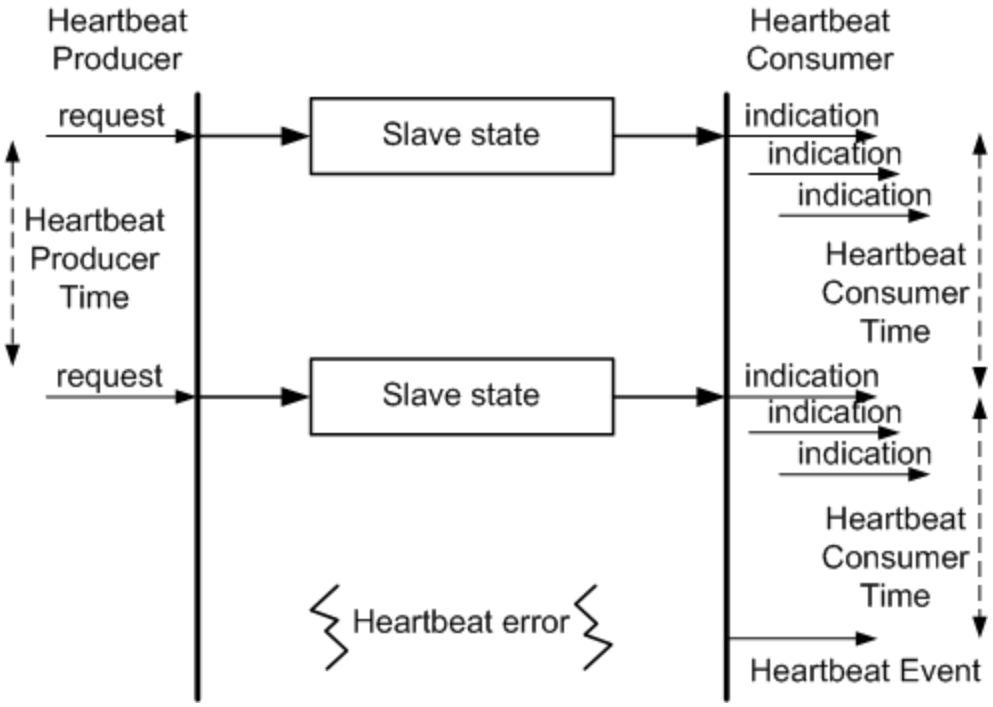
每隔一段固定的時間間隔，NMT的主端就會去對每一個NMT僕端作輪詢(poll)的動作。此一時間間隔即所謂的節點守衛時間(Node Guard Time)，使用者可以對不同的NMT僕端，設定不同的節點守衛時間。

NMT僕端收到主端的詢問之後，會回答NMT僕端目前所處的狀態，像是停止、操作或著預操作等狀態。節點生存時間(Node Life Time)為節點守衛時間乘上生存時間係數(life time factor)。不同的NMT僕端，其生存時間係數同樣可以作不同的設定。如果NMT僕端在其節點生存時間內沒有收到NMT主端的詢問，那NMT僕端就會發生生存守衛事件(Life Guarding Event)，這也就代表著遠端節點錯誤(remote node error)的產生。

NMT僕端裝置的物件字典內，主索引為0x6206和0x6443的物件項目可用來控制DO和AO通道的錯誤模式(error mode)。當發生生存守衛事件時，若0x6206和0x6443的物件項目設定錯誤模式為啟動(enable)，則NMT僕端裝置的DO和AO通道就會輸出記錄在主索引x6207和0x6444物件項目中的錯誤模式值(error mode value)，若0x6206和0x6443的物件項目設定錯誤模式為抑制(disable)，則NMT僕端裝置的DO和AO通道就會保留原先的輸出值。更多有關0x6206、0x6207、0x6443和0x6444物件的資訊，請參照第6章。

心跳產生者協定(Heartbeat Protocol)

心跳協定的通訊模型依循著產生者與消費者(Master/Slave)的關係，使用者可以透過這個協定監測網路中CANopen節點的狀態。其通訊協定如下圖所示：



心跳協定是一種不需要遠端請求訊息(remote frames)的錯誤控制服務(Error Control Service)，一個心跳產生者會循環地產生心跳訊息，會有一個或多個心跳消費者(Heartbeat Consumer)接收這個訊號，而產生者與消費者之間的關係可以經由物件字典來做設定，心跳消費者在心跳消費時間(Heartbeat Consumer Time)內會保持心跳的接受，如果在心跳消費時間內沒有收到心跳的訊息，心跳消費者將會產生一個心跳事件。

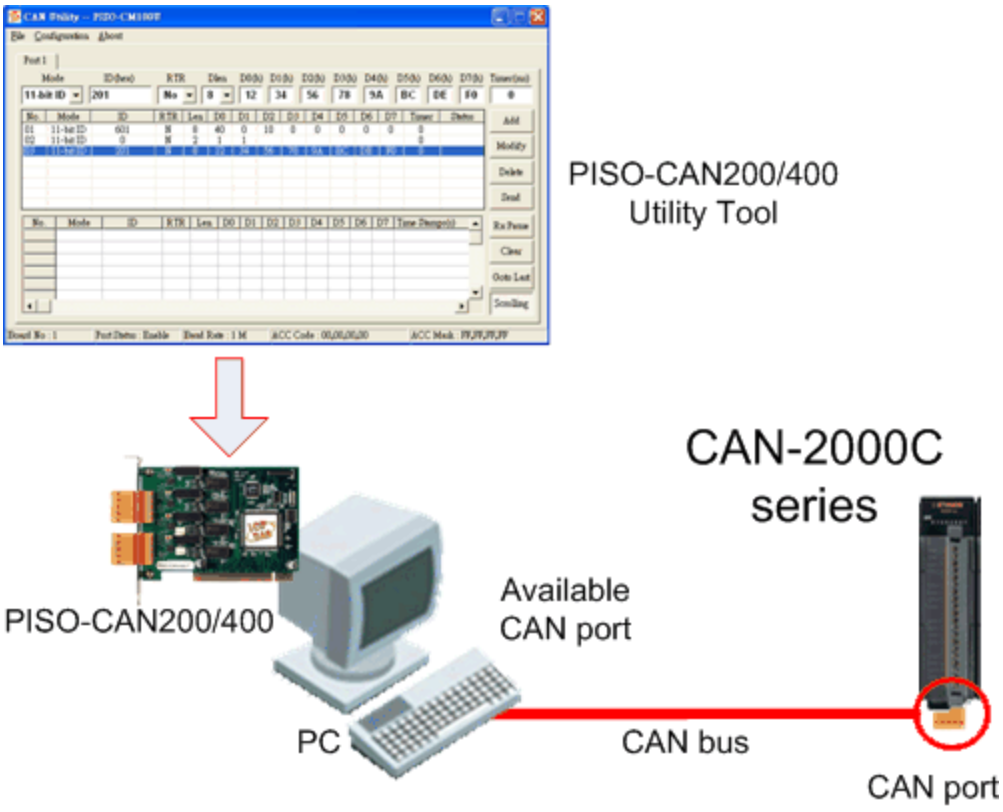
4 開始使用

當開始要使用一個 CAN-2000C 系列模組時，先調整適當的站號 ID 跟鮑率。關電後在重新上電，CANopen 的主端就能收到一個開機訊息。如果沒有收到開機訊息的話，可以先檢查鮑率的設定與接線是否正確。

5 CANopen 通訊集

於這一章內將介紹數種CANopen的通訊協定(communication protocol)，並利用實例來示範如何使用這些協定。為了實作這些例子，我們必須有一個CAN的介面，來和CAN bus上面的CAN-2000C模組進行訊息的溝通，於此處，我們採用了型號為PISO-CAN200/400的CAN介面卡。PISO-CAN200/400是一個具有2/4 CAN埠的PCI介面卡，其提供了易於使用的工具程式，可在PC端透過PISO-CAN200/400對CAN bus上的裝置傳送或接收CAN 2.0A/2.0B的訊息。

本章內各範例所使用的的硬體架構如下所示：



註：有關PISO-CAN200/400工具程式的使用方式，請參考PISO-CAN200/400的使用者手冊。

5.1 SDO 通訊集

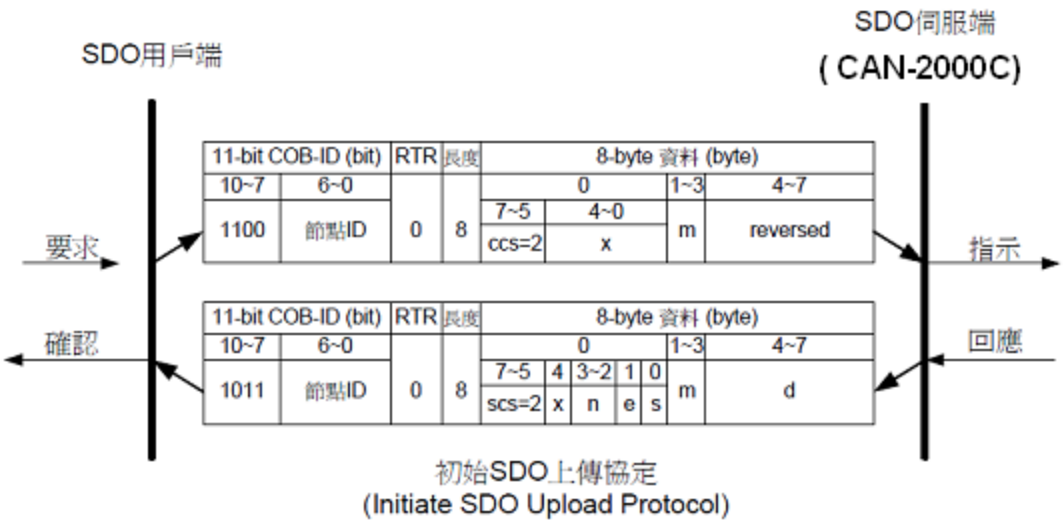
5.1.1 上傳 SDO 協定

<u>初始 SDO 上傳協定(Initiate SDO Upload Protocol)</u>

在傳輸SDO區段(SDO segments)之前，用戶端(client)和伺服端(server)必須先利用初始SDO上傳協定來進行溝通。SDO用戶端可以利用初始SDO上傳協定告訴SDO伺服端，其想要請SDO伺服端上傳的物件為何。

另外，由於初始SDO上傳協定也可以同時夾帶4 bytes的資料進行傳輸。因此，如果SDO用戶端要求SDO伺服端上傳的物件，其資料長度小於或等於4 bytes，則僅使用初始SDO上傳協定便可以完成資料的上傳，也就是不需進行上傳SDO區段協定(Upload SDO segment protocol)。

有關初始SDO上傳協定的細節，使用者可參考下列圖表：



ccs : 用戶端命令識別符(client command specified)
2: 初始上傳要求(initiate upload request)

scs : 伺服端命令識別符(server command specified)
2: 初始上傳回應(nitiate upload response)

n : 只有當**e**=1且**s**=1此欄位才有意義，否則**n**=0。若此欄位有意義則**n**代表**d**欄位內沒有資料的byte數目，即第8-**n**個byte到第7個byte內，沒有節段資料(segment data)。

e : 傳輸形式(transfer type)
0: 正規傳輸(normal transfer)
1: 附件傳輸(expedited transfer)
若**e**=1，即代表欲傳輸物件的資料量小於或等於4 bytes，僅需使用初始SDO上傳協定即可傳送完畢。若**e**=0，就必需要進行上傳SDO區段協定。

s : 資料量指示符(size indicator)
0: 代表幀(frame)內沒有資料大小的資訊。
1: 代表幀(frame)內有資料大小的資訊。

m : 多工器(multiplexer)
其代表欲傳輸SDO物件內含的資料，在物件字典的主索引和子索引。前兩個byte代表主索引，後一個byte代表子索引。

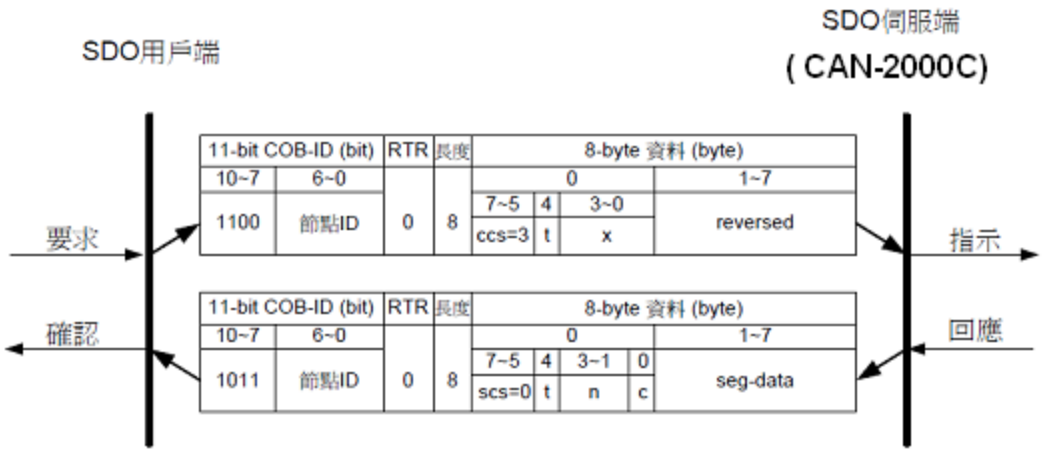
d : 資料
e=0, **s**=0: 表示 **d** 被保留，留待進一步的使用。
e=0, **s**=1: **d** 包含了欲上傳 byte 的數目，其中 byte 4 內含最低有效位元(least significant bit)，byte 7 內含最高有效位元(most significant bit)。
e=1, **s**=1: **d** 的內容為欲上傳的資料，其長度為 4-**n**。資料編碼的方式取決於主索引和子索引在物件字典中參照項目的資料型別。
e=1, **s**=0: **d** 的內容為未標示長度的上傳資料。

x : 沒有被使用，數值永為 0。

reserved : 保留，留待進一步的使用，數值永為 0。

上傳 SDO 區段協定(Upload SDO Segment Protocol)

當欲上傳的資料大小超過4 bytes，除了初始SDO上傳協定，此時還需要透過上傳SDO區段協定來進行資料的上傳。當初始SDO上傳協定結束之後，SDO用戶端便開始利用上傳SDO區段協定，請SDO伺服端上傳資料。有關上傳SDO區段協定的細節如下所示：



上傳SDO區段協定
(Upload SDO Segment Protocol)

ccs : 用戶端命令識別符(client command specified)
3: 上傳節段要求(upload segment reques)

scs : 伺服端命令識別符(server command specified)
0: 上傳節段回應(upload segment response)

t : 交替位元(toggle bit)
對每一連續的上傳節段而言,每一個節段的交替位元均需與其上一個或下一個節段的交替位元不同。而第1個節段的交替位元必須設定為0，另外要求訊息和回應訊息的交替位元必須相同

c : 用來指出是否還有節段要被上傳。
0: 還有節段等待上傳。
1: 已經沒有節段需要上傳。

seg-data : 其內為欲上傳的節段資料，一次最多可上傳7 bytes。資料編碼的方式取決於初始SDO上傳協定內，主索引和子索引在物件字典中參照項目的資料型別。

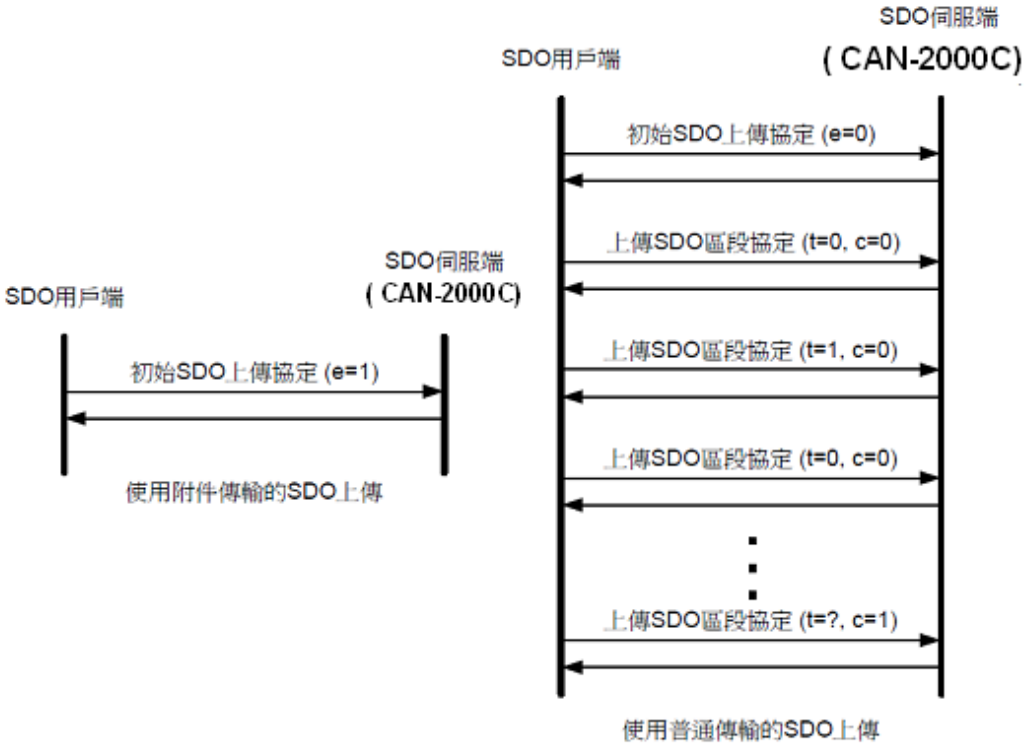
n : **n**內含**seg-data**欄位內沒有節段資料的byte數目，即第8-**n**個byte到第7個byte內，沒有節段資料(segment data)。如果**n**=0，代表節段的大小沒有被指示。

x : 沒有被使用，數值永為 0。

reserved : 保留，留待進一步的使用，數值永為 0。

SDO 上傳範例

下圖為實際利用上傳 SDO 協定上傳資料的過程：



底下將根據上圖，用實際的範例對附件傳輸(expedited transfer)和正規傳輸(normal transfer)的細節做更深入的描述。同時也會示範如何取得儲存在物件字典內的資料。

舉例來說，若使用者想要知道物件字典中主索引為0x1400的物件其子索引數目，則使用者可以透過初始SDO上傳協定，要求SDO伺服端上傳主索引為0x1400且子索引為0x00的物件項目。若使用者想要知道製造商裝置名稱(manufacturer device name)，則可以透過初始SDO上傳協定和上傳SDO區段協定，由SDO伺服端上傳主索引為1008的物件給使用者。

- **附件傳輸(expedited transfer)的範例**

步驟1：傳送如下的SDO訊息給CAN-2000C模組，以取得在物件字典的通訊描述文件區域內，主索引為0x1400且子索引為0x00的項目。有關此項目的作用，使用者可以參照第6章。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0	0	1	0	8	40	00	14	00	00	00	00	00

SDO 用戶端 → **SDO 伺服端 (CAN-2000C)**

ccs : 2

m : 00 14 00

因為低字組會先被傳送，所以第1個byte“00”代表0x1400的低字組，而第2個byte“14”代表0x1400的高字組，至於最後一個byte“00”則代表子索引0x00。

步驟2：CAN-2000C模組將會回應主索引為0x1400且子索引為0x00的物件項目內所儲存的資料。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	0	1	0	5	4F	00	14	00	02	--	--	--

SDO 用戶端 ← **SDO 伺服端 (CAN-2000C)**

scs : 2

n : 3

e : 1

s : 1

m : 00 14 00

d : 02 -- -- --

因為 **n**=3，所以只有第 4 個 byte 會有資料，於此處其值為 0x02。

- **正規傳輸(normal transfer)的範例**

步驟1：傳送SDO訊息給CAN-2000C模組(對CAN-2000C模組而言，此SDO訊息為RxSDO)，以取得儲存在物件字典的通訊描述文件區域中，主索引為0x1008，子索引為0x00的物件項目。關於主索引為0x1008的物件，其內容將於第6章再行介紹。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0	0	1	0	8	40	08	10	00	00	00	00	00

SDO 用戶端 ⟶ **SDO 伺服端 (CAN-2000C)**

ccs : 2
m : 08 10 00

步驟2：CAN-2000C模組會回應SDO訊息以告知使用者，CAN-2000C預計上傳給使用者的資料量(byte)。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	0	1	0	8	41	08	10	00	09	00	00	00

SDO 用戶端 ⟵ **SDO 伺服端 (CAN-2000C)**

scs : 2
n : 0
e : 0
s : 1
m : 08 10 00
d : 09 00 00 00

因為**e**=0且**s**=1，所以**d**代表CAN-8423預計上傳給使用者的資料量(byte)。而“09”代表資料長度中的最低的字組，因此“09 00 00 00”這4個byte就代表CAN-2000C模組將上傳9個bytes給使用者。

步驟 3：於此處傳送以下訊息，要求 CAN-2000C 模組開始傳送資料。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0	0	1	0	8	60	00	00	00	00	00	00	00

SDO 用戶端 → **SDO 伺服端 (CAN-2000C)**

ccs : 3
t : 0

步驟4：CAN-2000C模組會回應主索引0x1008，子索引0x00的項目內，第0個byte到第7個byte

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	0	1	0	8	00	43	41	4E	2D	32	30	38

SDO 用戶端 ← **SDO 伺服端 (CAN-2000C)**

scs : 0
t : 0
n : 0
c : 0
seg-data : 43 41 4E 2D 32 30 38

使用者若是查閱第6章，可發現主索引0x1008，子索引0x00的項目，其資料型態為“VISIBLE_STRING”。因此，使用者在接收到這個物件之後，可根據ASCII表來對**seg-data**內的資料做轉換，此處轉換之後，這7個字元分別為“CAN-208”。

步驟 5：於此處傳送以下訊息，要求 CAN-2000C 模組傳送剩餘的資料。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0	0	1	0	8	70	00	00	00	00	00	00	00

SDO 用戶端 ⟶ **SDO 伺服端 (CAN-2000C)**

ccs : 3
t : 1

步驟 6：接收由 SDO 伺服端(CAN-2000C 模組)傳送來的剩餘資料

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	0	1	0	3	1B	38	00	--	--	--	--	--

SDO 用戶端 ⟵ **SDO 伺服端 (CAN-2000C)**

scs : 0
t : 1
n : 5
c : 1
seg-data : 38 00 -- -- -- -- --

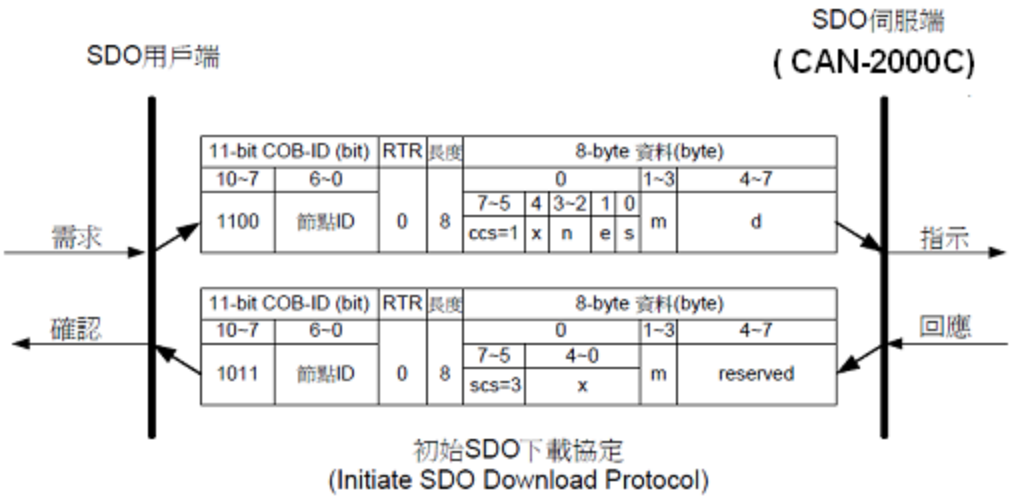
因為**n**=5，也就是說在**data**欄位內，只有前3個bytes的資料有意義，而0x1B為上傳SDO節段協定的標頭，扣掉這個byte，剩下的0x38和0x00才是資料。對照ASCII表，0x38和0x00分別代表字元“8”和字元“Null”。

5.1.2 下載 SDO 協定

初始 SDO 下載協定(Initiate SDO Download Protocol)

下載SDO協定(Download SDO Protocol)與上傳SDO協定十分類似，其通訊協定僅有部份參數與上傳SDO協定不同。

下載SDO協定也可被分為兩個部份，包括初始SDO下載協定和下載SDO區段協定。若欲下載的資料長度小於4 bytes，則僅使用初始SDO下載協定即可完成資料的下載，若欲下載的資料長度大於4 bytes，除了初始SDO下載協定，還必須進行下載SDO區段協定，才能把資料下載完畢。底下將針對這兩種協定進行描述



初始SDO下載協定
(Initiate SDO Download Protocol)

ccs : 用戶端命令識別符(client command specifier)
1：初始下載要求(initiate download request)

scs : 伺服端命令識別符(server command specifier)
3：初始下載回應(initiate download response)

n : 只有當**e**=1且**s**=1此欄位才有意義，否則**n**=0。若此欄位有意義，則**n**代表**d**欄位內沒有資料的byte數目，即第8-**n**個byte到第7個byte內，沒有節段資料(segment data)。

e : 傳輸型式(transfer type)
0：正規傳輸(normal transfer)
1：附件傳輸(expedited transfer)
若**e**=1，即代表欲傳輸物件的資料量小於或等於4 bytes，僅需使用初始SDO下載協定即可傳送完畢。若**e**=0，就必需要進行下載SDO區段協定。

s : 資料量指示符(size indicator)
0：代表幀(frame)內沒有資料集(Data Set)大小的資訊。
1：代表幀(frame)內有資料集(Data Set)大小的資訊。

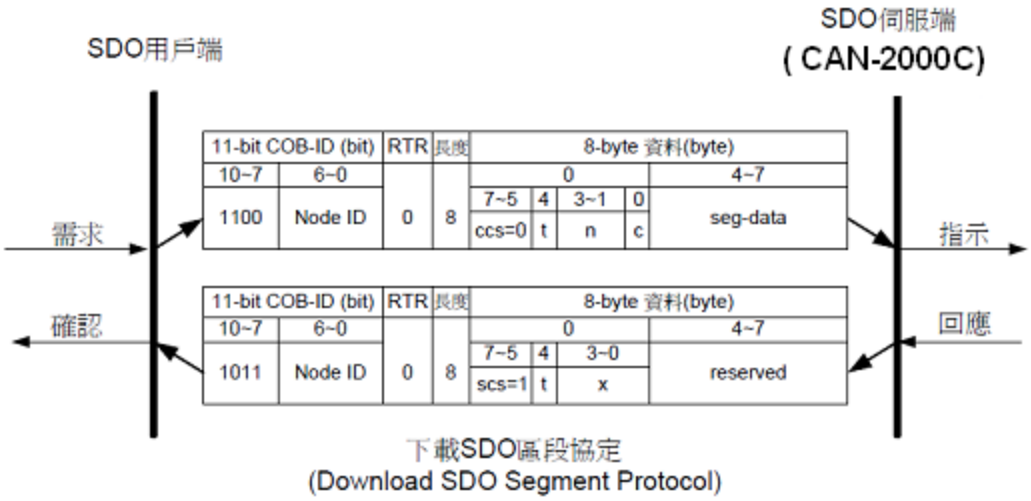
m : 多工器(multiplexor)
其代表欲傳輸SDO物件其內含的資料在物件字典的主索引和子索引。前兩個byte代表主索引，後一個byte代表子索引。

d : 資料
e=0,**s**=0: 表示 **d** 被保留，留待進一步的使用。
e=0,**s**=1: **d** 包含了欲上傳 byte 的數目，其中 byte 4 內含最低有效位元(least significant bit)，byte 7 內含最高有效位元(most significant bit)。
e=1,**s**=1: **d** 的內容為欲上傳的資料，其長度為 4-**n**。資料編碼的方式取決於主索引和子索引在物件字典中參照項目的資料型別。
e=1,**s**=0: **d** 的內容為未標示長度的上傳資料。

x : 沒有被使用，數值永為 0。

reserved : 保留，留待進一步的使用，數值永為 0。

下載節段協定(Download Segment Protocol)



ccs ： 用戶端命令識別符(client command specifier)
0：下載節段要求(download segment request)

scs ： 伺服端命令識別符(server command specifier)
1：下載節段回應(download segment response)

seg-data ： 其內為欲下載的節段資料，一次最多可下載7 bytes。資料編碼的方式取決於初始SDO下載協定內，主索引和子索引在物件字典中參照項目的資料型別。

n ： **n**內含**seg-data**欄位內沒有節段資料的byte數目，即第8-**n**個byte到第7個byte內，沒有節段資料(segment data)。如果**n** = 0，代表節段的大小沒有被指示。

c ： 用來指出是否還有節段要被上傳。
0：還有節段等待上傳。
1：已經沒有節段需要上傳。

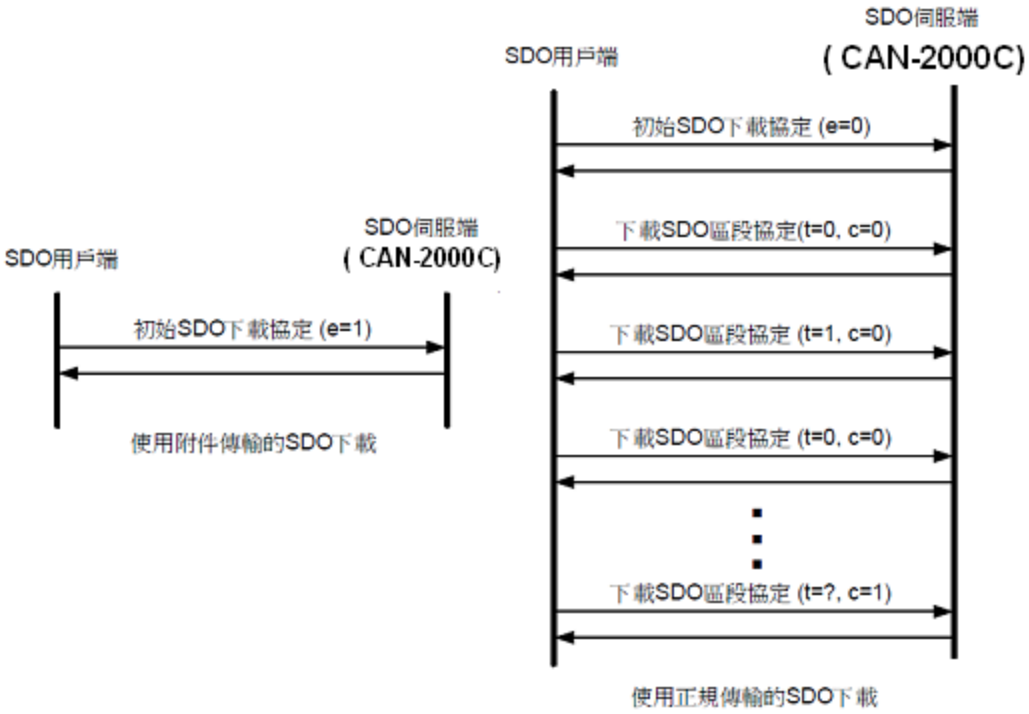
t ： 交替位元(toggle bit)
對每一連續的下載節段而言，每一個節段的交替位元均需與其上一個或下一個節段的交替位元不同。而第1個節段的交替位元必須設定為0，另外要求訊息和回應訊息的交替位元必須相同。

x ： 沒有被使用，數值永為 0。

reserved ： 保留，留待進一步的使用，數值永為 0。

SDO 下載範例

底下將使用實際的例子來示範如何用下載 SDO 協定來進行資料的下載：



由於在CAN-2000C模組內，所有可被寫入的物件項目，其資料長度均不超過4 bytes，因此底下僅示範如何使用附件傳輸。

- **附件傳輸(expedited transfer)的範例**

步驟1：傳送Rx SDO訊息給CAN-2000C模組，以更改在通訊描述文件區域內，主索引為0x1400且子索引為0x02的物件項目，於此處將此物件項目的值更改為5。另外假設CAN-2000C模組的節點ID被設定為1。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0	0	1	0	8	2F	00	14	02	05	00	00	00

SDO 用戶端 → **SDO 伺服端 (CAN-2000C)**

ccs : 1
n : 3
e : 1
s : 1
m : 00 14 02
d : 05 00 00 00
因為 **n**=3，所以只有第 4 個 byte 會有資料，於此處其值為 0x05

步驟 2：CAN-2000C 模組將會回覆此訊息以結束資料的下載。在結束資料的下載後，使用者可以使用前面所介紹過的上傳 SDO 通訊協定來把寫入的資料再讀出來比對，以確定資料是否正確地下載至 CAN-2000C 模組
。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	0	1	0	4	60	00	14	02	--	--	--	--

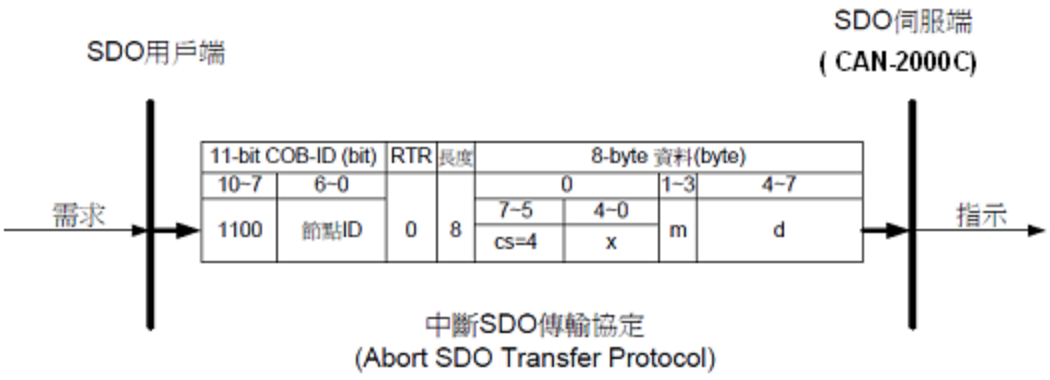
SDO 用戶端 ← **SDO 伺服端 (CAN-2000C)**

scs : 3
m : 00 14 02

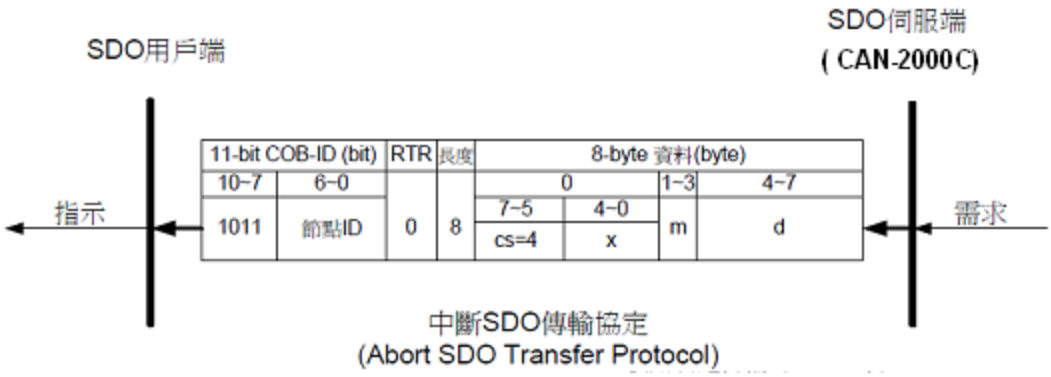
5.1.3 中斷 SDO 傳輸協定

在某些情況下，SDO用戶端和SDO伺服端會需要中斷SDO的傳輸。舉例來說，像是想要更改的物件項目是唯讀的，或著要存取的物件項目根本不存在，亦或是用戶端和伺服端基於其它理由不想完成SDO的傳輸。當這些情況發生時，SDO用戶端和SDO伺服端均可以主動的透過中斷SDO傳輸協定，傳送中斷訊息以中斷SDO的傳輸。

底下將介紹中斷SDO傳輸協定(Abort SDO Transfer Protocol)：



中斷SDO傳輸協定
(Abort SDO Transfer Protocol)



中斷SDO傳輸協定
(Abort SDO Transfer Protocol)

cs ： 命令識別符(command specifier)
4：中斷傳輸要求(abort transfer request)

x ： 沒有被使用，數值永為 0。

m ： 多工器(multiplexor)
其代表欲中斷SDO物件其內含的資料在物件字典的主索引和子索引。前兩個byte代表主索引，後一個byte代表子索引。

d ： 內含 4 bytes 的中斷碼(abort code)，其代表中斷的原因。

中斷碼 (Abort code)	描述
0503 0000h	交替位元(Toggle bit)沒有變化
0504 0000h	SDO 協定逾時(timed out)。
0504 0001h	用戶端/伺服端的命令識別符(command specifier)無效或著無法解譯。
0504 0002h	無效的區塊大小。(僅限區塊模式)
0504 0003h	無效的序號(sequence number)。(僅限區塊模式)
0504 0004h	CRC 錯誤。(僅限區塊模式)
0504 0005h	記憶體不足。
0601 0000h	物件的存取不被支援。(不可被讀寫)
0601 0001h	嘗試讀取唯寫(write only)的物件。
0601 0002h	嘗試寫入唯讀(read only)的物件。
0602 0000h	物件字典內不存在此物件。
0604 0041h	物件無法映射(mapped)給 PDO。
0604 0042h	被映射的物件數量和長度超過 PDO 所能承載的負荷。
0604 0043h	一般的參數設定相容性問題。
0604 0047h	一般的裝置內部相容性問題。
0606 0000h	基於硬體錯誤導致的存取失敗。
0607 0010h	服務參數(service parameter)長度不符引起的資料型別不符。
0607 0012h	服務參數(service parameter)長度過長引起的資料型別不符。
0607 0013h	服務參數(service parameter)長度過短引起的資料型別不符。
0609 0011h	子索引不存在。
0609 0030h	欲寫入的參數，其數值超出範圍。
0609 0031h	欲寫入的參數數值過高。
0609 0032h	欲寫入的參數數值過低。
0609 0036h	最大數值小於最小數值。
0800 0000h	一般性的錯誤。
0800 0020h	資料無法傳送或著儲存到應用程式(application)。
0800 0021h	因為本地端控制(local control)，資料無法傳送或著儲存到應用程式。
0800 0022h	因為目前的裝置狀態，資料無法傳送或著儲存到應用程式。
0800 0023h	物件字典在動態產生時發生錯誤，或著物件字典不存在(若物件字典的產生必須藉由載入檔案來完成，萬一檔案載入發生發生錯誤，那麼物件字典就無法被建立)。

中斷 SDO 傳輸範例

在物件字典中，主索引為0x1008的物件，沒有子索引0x01的項目。因此，若使用者想要透過SDO協定來讀取主索引為0x1008且子索引為0x01的物件項目，此時CAN-2000C模組就會回應中斷SDO傳輸訊息。於下將詳細描述這個過程：

步驟1：SDO用戶端透過初始SDO上傳協定，傳送RxSDO訊息給CAN-2000C模組，表示希望讀取物件字典中主索引為0x1008且子索引為0x01的物件項目。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0	0	1	0	8	40	08	10	01	00	00	00	00

SDO 用戶端 ⟶ **SDO 伺服端 (CAN-2000C)**

ccs : 2

m : 08 10 01

步驟 2：CAN-2000C 模組會回應中斷 SDO 訊息給 SDO 用戶端。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	0	1	0	8	80	08	10	01	11	00	09	06

SDO 用戶端 ⟵ **SDO 伺服端 (CAN-2000C)**

cs : 4

m : 08 10 01

d : 11 00 09 06

因為低字組會被先傳送，所以接收下來的資料在轉換之後，正確的順序應該為"06 09 00 11"。若是查找上一頁的中斷碼表格，將會發現中斷碼代表的意義為"子索引不存在"。

5.2 PDO 通訊集

5.2.1 PDO COB-ID 參數

每一個PDO在物件字典內都會有其對應的PDO通訊參數(PDO communication parameter)，在使用PDO之前，必須要先查詢物件字典中，PDO通訊參數物件內的COB-ID項目(子索引0x01)。COB-ID項目內記錄了PDO在傳輸時會使用的COB-ID，其共有32 bits，而此處將每一個bit所代表的意義整理如下表：

Bit 編號	值	代表的意義
31 (MSB)	0	PDO 存在 (此 PDO 是有效的，valid)
	1	PDO 不存在 (此 PDO 是無效的，invalid)
30	0	此 PDO 允許 RTR 的傳輸方式
	1	此 PDO 不允許 RTR 的傳輸方式
29	0	11-bit ID (CAN 2.0A)
	1	29-bit ID (CAN 2.0B)
28-11	0	若 bit 29=0，此欄位的數值便為 0
	x	若bit 29=1：則此欄位就是29bits COB-ID 內的第 28~11 bits
10-0 (LSB)	x	10-0 bits of COB-ID

註：CAN-2000C模組僅支援CAN 2.0 A，也就是僅支援11 bit的COB-ID。

此處將預設的 PDO COB-ID 參數整理如下表：

PDO 的編號	預設的 PDO COB-ID	
	Bit10~Bit7 (功能碼)	Bit6~Bit0
TxPDO1	0011	節點 ID
TxPDO2	0101	節點 ID
TxPDO3	0111	節點 ID
TxPDO4	1001	節點 ID
RxPDO1	0100	節點 ID
RxPDO2	0110	節點 ID
RxPDO3	1000	節點 ID
RxPDO4	1010	節點 ID

註： 1. 除了在3.1節內，提到被保留的COB-ID使用者不可以拿來自行定義之外，其他的COB-ID使用者均可以拿來定義為PDO的COB-ID。當使用者欲自行定義PDO的COB-ID時，必須小心避免在同一節點(同一裝置)上，某COB-ID被不同COB重複使用的情形。

2. 若PDO為有效狀態(bit 31=0)，則此時PDO的COB-ID參數便不允許被更改。

5.2.2 傳輸型態

PDO通訊參數內含數個具有不同作用的參數，其中子索引為0x02的參數為傳輸型態(transmission type)，而每一個PDO均可對其設定傳輸型態。我們可以透過傳輸型態來了解此PDO在傳送與接收時的特性。

舉例來說，若使用者設定第1個TxPDO的傳輸型態為0，則CANopen的裝置便會利用非循環同步的方式來進行第1個TxPDO的傳輸。此處將不同傳輸型態與其對應的PDO特性之關係整理如下表。

傳輸型態	**PDO** 傳輸方式				
	循環	**非循環**	**同步**	**非同步**	**唯遠端傳送要求**
0		**O**	**O**		
1-240	**O**		**O**		
241-251	----------保留----------				
252			**O**		**O**
253				**O**	**O**
254				**O**	
255				**O**	

註： 1. TxPDO的傳輸型態若是1-240，則代表需要接收到這麼多個SYNC物件才能夠觸發TxPDO的傳送。RxPDO的傳輸型態若是0-240，則僅需要一個SYNC物件的接收便可以啟用在此之前收到且尚未被啟用的RxPDO物件，與傳輸型態的數字大小無關。

2. 只有TxPDO的傳輸型態可以被設定為252和253。傳輸型態若被設定為252，則代表裝置在接收到SYNC物件時，才會更新TxPDO內的資料。傳輸型態若被設定為253，則在接收到RTR訊息時，裝置才會更新TxPDO內的資料。TxPDO若是被設定為這兩種型態，則只有在接收到此TxPDO的RTR訊息時，裝置才會對外傳送TxPDO。

3. 傳輸型態若是被設定為254和255，便可以使用事件計時器(event timer)來觸發TxPDO的傳送。另外若某DI被映射到某個PDO，當此DI的值被變更時，也會觸發其對應TxPDO的傳送。對RxPDO而言，若是傳輸型態被設定為254或255，則在接收到RxPDO之後，就必須立即啟用此RxPDO。

5.2.3 PDO 通訊規則

根據CANopen DS-301的規範，與PDO有關的物件乃存放於物件字典中主索引0x1400到0x1BFF之間。而在CAN-2000C模組內，其RxPDO的通訊參數(communication parameter)位於物件字典中主索引0x1400到0x140F之間，其RxPDO的映射參數(mapping parameter)位於物件字典中主索引0x1600到0x160F之間，其TxPDO的通訊參數位於物件字典中主索引0x1800到0x180F之間，其TxPDO的映射參數位於物件字典中主索引0x1A00到0x1A0F之間。此外，每一個PDO的通訊參數物件均會對應到一個映射參數物件，兩者之間為一對一的關係。

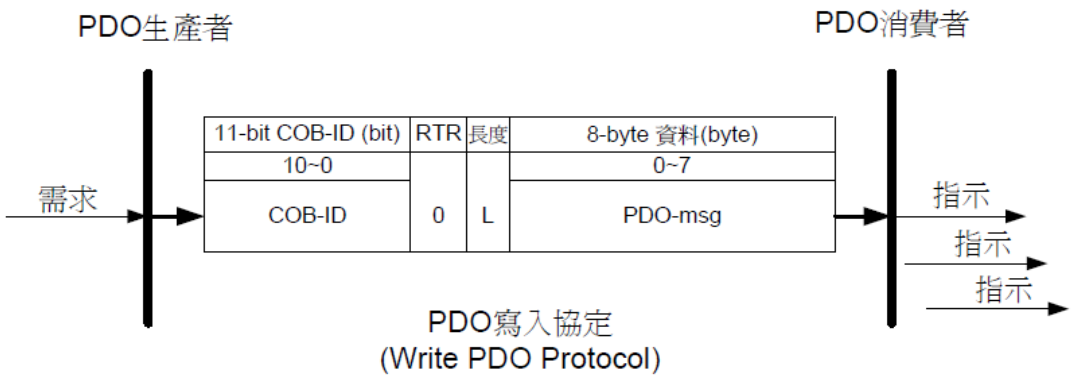
更進一步的來說，譬如第1組RxPDO通訊參數存放於物件字典中主索引為0x1400的地方，而其相對應的映射參數便會存放於物件字典中主索引為0x1600的地方，可依序推得而主索引0x1401和0x1601為一對，主索引0x1402和0x1602為一對等…TxPDO的通訊參數和映射參數的關係同樣依循這樣的關係，譬如第1個TxPDO通訊參數存放於物件字典中主索引為0x1800的地方，而其相對應的映射參數便會存放於物件字典中主索引為0x1A00的地方，可依序推得而主索引0x1801和0x1A01為一對，主索引0x1802和0x1A02為一對等…在使用者開始利用PDO對實際的I/O通道作存取前，必須要先取得PDO的通訊參數和映射參數。

此外，PDO的通訊只能在NMT的操作(operational)狀態下使用，若使用者要使用PDO來進行資料的傳輸，可以透過NMT模組控制協定(NMT module control protocol)，傳送模組控制訊息給CAN-2000C模組，要求裝置改變NMT狀態為操作狀態。詳細的內容將於5.3節內作介紹。

附帶一提，透過PDO來傳送訊息，PDO內的資料長度必須和其對應PDO映射參數內所記載的資料長度相吻合，當PDO消費者收到PDO訊息時，會根據此PDO的COB-ID來查找相對應的RxPDO映射參數。若此PDO內的資料長度(假設為L bytes)大於其映射參數所記載的長度(假設為n bytes)，則PDO消費者只會取前n bytes來使用，其餘部份則丟棄。若此PDO內的資料長度小於其映射參數所記載的長度，則PDO消費者將不會處理這個PDO，並且會發出一個錯誤碼為8210h的EMCY(Emergency)訊息給PDO的生產者。

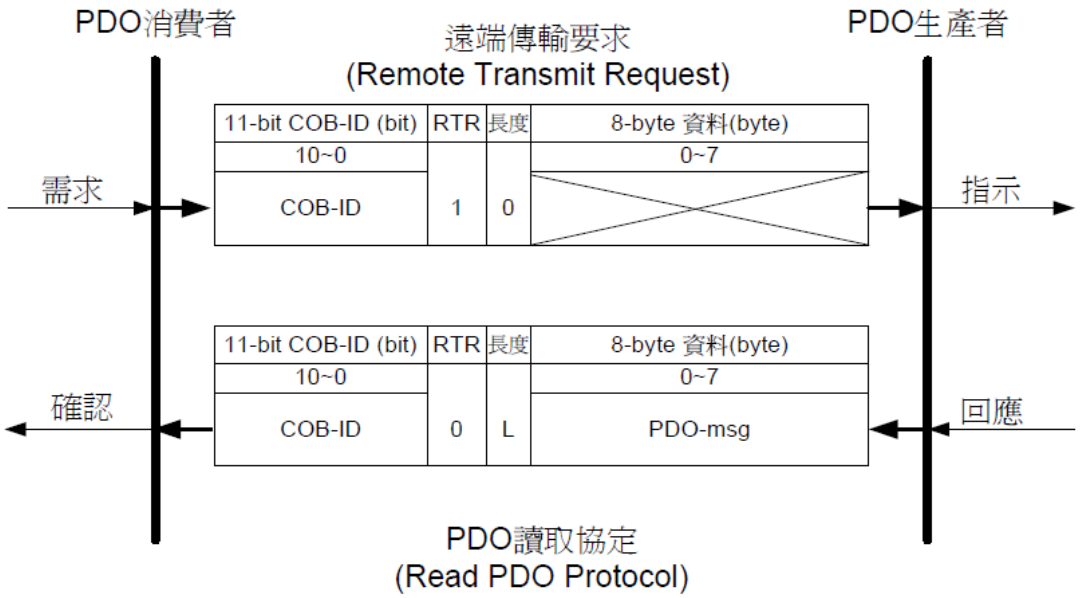
註： 關於EMCY訊息，請參照5.3節。

底下為PDO的通訊協定：



PDO寫入協定
(Write PDO Protocol)

COB-ID ： 預設的 PDO COB-ID，或是使用者定義的 PDO COB-ID。
L ： PDO 訊息所使用的資料長度(bytes)。
PDO-msg ： 即時性的資料，或著可以用作 PDO 映射的資料。

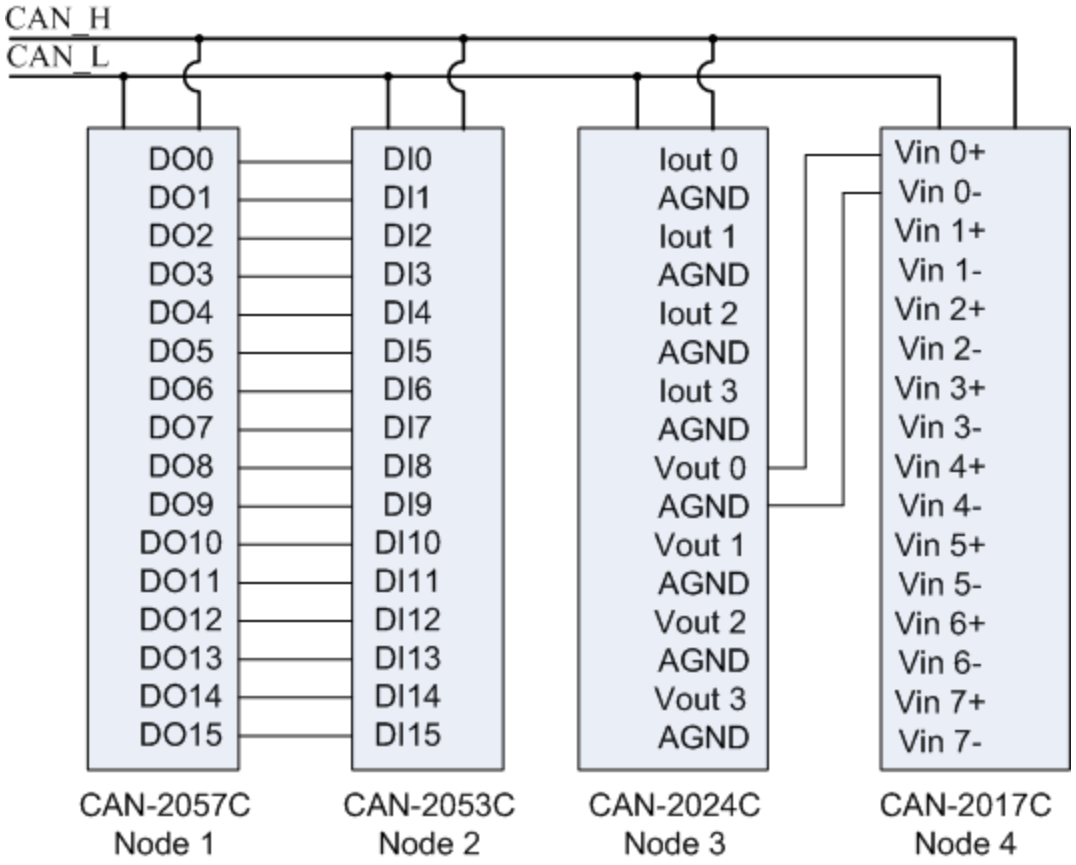


PDO讀取協定
(Read PDO Protocol)

COB-ID ： 預設的 PDO COB-ID，或是使用者定義的 PDO COB-ID
L ： PDO 訊息所使用的資料長度(bytes)。
PDO-msg ： 即時性的資料，或著可以拿來作 PDO 映射的資料。

PDO 通訊範例

在開始敘述PDO的通訊範例之前，我們會需要使用四個CAN-2000C模組，分別為CAN-2057C、CAN-2053C、CAN-2024C以及CAN-2017C。並將這些模組的輸出輸入埠連接如下圖：



利用CAN-2000C模組面板上的旋鈕，設定裝置的節點ID為1到4，設定CAN bus的鮑率為125Kbps。再參考使用手冊分別將CAN-2024C和CAN-2017C輸入/輸出範圍設定為-10V ~ +10V。此四個模組的資訊如下:

PDO 資訊:

節點	名稱	PDO 類型	COB-ID	傳輸類別	抑制時間	事件計時器
1	CAN-2057C	Rx PDO	0x201	0xFF	0x00	0x00
3	CAN-2024C	Rx PDO	0x303	0xFF	0x00	0x00
2	CAN-2053C	Tx PDO	0x182	0xFF	0x00	0x00
4	CAN-2017C	Tx PDO	0x284	0xFF	0x00	0x00
4	CAN-2017C	Tx PDO	0x384	0xFF	0x00	0x00

PDO I/O 映射資訊:

COB-ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x201	DO 0~7	DO 8~15						
0x303	AO 00_L	AO 0_H	AO 1_L	AO 1_H	AO 2_L	AO 2_H	AO 3_L	AO 3_H
0x182	DI 0~7	DI 8~15						
0x284	AI 0_L	AI 0_H	AI 1_L	AI 1_H	AI 2_L	AI 2_H	AI 3_L	AI 3_H
0x384	AI 4_L	AI 4_H	AI 5_L	AI 5_H	AI 6_L	AI 6_H	AI 7_L	AI 7_H

SDO I/O 埠資訊:

名稱	CAN-2057C	CAN-2024C	CAN-2053C	CAN-2017C
節點	1	3	2	4
主索引	0x6200	0x6411	0x6000	0x6401
描述	DO	AO	DI	AI
子索引 0	2	4	2	8
子索引 1	00 ~ 07 ch	00 ch	00 ~ 07 ch	00 ch
子索引 2	07 ~ 15 ch	01 ch	07 ~ 15 ch	01 ch
子索引 3		02 ch		02 ch
子索引 4		03 ch		03 ch
子索引 5				04 ch
子索引 6				05 ch
子索引 7				06 ch
子索引 8				07 ch

在這一節內，將會利用實際的例子來示範如何使用PDO通訊的各種功能，於下會介紹的項目有：

- 利用非同步的 PDO 來存取數位 I/O 和類比 I/O。
- 利用事件計時器來取得輸入的數值。
- 非循環同步 RxPDO 的功能展示。
- 非循環同步 TxPDO 的功能展示。
- 循環同步 TxPDO 的功能展示。

- 同步、唯遠端傳輸要求 TxPDO 的功能展示。
- 非同步、唯遠端傳輸要求 RxPDO 的功能展示。
- 針對 DI/AI/DO/AO 通道的動態 PDO 映射。

另外假設於範例中所使用的通訊物件，其 COB-ID 均為預設值。

在開始用例子來示範PDO通訊的各項功能之前，必須先更改裝置的NMT狀態，因為PDO傳輸只能在NMT操作狀態(operational state)下進行。

步驟0：CAN-2000C模組在接收到以下的訊息時，若無異常，便會更改其NMT狀態為NMT操作狀態。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0	0	0	0	8	01	01	00	00	00	00	00	00

NMT 主端 → **NMT 僕端 (CAN-2000C)**

cs : 1

Node ID : 1

- 存取數位 **I/O** 和類比

步驟1：如果要更改CAN-2057C的數位輸出值為0x1234，使用者可以傳送物件字典中的第1組RxPDO給CAN-2057C，如下：

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	1	0	0	0	0	0	0	0	0	1	0	8	34	12	00	00	00	00	00	00

PDO 生產者 → **PDO 消費者 (CAN-2057C)**

COB-ID : 0x201

L : 8

PDO-msg : 34 12 00 00 00 00 00 00

根據裝置預設的 PDO 映射，可以知道第 1 組 RxPDO 內只有 2 bytes 的資料。即使此處將資料長度 L 設定為 8，CAN-2057C 在收到這個 PDO 之後，依據這個 PDO 的映射參數，還是會把後 6 個 bytes 丟掉，僅處理前 2 個 bytes 的資料。而第 1 個 byte 的內容為 CAN-2057C 上通道 DO0~DO7 的數值，第 2 個 byte 的內容為 CAN-2057C 上通道 DO8~DO15 的數值。

步驟2：CAN-2053C / CAN-2057C的DI/DO通道特性如下，DI/DO通道的數值若為1，則代表DI/DO通道目前為低邏輯準位，I/O模組上的LED燈號為ON，DI/DO通道的數值若為0，則代表DO為高邏輯準位，I/O模組上的LED燈號為OFF。

根據CANopen DS-401，DO通道輸出預設值為0，故上個步驟執行完之後，CAN-2057C的DO值會由0x0000變為0x1234。即代表其DO2、DO4、DO5、DO9、DO12的LED燈號會由OFF轉ON，且其輸出由高邏輯準位變為低邏輯準位，其餘DO通道則反之。

此時CAN-2053C的DI2、DI4、DI5、DI9、DI12的LED燈號會變由OFF轉ON，且會測量到DI數值為由0x0000變為0x1234。

CAN-2053C的DI發生變化，連帶會使得CAN-2053C儲存到物件字典內的DI通道數值也會跟著改變，由於CAN-2053C的16個DI通道是映射給第1個TxPDO，且此PDO的傳輸型態為255，所以當一偵測到通道狀態的改變，就會立即傳送第1組TxPDO給PDO消費者，如下：

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	0	1	1	0	0	0	0	0	1	0	0	2	34	12	--	--	--	--	--	--

PDO 消費者 ← **PDO 消費者 (CAN-2053C)**

COB-ID : 0x182

L : 2

PDO-msg : 34 12 -- -- -- -- -- --

因為此處的資料長度設定為 2，因此只有前 2 bytes 是有效的。

步驟3：使用者可以透過傳送如下的第2組RxPDO訊息，使CAN-2024C的通道AO0輸出5V。(根據CANopen DS-401，AO通道輸出預設值為0x0000)

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	1	1	0	0	0	0	0	0	1	1	0	8	FF	3F	00	00	00	00	00	00

PDO 生產者 → **PDO 消費者 (CAN-2024C)**

COB-ID : 0x303

L : 8

PDO-msg : FF 3F 00 00 00 00 00 00

此處8 bytes的資料分別映射到CAN-2024C的4個AO通道，每個AO通道佔用2 bytes，依序分別為通道AO0、AO1、AO2、AO3。CAN-2024C的輸出範圍為-10V~10V。根據附錄的轉換表，可知CAN-2024C預設的通道值float和通道值hex範圍分別為-10V~10V和0x8000(-32768)~0x7FFF(32767)。若我們欲透過AO0輸出5V，透過下面的轉換公式

$$HexValue = \left(\frac{5V-(-10V)}{10V-(-10V)}\right) * \left(32767-(-32768)\right) + (-32768)$$

$$= 16383.25 \approx 16383 = 0x3FFF$$

可知PDO訊息的前2 bytes需填入“FF” 和“3F” 。因為此處其餘AO通道的輸出欲設定為0V，所以剩下6 bytes便填入“000000” 。

步驟4：若AI通道所量測到的數值發生變化，此時CAN-2017C模組並不會自動傳送AI對應的TxPDO(第2組TxPDO)給PDO消費者。因此，使用者需要傳送第2組TxPDO的RTR訊息，要求CAN-2017C模組傳送AI通道的數值給使用者。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	1	0	1	0	0	0	0	1	0	0	1	0	00	00	00	00	00	00	00	00

PDO 消費者 → **PDO 消費者 (CAN-2017C)**

COB-ID : 0x284

步驟 5：CAN-2017C 在收到 RTR 訊息之後，就會傳送第 2 組 TxPDO 給使用者。回傳的值為 5V。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	1	0	1	0	0	0	0	0	0	1	0	8	**00**	**40**	**FD**	**FF**	**FD**	**FF**	**FD**	**FF**

PDO 消費者 ← **PDO 消費者 (CAN-2017C)**

COB-ID : 0x284

L : 8

PDO-msg : 00 40 FD FF FD FF FD FF

這 8 bytes 分別代表為 CAN-2017C 的 AI0、AI1、AI2、AI3 通道所量到的 AI 數值，根據低字組先傳的原則，可知“00 40” 代表 AI0 量到的數值為 0x4000(16384)。據附錄的轉換表，可知 CAN-2017C 預設的通道值 float 和通道值 hex 範圍分別為 -10V~10V 和-32768~32767。透過轉換公式，可以將 0x4000 轉換為實際的通道數值，如下：

$$FloatValue = \left(\frac{16384-(-32768)}{32767-(-32768)}\right) * (10V-(-10V)) + (-10V)$$

$$\approx 5.001V$$

- 事件計時器**(Event Timer)**功能展示

步驟6：此處利用SDO存取物件字典內主索引為0x1801且子索引為0x05的項目，將第2組TxPDO的事件計時器數值設定為1000。由於事件計時器內數值的單位為毫秒，所以此處的1000即代表1秒。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	1	0	0	0	8	2B	01	18	05	E8	03	00	00

SDO 用戶端 → **SDO 伺服端 (CAN-2017C)**

ccs : 1
n : 2
e : 1
s : 1
m : 01 18 05
d : E8 03 00 00
16進制的0x03E8即代表十進制的1000。
此外，因為**n**=2，所以最後2 bytes 的內容“0000”不具任何意義。

步驟 7：如果 SDO 傳輸成功，CAN-2017C 會回覆以下訊息以結束 SDO 的通訊。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	1	0	0	0	4	60	01	18	05	--	--	--	--

SDO 用戶端 ← **SDO 伺服端 (CAN-2017C)**

scs : 3
m : 01 18 05

步驟8：在事件計時器的數值被改變之後，第2組的TxPDO便會每1秒傳送1次。底下是第1次接收到的第2組TxPDO訊息。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	1	0	1	0	0	0	0	1	0	0	0	8	00	40	FD	FF	FF	FF	FF	FF

PDO 消費者 ← **PDO 生產者 (CAN-2017C)**

COB-ID : 0x284
L : 8
PDO-msg : 00 40 FD FF FF FF FF FF

步驟 9：底下是第 2 次接收到的第 2 組 TxPDO 訊息。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	1	0	1	0	0	0	0	1	0	0	0	8	06	40	FF	FF	FF	FF	FF	FF

PDO 消費者 ← **PDO 生產者 (CAN-2017C)**

COB-ID : 0x284
L : 8
PDO-msg : 06 40 FF FF FF FF FF FF

0x4006 在轉換為通道值(float)後，相當於 5.002V，此處 AI 的數值會出現小幅度變動的原因，可能是因為雜訊的干擾或著其他的因素。類比輸出輸入，數值變動的範圍須參考硬體本身的規格。

步驟 10：底下是第 3 次接收到的第 2 組 TxPDO 訊息。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	1	0	1	0	0	0	0	1	0	0	0	8	00	40	FF	FF	FD	FF	FF	FF

PDO 消費者 ← **PDO 生產者 (CAN-2017C)**

COB-ID : 0x284
L : 8
PDO-msg : 00 40 FF FF FD FF FF FF

步驟 11：將事件計數器的數值設定回 0，以結束事件計數器的功能測試。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	1	0	0	0	8	2B	01	18	05	00	00	00	00

SDO 用戶端 → **SDO 伺服端 (CAN-2017C)**

ccs : 1
n : 2
e : 1
s : 1
m : 01 18 05
d : 00 00 00 00

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	1	0	0	0	4	60	01	18	05	--	--	--	--

SDO 用戶端 ← **SDO 伺服端 (CAN-2017C)**

scs : 3
m : 01 18 05

- 非循環同步 **RxPDO** 的功能展示**(**傳輸型態為 **0)**

步驟 12：設定第 1 組 RxPDO 的傳輸型態為 0

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0	0	1	0	8	2F	00	14	02	00	00	00	00

SDO 用戶端 → **SDO 伺服端 (CAN-2057C)**

ccs : 1
n : 3
e : 1
s : 1
m : 00 14 02
d : 00

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	0	1	0	4	60	00	14	02	--	--	--	--

SDO 用戶端 ← **SDO 伺服端 (CAN-2057C)**

scs : 3
m : 00 14 02

步驟13：使用第1組RxPDO通知CAN-2057C，準備將CAN-2057C的DO數值變更為0x5678。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	1	0	0	0	0	0	0	0	0	1	0	8	78	56	00	00	00	00	00	00

PDO 生產者 → **PDO 消費者 (CAN-2057C)**

COB-ID : 0x201

L : 8

PDO-msg : 78 56 00 00 00 00 00 00

步驟14：在進行上個步驟之後，DO通道數值並不會馬上改變，這是因為此時第1組的RxPDO，其傳輸型態被設定為0(非循環同步)。如果要使DO通道數值變更為0x5678，使用者可傳送一SYNC物件給CAN-2057C，以觸發前一個步驟所傳輸的RxPDO，如下：

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	0	0	1	0	0	0	0	0	0	0	0	0	--	--	--	--	--	--	--	--

SYNC 生產者 → **SYNC 消費者 (CAN-2057C)**

COB-ID : 0x80

T SYNC 訊息的格式是固定的，如上所示。

每一台裝置的 SYNC 訊息，其 COB-ID 均可依據實際需求透過 SDO 通訊協定做修改，SYNC 的通訊協定為生產者/消費者的架構。

步驟15：當先前傳送的第1組RxPDO被SYNC物件觸發之時，CAN-2057C的DO數值便會立刻改變為0x5678，因此CAN-2053C的DI數值也會跟著改變，此時使用者就會收到CAN-2053C傳來的第1組TxPDO。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	0	1	1	0	0	0	0	0	1	0	0	2	78	56	--	--	--	--	--	--

PDO 消費者 ← **PDO 生產者 (CAN-2053C)**

COB-ID : 0x182
L : 2
PDO-msg : 78 56 -- -- -- -- -- --

步驟 16：把第 1 組 RxPDO 的傳輸型態設回 255。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0	0	1	0	8	2F	00	14	02	FF	00	00	00

SDO 用戶端 → **SDO 伺服端 (CAN-2057C)**

ccs : 1
n : 3
e : 1
s : 1
m : 00 14 02
d : FF 00 00 00

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	0	1	0	4	60	00	14	02	--	--	--	--

SDO 用戶端 ← **SDO 伺服端 (CAN-2057C)**

scs : 3

m : 00 14 02

- 非循環同步 **TxPDO** 的功能展示**(**傳輸型態為 **0)**

步驟 17：設定第 1 組 TxPDO 的傳輸型態為 0。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0	1	0	0	8	2F	00	18	02	00	00	00	00

SDO 用戶端 → **SDO 伺服端 (CAN-2053C)**

ccs : 1
n : 3
e : 1
s : 1
m : 00 18 02
d : 00 00 00 00

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	1	0	0	4	60	00	18	02	--	--	--	--

SDO 用戶端 ← **SDO 伺服端 (CAN-2053C)**

scs : 3
m : 00 18 02

步驟 18：利用第 1 組 RxPDO 將 CAN-2057C 的 DO 值變更為 0x90AB

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	1	0	0	0	0	0	0	0	0	1	0	8	AB	90	00	00	00	00	00	00

PDO 生產者 → **PDO 消費者 (CAN-2057C)**

COB-ID : 0x201
L : 8
PDO-msg : AB 90 00 00 00 00 00 00

步驟19：因為第1組TxPDO的傳輸型態被設定為0，此時即使DI測量到的數值發生變化，CAN-2053C也不會傳送第1組的TxPDO給使用者。此時若CAN-2053C收到SYNC訊後，則第1組TxPDO的傳送就會被觸發。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	0	0	1	0	0	0	0	0	0	0	0	0	---	---	---	---	---	---	---	---

SYNC 生產者 → **SYNC 消費者 (CAN-2053C)**

COB-ID : 0x80

步驟20：在CAN-2053C收到SYNC訊息之後，便會觸發第1組TxPDO的傳送，此時使用者便可以接收到由CAN-2053C傳來的第1組TxPDO。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	0	1	1	0	0	0	0	0	1	0	0	2	90	AB	--	--	--	--	--	--

PDO 消費者 ← **PDO 生產者 (CAN-2053C)**

COB-ID : 0x182
L : 2
PDO-msg : 90 AB -- -- -- -- -- --

步驟 21：再傳送一次 SYNC 訊息給 CAN-2053C。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	0	0	1	0	0	0	0	0	0	0	0	0	---	---	---	---	---	---	---	---

SYNC 生產者 → **SYNC 消費者 (CAN-2053C)**

SYNC COB-ID : 0x80

步驟22：若此時DI的值沒有發生變化，則CAN-2053C即使收到SYNC訊息，也不會觸發第一組TxPDO的傳送。如果此時第1組TxPDO的傳輸型態被設定為1，則不管DI的值有沒有被改變，CAN-2053C在收到SYNC訊息之後，都會觸發第一組TxPDO的傳送。傳輸型態0或1，這兩者最大的差別就在於SYNC訊息對其的觸發行為。

- 循環同步 **TxPDO** 的功能展示**(**傳輸型態為 **3)**

步驟 23：設定第 1 組 TxPDO 的傳輸型態為 3。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0	1	0	0	8	2F	00	18	02	03	00	00	00

SDO 用戶端 → **SDO 伺服端 (CAN-2053C)**

ccs : 1
n : 3
e : 1
s : 1
m : 00 18 02
d : 03 00 00 00

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	1	0	0	4	60	00	18	02	--	--	--	--

SDO 用戶端 ← **SDO 伺服端 (CAN-2053C)**

scs : 3
m : 00 18 02

步驟 24：利用第 1 組 RxPDO 將 CAN-2057C 的 DO 值變更為 0xCDEF。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	1	0	0	0	0	0	0	0	0	1	0	8	EF	CD	00	00	00	00	00	00

PDO 生產者 → **PDO 消費者 (CAN-2057C)**

COB-ID : 0x201
L : 8
PDO-msg : EF CD 00 00 00 00 00 00

步驟25：於此傳送3個SYNC訊息給CAN-2053C因為目前第1組TxPDO的傳輸型態被設定為3，所以CAN-2053C在收到3個SYNC訊息，便會觸發第1組TxPDO的傳送。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	0	0	1	0	0	0	0	0	0	0	0	0	---	---	---	---	---	---	---	---

SYNC 生產者 → **SYNC 消費者 (CAN-2053C)**

COB-ID : 0x80

步驟26：CAN-2053C在收到3個SYNC訊息之後，便會觸發第1組TxPDO的傳送，此時使用者便可以收到由CAN-2053C所傳來的第1組TxPDO。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	0	1	1	0	0	0	0	0	1	0	0	2	EF	CD	---	---	---	---	---	---

PDO 消費者 ← **PDO 生產者 (CAN-2053C)**

COB-ID : 0x182
L : 2
PDO-msg : EF CD -- -- -- -- -- --

- 同步、唯遠端傳輸要求 **TxPDO** 的功能展示**(**傳輸型態 **252)**

步驟27：設定第1組TxPDO的傳輸型態為252，若TxPDO被設定為此一傳輸型態，則在收到SYNC訊息時，才會更新PDO的內容，並且在收到此TxPDO的RTR訊息之後，才會觸發此TxPDO的傳送。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0	1	0	0	8	2F	00	18	02	FC	00	00	00

SDO 用戶端 → **SDO 伺服端 (CAN-2053C)**

ccs : 1
n : 3
e : 1
s : 1
m : 00 18 02
d : FC 00 00 00

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	1	0	0	4	60	00	18	02	--	--	--	--

SDO 用戶端 ← **SDO 伺服端 (CAN-2053C)**

scs : 3
m : 00 18 02

步驟 28：利用第 1 組 RxPDO 將 CAN-2057C 的 DO 值變更為 0x1234。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	1	0	0	0	0	0	0	0	0	1	0	8	34	12	00	00	00	00	00	00

PDO 生產者 → **PDO 消費者 (CAN-2057C)**

COB-ID : 0x201

L : 8

PDO-msg : 34 12 00 00 00 00 00 00

步驟29：此處傳送第1組TxPDO的RTR訊息給CAN-2053C，要求其回傳第一組TxPDO。因為第1組TxPDO的傳輸型態被設定為252，所以必須等到CAN-2053C收到RTR訊息之後，第1組TxPDO的傳送才會被觸發。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	0	1	1	0	0	0	0	0	1	0	1	0	---	---	---	---	---	---	---	---

PDO 消費者 → **PDO 生產者 (CAN-2053C)**

COB-ID : 0x182

步驟30：使用者在收到CAN-2053C傳來的第1組TxPDO訊息之後，可以跟CAN-2053C面板上的LED燈號做比較，於此處會發現PDO內記錄的為更新前的DO數值。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	0	1	1	0	0	0	0	0	1	0	0	2	34	12	00	00	00	00	00	00

PDO 消費者 ← **PDO 生產者 (CAN-2053C)**

COB-ID : 0x182
L : 2
PDO-msg : 34 12 00 00 00 00 00 00

步驟 31：傳送 SYNC 訊息給 CAN-2053C，以更新第一組 TxPDO 內記錄的值。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	0	0	1	0	0	0	0	0	0	0	0	0	---	---	---	---	---	---	---	---

SYNC 生產者 → **SYNC 消費者 (CAN-2053C)**

COB-ID : 0x80

步驟32：此處再傳送一次第1組TxPDO的RTR訊息給CAN-2053C，要求其回傳第一組TxPDO。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	0	1	1	0	0	0	0	0	1	0	1	0	---	---	---	---	---	---	---	---

PDO 消費者 → **PDO 生產者 (CAN-2053C)**

COB-ID : 0x182

步驟33：此時使用者在收到CAN-2053C傳來的第1組TxPDO訊息之後，就會發現PDO內記錄的DO數值和CAN-2053C面板上的LED燈號相符合。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	0	1	1	0	0	0	0	0	1	0	0	2	34	12	---	---	---	---	---	---

PDO 消費者 ← **PDO 生產者 (CAN-2053C)**

COB-ID : 0x182

L : 2

PDO-msg : 34 12 -- -- -- -- -- --

- 非同步、唯遠端傳輸要求 **TxPDO** 的功能展示**(**傳輸型態 **253)**

步驟 34：設定第 1 組 TxPDO 的傳輸型態為 253。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0	1	0	0	8	2F	00	18	02	FD	00	00	00

SDO 用戶端 → **SDO 伺服端 (CAN-2053C)**

ccs : 1
n : 3
e : 1
s : 1
m : 00 18 02
d : FD 00 00 00

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	1	0	0	4	60	00	18	02	--	--	--	--

SDO 用戶端 ← **SDO 伺服端 (CAN-2053C)**

scs : 3
m : 00 18 02

步驟 35：利用第 1 組 RxPDO 將 CAN-2057C 的 DO 值變更為 0x5678。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	1	0	0	0	0	0	0	0	0	1	0	8	78	56	00	00	00	00	00	00

PDO 生產者 → **PDO 消費者 (CAN-2057C)**

COB-ID : 0x201
L : 8
PDO-msg : 78 56 00 00 00 00 00 00

步驟36：因為第1組TxPDO的傳輸型態被設定為253，所以CAN-2053C僅在收到此TxPDO的RTR訊息之後，才會更新TxPDO的內容與對外傳送。於下，先傳送第1組TxPDO的RTR訊息給CAN-2053C，CAN-2053C在收到RTR訊息之後會回傳第一組TxPDO，內容為CAN-2053C最新的DI數值。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	0	1	1	0	0	0	0	0	1	0	1	0	---	---	---	---	---	---	---	---

PDO 消費者 ⟵ **PDO 生產者 (CAN-2053C)**

COB-ID : 0x182

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	0	1	1	0	0	0	0	0	1	0	0	2	78	56	---	---	---	---	---	---

PDO 消費者 ⟵ **PDO 生產者 (CAN-2053C)**

COB-ID : 0x182

L : 2

PDO-msg : 78 56 -- -- -- -- -- --

步驟 37：把第 1 組 TxPDO 的傳輸型態設回 255。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0	1	0	0	8	2F	00	18	02	FF	00	00	00

SDO 用戶端 ⟶ **SDO 伺服端 (CAN-2053C)**

ccs : 1

n : 3

e : 1

s : 1

m : 00 18 02

d : FF 00 00 00

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	1	0	0	4	60	00	18	02	--	--	--	--

SDO 用戶端 ◄ **SDO 伺服端 (CAN-2053C)**

scs : 3

m : 00 18 02

- 對 **DI/AI/DO/AO** 通道使用動態 **PDO** 映射

步驟38：底下將以CAN-2017C為例示範使用者要如何建立新的TxPDO，於下將建立第5組的TxPDO(使用者可自行選用空餘的TxPDO)。首先必須決定TxPDO的COB-ID，即設定此TxPDO的通訊參數(主索引0x1804)的COB-ID項目(子索引0x01)，另外需確認此項目的第31個bit為1，才可以更改此項目，若此bit為0，則無法進行更改，必須將此項目的第31個bit設定為1，才可以進行接下來的修改動作，在修改COB-ID項目的時後，需再把第31個bit設定為0。

使用者在設定COB-ID時，必須確定所使用的COB-ID不是CANopen規範保留的COB-ID，也沒有被裝置上的其他通訊物件使用，或著和網路中其他裝置的通訊物件衝突。因為第5組TxPDO的通訊參數的COB-ID項目，其第31 bit預設為1，所以可以直接修改通訊參數的COB-ID項目，於此將其設定為0x333。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	1	0	0	0	8	23	05	18	01	33	03	00	00

SDO 用戶端 → **SDO 伺服端 (CAN-2017C)**

ccs : 1
n : 0
e : 1
s : 1
m : 05 18 01
d : 33 03 00 00

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	1	0	0	0	4	60	05	18	01	---	---	---	---

SDO 用戶端 ← **SDO 伺服端 (CAN-2017C)**

scs : 3
m : 05 18 01

步驟39：於此處設定第5組TxPDO的映射參數(主索引0x1A04)。在開始設定PDO的映射參數之前，首先必須要確認物件字典中主索引為0x1A04且子索引為0x00的項目之數值為0，如果此數值不為0**(*Note:請參考第104頁)**，則對映射參數所做的設定便會失敗。於此處，由於主索引為0x1A05的物件並沒有被使用過，其子索引0x00項目的數值預設便為0，所以可以直接對此PDO的映射參數作設定。於下將設定CAN-2017C的AI2和AI5到第5組TxPDO映射參數的子索引0x01和0x02。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	1	0	0	0	8	23	05	1A	01	10	03	01	64

SDO 用戶端 → **SDO 伺服端 (CAN-2017C)**

ccs : 1
n : 0
e : 1
s : 1
m : 05 1A 01
d : 10 03 01 64

“10 03 01 64” 即代表 PDO 映射參數(0x1A05)的子索引 0x01 映射到主索引為 0x6401 且子索引為 0x03 的項目，其長度為 16bits。使用者可以參照本節一開始所提過的“標準化裝置描述文件區域”也就是說第 5 組 TxPDO 訊息會映射到 CAN-2017C 的 AI2。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	1	0	0	0	4	60	05	1A	01	---	---	---	---

SDO 用戶端 ← **SDO 伺服端 (CAN-2017C)**

scs : 3
m : 05 1A 01

步驟40：於下設定CAN-2017C的AI5到第5組TxPDO映射參數(主索引0x1A05)的子索引0x02。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	1	0	0	0	8	23	05	1A	02	16	06	01	64

SDO 用戶端 → **SDO 伺服端 (CAN-2017C)**

ccs : 1
n : 0
e : 1
s : 1
m : 05 1A 02
d : 10 06 01 64

“10 06 01 64”即代表 PDO 映射參數(0x1A05)的子索引 0x02 映射到主索引為 0x6401 且子索引為 0x06 的項目，其長度為 16bits。使用者可以參照本節一開始所提過的“標準化裝置描述文件區域”也就是說第 5 組 TxPDO 訊息會映射到 CAN-2017C 的 AI5。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	1	0	0	0	4	60	05	1A	02	---	---	---	---

SDO 用戶端 ← **SDO 伺服端 (CAN-2017C)**

scs : 3
m : 05 1A 02

步驟41：設定PDO映射參數的最後一個步驟就是在其子索引0x00的地方填入此PDO內含的映射個數，於此處在主索引0x1A05且子索引為0x00的地方填入2。2代表第5組的TxPDO內含2個映射，分別指向主索引為0x6401且子索引為0x03處、主索引為0x6401且子索引為0x06處。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	1	0	0	0	8	2F	05	1A	00	02	00	00	00

SDO 用戶端 → **SDO 伺服端 (CAN-2017C)**

ccs : 1
n : 3
e : 1
s : 1
m : 05 1A 00
d : 02 00 00 00

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	1	0	0	0	4	60	05	1A	00	---	---	---	---

SDO 用戶端 ← **SDO 伺服端 (CAN-2017C)**

scs : 3
m : 05 1A 00

步驟42：底下將以CAN-2024C為例示範使用者要如何建立新的RxPDO通訊物件，於此處使用第5組的RxPDO(使用者可自行選用空餘的RxPDO)設定COB-ID為0x222，映射到CAN-2024C的AO2和AO5。RxPDO的設定方式和TxPDO相同，於此處便不再贅述。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0	1	1	0	8	23	05	14	01	22	02	00	00

SDO 用戶端 → **SDO 伺服端 (CAN-2024C)**

ccs : 1
n : 0
e : 1
s : 1
m : 05 14 01
d : 22 02 00 00

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	1	1	0	4	60	05	14	01	---	---	---	---

SDO 用戶端 ← **SDO 伺服端 (CAN-2024C)**

scs : 3
m : 05 14 01

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0	1	1	0	8	23	05	16	01	16	03	11	64

SDO 用戶端 → **SDO 伺服端 (CAN-2024C)**

ccs : 1
n : 0
e : 1
s : 1
m : 05 16 01
d : 10 03 11 64

"10 03 11 64"即代表PDO映射參數(0x1605)的子索引0x01映射到主索引為0x6411且子索引為0x03的項目，其長度為16 bits。使用者可以從上面的標準物件映射表中了解到，這是CAN-2024C AO2的值。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	1	1	0	4	60	05	16	01	---	---	---	---

SDO 用戶端 ← **SDO 伺服端 (CAN-2024C)**

scs : 3
m : 05 16 01

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0	1	1	0	8	23	05	16	02	16	06	11	64

SDO 用戶端 → **SDO 伺服端 (CAN-2024C)**

ccs : 1
n : 0
e : 1
s : 1
m : 05 16 02
d : 10 06 11 64

“10 06 11 64”即代表 PDO 映射參數(0x1605)的子索引 0x02 映射到主索引為 0x6411 且子索引為 0x06 的項目，其長度為 16 bits。使用者可以從上面的標準物件映射表中了解到，這是 CAN-2024C AO5 的值。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	1	1	0	4	60	05	16	02	---	---	---	---

SDO 用戶端 ← **SDO 伺服端 (CAN-2024C)**

scs : 3
m : 05 16 02

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0	1	1	0	8	2F	05	16	00	02	00	00	00

SDO 用戶端 → **SDO 伺服端 (CAN-2024C)**

ccs : 1
n : 3
e : 1
s : 1
m : 05 16 00
d : 02 00 00 00

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	1	1	0	4	60	05	16	00	---	---	---	---

SDO 用戶端 ← **SDO 伺服端 (CAN-2024C)**

scs : 3
m : 05 16 00

步驟43：透過底下PDO訊息的傳送，分別將CAN-2024C的AO2和AO5設定為0V和5V。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	1	0	0	0	1	0	0	0	1	0	0	4	00	00	FF	3F	---	---	---	---

PDO 消費者 → **PDO 生產者 (CAN-2024C)**

COB-ID : 0x222

PDO-msg : 00 00 FF 3F -- -- -- --

這個PDO訊息前2 bytes為0x0000是CAN-2024C的AO2資料。第3和第4個byte為0x3FFF是CAN-2024C的AO5資料。總共有4 bytes的有效資料。

步驟44：使用者可以送出一個COB-ID為0x333的TxPDO來得到AI值。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	1	1	0	0	1	1	0	0	1	1	1	0	---	---	---	---	---	---	---	---

PDO 消費者 → **PDO 生產者 (CAN-2017C)**

COB-ID : 0x333

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	0	1	1	0	0	0	0	1	0	1	0	4	FF	FF	00	40	---	---	---	---

PDO 消費者 ← **PDO 生產者 (CAN-2017C)**

COB-ID : 0x333

L : 4

PDO-msg : FF FF 00 40 -- -- -- --

這個PDO訊息前2 bytes為0xFFFF是CAN-2017C的AI2資料。第3和第4個byte為0x4000是CAN-2017C的AI5資料。換算之後AI2的值為-0.001V，而AI5的值為5.000V。

Note: 如果物件 **0x1A00~0x1A09 / 0x1600~0x1609** 的子索引 **0** 不為 **0**，則必須先將子索引 **0** 變更為 **0**，然後才能修改要對應的物件。以下是一個範例：

(1) 將物件 0x1A01 的子索引 0 設定為 0。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	1	0	0	0	8	2F	01	1A	00	00	00	00	00

SDO 用戶端 → **SDO 伺服端 (CAN-2017C)**

ccs : 1
n : 3
e : 1
s : 1
m : 01 1A 00
d : 00 00 00 00

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	1	0	0	0	4	60	05	1A	00	---	---	---	---

SDO 用戶端 ← **SDO 伺服端 (CAN-2017C)**

scs : 3
m : 01 1A 00

（2）寫入要映射到物件 0x1A01 的對象

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	1	0	0	0	8	23	01	1A	01	10	03	01	64

SDO 用戶端 → **SDO 伺服端 (CAN-2017C)**

ccs : 1
n : 0
e : 1
s : 1
m : 01 1A 01
d : 10 03 01 64

“10 03 01 64” 即代表 PDO 映射參數(0x1A05)的子索引 0x01 映射到主索引為 0x6401 且子索引為 0x03 的項目，其長度為 16bits。使用者可以參照本節一開始所提過的“標準化裝置描述文件區域”也就是說第 5 組 TxPDO 訊息會映射到 CAN-2017C 的 AI2。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	1	0	0	0	4	60	01	1A	01	---	---	---	---

SDO 用戶端 ← **SDO 伺服端 (CAN-2017C)**

scs : 3
m : 01 1A 01

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	1	0	0	0	8	23	01	1A	02	16	06	01	64

SDO 用戶端 ⟶ **SDO 伺服端 (CAN-2017C)**

ccs : 1
n : 0
e : 1
s : 1
m : 01 1A 02
d : 10 06 01 64

“10 06 01 64”即代表 PDO 映射參數(0x1A05)的子索引 0x02 映射到主索引為 0x6401 且子索引為 0x06 的項目，其長度為 16bits。使用者可以參照本節一開始所提過的“標準化裝置描述文件區域”也就是說第 5 組 TxPDO 訊息會映射到 CAN-2017C 的 AI5。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	1	0	0	0	4	60	01	1A	02	---	---	---	---

SDO 用戶端 ⟵ **SDO 伺服端 (CAN-2017C)**

scs : 3
m : 01 1A 02

（3）將子索引的數量寫入物件 0x1A01 的子索引 0。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	1	0	0	0	8	2F	01	1A	00	02	00	00	00

SDO 用戶端 → **SDO 伺服端 (CAN-2017C)**

ccs : 1
n : 3
e : 1
s : 1
m : 01 1A 00
d : 02 00 00 00

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	1	0	0	0	4	60	01	1A	00	---	---	---	---

SDO 伺服端 (CAN-2017C) → **SDO 用戶端**

scs : 3
m : 01 1A 00

5.3 EMCY 通訊集

5.3.1 EMCY COB-ID 參數

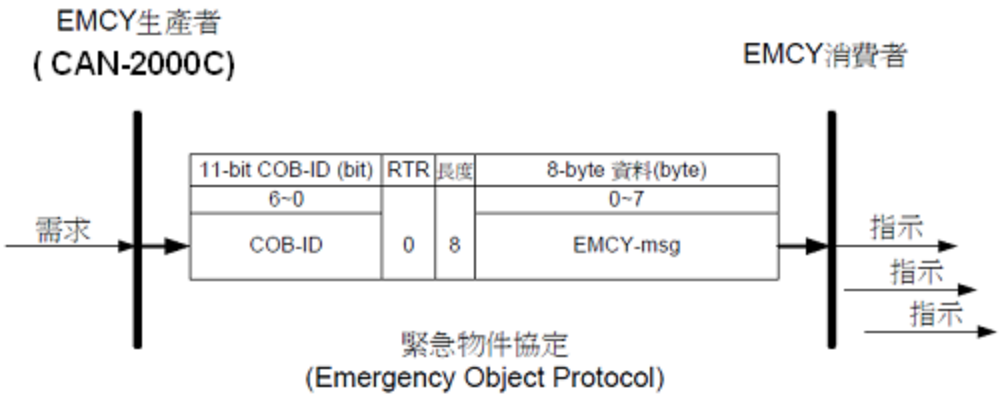
EMCY訊息的COB-ID和PDO訊息的COB-ID十分類似，使用者可以直接使用裝置開機之後的預設值，或著透過SDO訊息來對其作更改。EMCY訊息的COB-ID儲存在物件字典中主索引為0x1014的物件內，其資料格式如下表所示：

Bit 編號	值	代表的意義
31 (MSB)	0	EMCY 存在 (EMCY 為有效的，valid)
	1	EMCY 不存在 (EMCY 是無效的，invalid)
30	0	被保留 (此 bit 的值永為 0)
29	0	11-bit ID (CAN 2.0A)
	1	29-bit ID (CAN 2.0B)
28-11	0	若 bit 29=0，此欄位的數值便為 0
	x	若bit 29=1：則此欄位就是29bits COB-ID 內的第 28~11 bits
10-0 (LSB)	x	COB-ID 內的第 10~0 bits

在使用 EMCY訊息之前，必須先確認0x1014這個物件的bit 31為0，也就是這個裝置上的EMCY訊息是有效的。

5.3.2 EMCY 通訊協定

當裝置內產生某些特定的內部錯誤，亦即發生EMCY事件，此時便會觸發EMCY訊息的傳送。在物件字典內主索引0x1003的物件中，會記錄裝置上曾經發生過的EMCY事件，使用者可以透過檢視這個物件來了解裝置發生錯誤的記錄。CAN-2000C模組在主索引0x1003的物件內，最多可儲存5筆記錄，由子索引0x01到子索引0x05，其儲存的方式為先進先出，最近的EMCY事件會儲存在子索引0x01。底下將介紹傳送EMCY訊息所使用的通訊協定：



COB-ID ： 使用者可以自行定義EMCY 訊息所使用的COB-ID，定義的方式和PDO 相同。EMCY 訊息預設的COB-ID 為4-bit 的功能碼“0001”加上7-bit 的節點ID。

EMCY-msg ： 內含緊急物件資料(emergency object data)，此欄位所記錄的資料可以用來識別裝置上發生錯誤的種類。

緊急物件資料的資料格式如下表所示：

Byte	0	1	2	3	4	5	6	7
內容	緊急錯誤碼 (Emergency Error Code)		錯誤暫存器 (Error register)	製造商特定錯誤區域 (Manufacturer specific Error Field)				

錯誤暫存器長度 8-bit，其內每一 bit 所代表的意義如下表所示。CAN-2000C 模組僅支援 bit0、bit4 和 bit7。

Bit 編號	代表的意義
0	一般性的錯誤
1	電流錯誤
2	電壓錯誤
3	溫度錯誤
4	通訊錯誤(過載，或著裝置正處於錯誤狀態)
5	於裝置描述文件內定義
6	保留(數值永為 0)
7	製造商自定義

針對CAN-2000C模組，緊急物件資料內不同的內容所代表的意義如下表所示：

緊急錯誤碼		錯誤暫存器	製造商特定錯誤區域			代表的意義
			前 2 Byte		後 3 Byte	
00	00	00	00	00	00 00 00	錯誤重置，或著沒有錯誤發生
10	00	81	01	00	00 00 00	CAN 控制器發生錯誤
50	00	81	02	00	00 00 00	EEPROM 存取錯誤
81	10	11	04	00	00 00 00	軟體的 Rx 緩衝區過載
81	10	11	05	00	00 00 00	軟體的 Tx 緩衝區過載
81	10	11	06	00	00 00 00	CAN 控制器過載
81	30	11	07	00	00 00 00	生存守衛事件錯誤
81	40	11	08	00	00 00 00	裝置已由 bus off 狀態下復原
82	10	11	09	00	00 00 00	PDO 資料長度錯誤
FF	00	80	0A	00	00 00 00	裝置需要重新啟動節點或著重新啟動通訊

裝置在產生EMCY訊息之後，會把EMCY訊息之中的緊急錯誤碼和製造商特定錯誤區域的前2 bytes 存到主索引為0x1003且子索引為0x01的項目之中，如下：

Byte：	3	2	1	0
	製造商特定錯誤區域的前 2 bytes		緊急錯誤碼	

另外在主索引0x1001 的物件內，則會記錄錯誤暫存器的內容。使用者可以查閱物件字典內的這兩個地方，以檢視裝置上發生錯誤的記錄，進一步了解CAN-2000C模組為何發出緊急訊息。

EMCY 通訊範例

底下範例所使用的硬體架構與PDO 通訊範例相同，使用者可參照5.2.3 節。

步驟1：為了產生EMCY事件，此處傳送一PDO訊息給CAN-2057C，設定此PDO的COB-ID為0x201，資料長度為0x01。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	1	0	0	0	0	0	0	0	0	1	0	1	00	---	---	---	---	---	---	---

PDO 消費者 ← **PDO 生產者 (CAN-2057C)**

COB-ID : 0x201
L : 1
PDO-msg : 00

步驟2：CAN-2057C在收到上一個步驟的PDO 訊息之後，根據這個PDO的COB-ID，會判斷這個PDO為第1 組RxPDO，且會因為其內所記錄的資料長度和PDO映射參數內所記錄的不吻合，所以會對外回應一緊急訊息，如下。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	0	0	1	0	0	0	0	0	0	1	0	8	10	82	11	09	00	00	00	00

EMCY 消費者 → **EMCY 生產者 (CAN-2057C)**

COB-ID : 0x81
EMCY-msg : 10 82 11 09 00 00 00 00

前2 bytes為緊急錯誤碼。第3個byte為錯誤暫存器，數值“11” 代表CAN-2057C有通訊錯誤或一般性的錯誤。後5 bytes為製造商特定錯誤區域。這8 bytes的緊急物件資料所代表的意義為裝置所接收到的PDO資料長度與PDO映射參數內所記錄的資料長度不吻合。

步驟3：於此處，使用者可以傳送以下的SDO訊息，以讀取裝置的物件字典內主索引為0x1003且子索引為0x01的項目，使用者可以獲得儲存在這個項目內的緊急錯誤碼和製造商特定錯誤區域的前2 bytes。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0	0	1	0	8	40	03	10	01	00	00	00	00

SDO 用戶端 → **SDO 伺服端 (CAN-2057C)**

ccs : 2
m 03 10 01

步驟4：CAN-2057C會回應以下訊息，內含主索引為0x1003且子索引為0x01的項目。其內容會和最近一次所收到的EMCY訊息相吻合。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	0	1	0	8	43	03	10	01	10	82	09	00

SDO 用戶端 ← **SDO 伺服端 (CAN-2057C)**

scs : 2
n : 0
e : 1
s : 1
m : 03 10 01
d : 10 82 09 00

步驟5：於此處，傳送以下的SDO訊息，讀取裝置的物件字典內主索引為0x1001的物件，以檢查此處錯誤暫存器的數值和最近一次收到的EMCY訊息是否吻合。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0	0	1	0	8	40	01	10	00	00	00	00	00

SDO 用戶端 ⟶ **SDO 伺服端 (CAN-2057C)**

ccs : 2
m 01 10 00

步驟6：CAN-2057C所回傳的SDO訊息中，錯誤暫存器的數值為0x11，代表通訊錯誤或一般性的錯誤，與最近一次所收到的EMCY訊息相吻合。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	0	1	0	5	4F	01	10	00	11	---	---	---

SDO 用戶端 ⟵ **SDO 伺服端 (CAN-2057C)**

scs : 2
n : 3
e : 1
s : 1
m : 01 10 00
d : 11 -- -- --

步驟7：於此步驟中，傳送一COB-ID為0x201的RxPDO訊息給CAN-2057C，並於此訊息中註記資料長度為0x02。CAN-2057C在收到這個訊息之後，會發出一錯誤重置的EMCY訊息，代表先前的錯誤已被排除。

另外因為TxPDO的值和原先相同所以DO的數值沒有改變，此時就不會觸發CAN-2057C傳送第1 組的TxPDO。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	1	0	0	0	0	0	0	0	0	1	0	2	00	00	---	---	---	---	---	---

PDO 消費者 ⟵ **PDO 生產者 (CAN-2057C)**

COB-ID : 0x201

L : 2

PDO-msg : 00 00 -- -- -- -- -- --

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	0	0	1	0	0	0	0	0	0	1	0	8	00	00	00	00	00	00	00	00

NMT 主端 ⟵ **NMT 僕端 (CAN-2057C)**

EMCY-msg : 00 00 00 00 00 00 00 00

錯誤重置 EMCY 訊息的資料欄位為“00 00 00 00 00 00 00 00” 即代表 CAN-2057C 目前沒有任何的錯誤。

步驟9:於此處,傳送以下的SDO訊息,以讀取裝置的物件字典內主索引為0x1003且子索引為0x0 的項目，使用者可以檢視儲存在這個項目內的緊急錯誤碼和製造商特定錯誤區域的前2 bytes，其內容會和最近一次所收到的EMCY訊息相吻合。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0	0	1	0	8	40	03	10	01	00	00	00	00

SDO 用戶端 → **SDO 伺服端 (CAN-2057C)**

ccs : 2
m 03 10 01

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	0	1	0	8	43	03	10	01	---	---	---	---

SDO 用戶端 ← **SDO 伺服端 (CAN-2057C)**

ccs : 1
n : 0
e : 1
s : 1
m : 03 10 01
d : -- -- -- --

步驟10：於此處，傳送以下的SDO訊息，以讀取裝置的物件字典內主索引為0x1003且子索引為0x02的項目，使用者可以檢視儲存在這個項目內的緊急錯誤碼和製造商特定錯誤區域的前2 bytes，其內容會和上上一次所收到的EMCY訊息相吻合。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0	0	1	0	8	40	03	10	02	00	00	00	00

SDO 用戶端 → **SDO 伺服端 (CAN-2057C)**

ccs : 2
m : 03 10 02

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	0	1	0	8	43	03	10	02	10	82	09	00

SDO 用戶端 ← **SDO 伺服端 (CAN-2057C)**

ccs : 1
n : 0
e : 1
s : 1
m : 03 10 02
d : 10 82 09 00

步驟11：於此處，傳送以下的SDO訊息，讀取裝置的物件字典內主索引為0x1001的物件，使用者會發現目前錯誤暫存器的數值為0x00，和最近一次收到的EMCY訊息相吻合。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0	0	1	0	8	40	01	10	00	00	00	00	00

SDO 用戶端 → **SDO 伺服端 (CAN-2057C)**

ccs : 2
m 01 10 00

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	0	1	0	8	4F	01	10	00	00	---	---	---

SDO 用戶端 ← **SDO 伺服端 (CAN-2057C)**

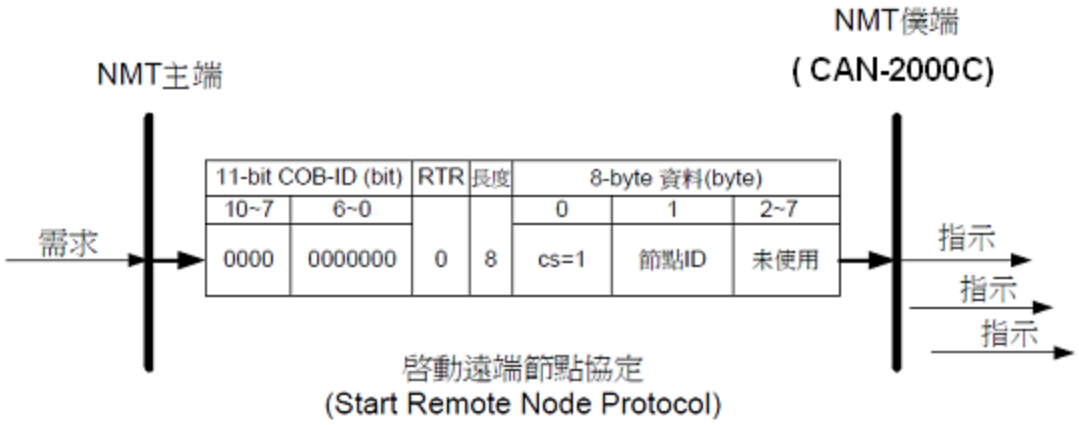
ccs : 1
n 2
e 1
s 1
m : 01 10 00
d : 00 -- -- --

5.4 NMT 通訊集

5.4.1 模組控制協定

NMT主端可以利用模組控制協定(Module Control Protocol)來改變NMT 僕端的NMT狀態，底下將詳細介紹CAN-2000C模組控制協定的細節，

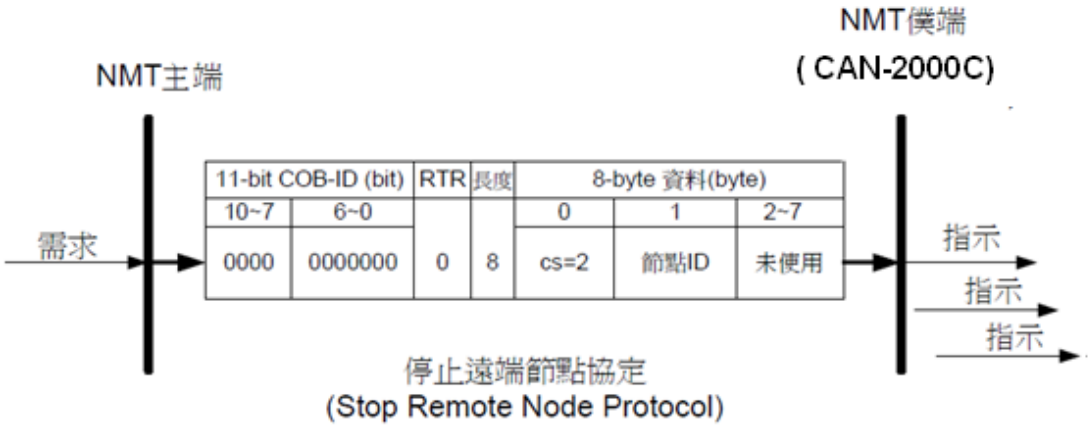
<u>啟動遠端節點協定(Start Remote Node Protocol)</u>



cs : NMT 命令識別符
1：啟動(start)

節點 ID : NMT 僕端裝置的節點 ID

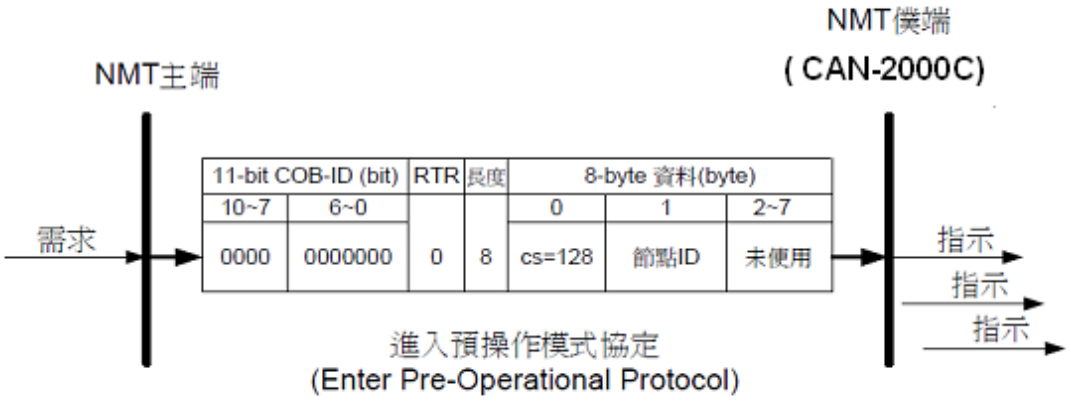
停止遠端節點協定(Stop Remote Node Protocol)



cs : NMT 命令識別符
2：停止(stop)

節點 ID : NMT 僕端裝置的節點 ID。

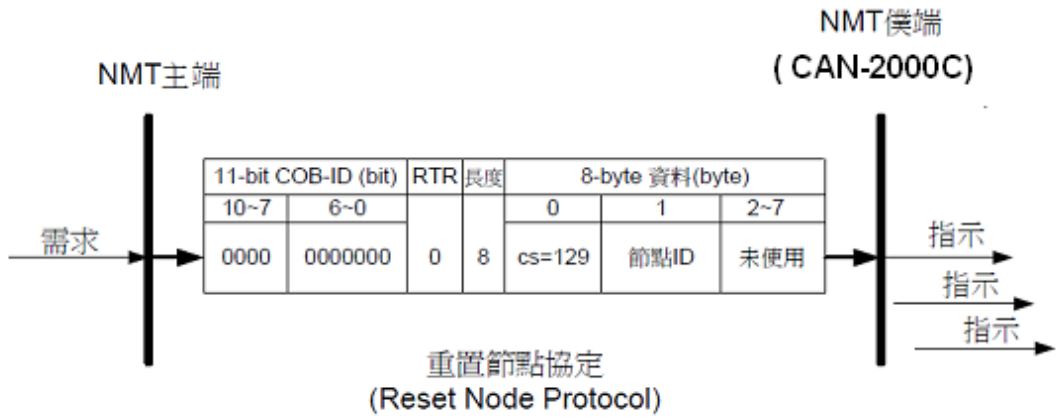
進入預操作狀態協定(Enter Pre-Operational Protocol)



cs : NMT 命令識別符
128: 進入預操作(PRE-OPERATIONAL)狀態

節點 ID : NMT 僕端裝置的節點 ID

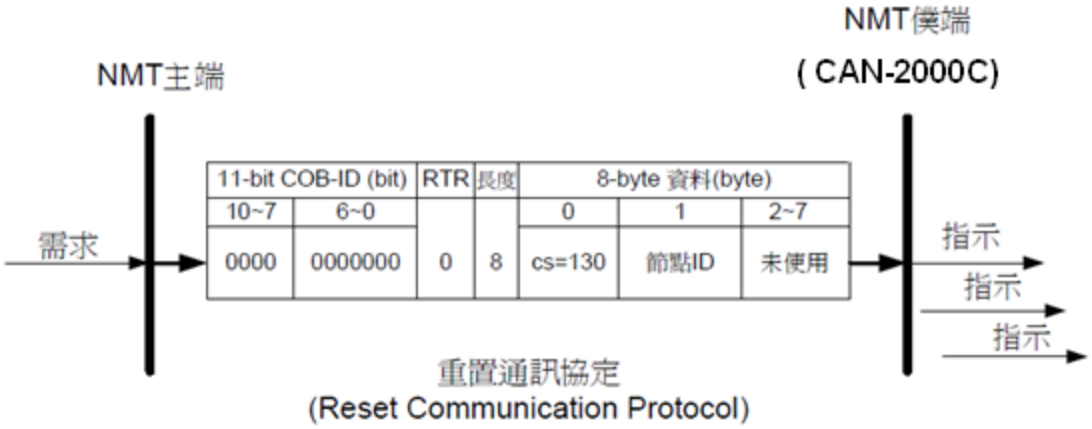
重置節點協定(Reset Node Protocol)



cs : NMT 命令識別符
129：重置(reset)節點

節點 ID : NMT 僕端裝置的節點 ID。

重置通訊協定(Reset Communication Protocol)



cs : NMT 命令識別符
130：重置通訊(Reset_Communication)

節點 ID : NMT 僕端裝置的節點 ID。

模組控制協定範例 (Module Control Protocol Example)

下面以 CAN-2000C 模組站點 ID 是 5 為例:

步驟 1：關閉 CAN-2000C。

步驟2：啟動CAN-2000C。若無異常，CAN-2000C在結束初始化(initialization)後，會自動進入預操作(Pre_Operational)狀態。此時使用者可以看到CAN-2000C面板上面的運行指示燈約每秒閃爍兩次。

步驟3:傳送底下的訊息給CAN-2000C，透過NMT模組控制協定要求CAN-2000C進入操作狀態(operational state)。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0	0	0	0	8	01	05	00	00	00	00	00	00

NMT 主端 → **NMT 僕端 (CAN-2000C)**

cs : 1

Node ID : 5

5.4.2 錯誤控制協定

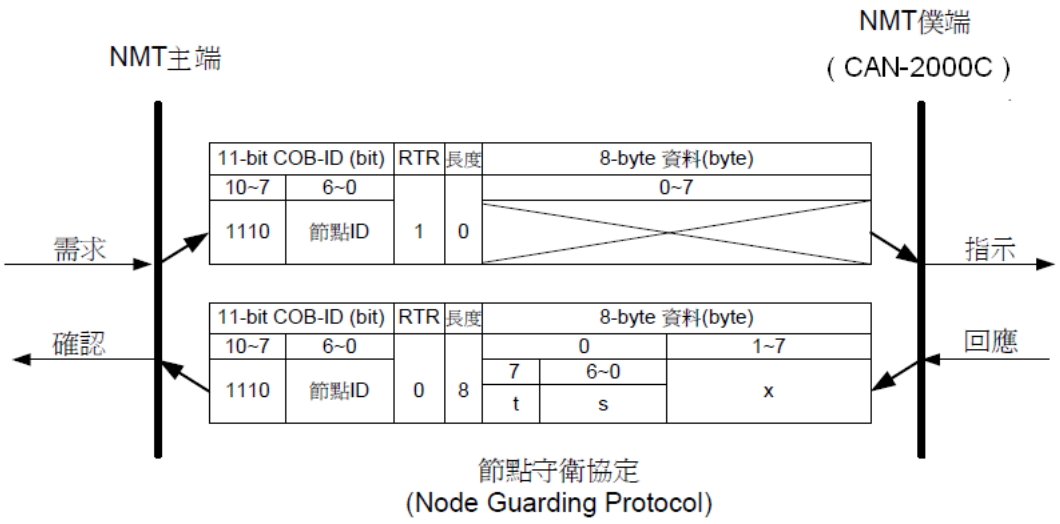
透過錯誤控制協定，我們可以檢查網路當中的CANopen裝置是否存活(運作是否正常)。在物件字典之中，主索引為0x100C的物件記錄了節點守衛時間(node guard time)，主索引為0x100D的物件記錄了生存時間係數(life time factor)。而節點生存時間(node life time)為節點守衛時間乘上生存時間係數。

錯誤控制協定是用來確認 CANopen 裝置是否存活的一種機制。分別有節點守衛協定跟心跳事件產生協定兩種方式。在物件字典之中，與節點守衛協定有關的是主索引為 0x100C 和 0x100D 的物件。而與心跳產生事件協定有關的是主索引為 0x1017 的物件。

註: CAN-2000C 模組不允許同時使用節點守衛協定和心跳產生事件協定。

節點守衛協定 (Node Guarding Protocol)

CAN-2000C模組的節點守衛計時器會在接收到第一個守衛要求訊息後啟動。節點守衛協定的細節如下所示：



t : 交替位元(toggle bit)
對CANopen的NMT僕端而言，在進行節點守衛協定的時候，每一次回覆訊息的交替位元均需與其上一個回覆訊息的交替位元不同。而在節點守衛協定開始進行時，NMT僕端第1次回覆的訊息，其交替位元必須設定為0。

s : NMT 僕端的狀態
4：停止(STOPPED)狀態
5：操作(OPERATIONAL)狀態
127：預操作(PRE-OPERATIONAL)狀態

節點守衛協定範例 (Node Guarding Protocol Example)

底下以一個節點 ID 被設為 1 的 CAN-2000C 模組為範例。步驟如下:

步驟1 ：移除CAN-2000C的電源，再重新把CAN-2000C接上電源。這時候CAN-2000C會自動進入預操作狀態。

步驟2：此處透過初始SDO下載協定，把節點守衛時間設定為250。節點守衛時間位於物件字典中主索引為0x100C之處。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0	0	1	0	8	2B	0C	10	00	FA	00	00	00

SDO 用戶端 ⟶ **SDO 伺服端 (CAN-2000C)**

ccs : 1
n : 2
e : 1
s : 1
m : 0C 10 00
d : FA 00 00 00

步驟 3 : 如果沒有錯誤的話，CAN-2000 會回覆以下訊息以結束初始 SDO 下載。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	0	1	0	4	60	0C	10	00	---	---	---	---

SDO 用戶端 ⟵ **SDO 伺服端 (CAN-2000C)**

scs : 3
m : 0C 10 00

步驟4：此處透過初始SDO下載協定，把生存時間係數設定為4。生存時間係數位於物件字典中主索引為0x100D之處。

.

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0	0	1	0	8	2F	0D	10	00	04	00	00	00

SDO 用戶端 ⟶ **SDO 伺服端 (CAN-2000C)**

ccs : 1
n : 3
e : 1
s : 1
m : 0D 10 00
d : 04 00 00 00

如果沒有錯誤的話，CAN-2000C 會回覆以下訊息以結束初始 SDO 下載。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	0	1	0	4	60	0D	10	00	---	---	---	---

SDO 用戶端 ⟵ **SDO 伺服端 (CAN-2000C)**

scs : 3
m : 0D 10 00

傳送以下的守衛要求訊息以開始節點守衛協定。這時候的生存時間為1秒，也就是1000毫秒(節點守衛時間 × 生存時間係數 = 250 × 4 = 1000)。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	1	0	0	0	0	0	0	0	1	1	0	---	---	---	---	---	---	---	---

NMT 主端 ⟶ **NMT 僕端 r (CAN-2000C)**

COB-ID : 0x701

步驟5：接著，使用者可以接收到如下訊息，其內記錄了CAN-2000C模組所回報的NMT 狀態。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	1	0	0	0	0	0	0	0	1	0	1	7F	---	---	---	---	---	---	---

NMT 主端 ← **NMT 僕端 (CAN-2000C)**

COB-ID : 0x701

t : 1

s 7F

數值 7F 代表 CAN-2000C 模組目前的 NMT 狀態為預操作狀態。

步驟6：因為生存時間現在被設定為1 秒，所以傳送守衛要求訊息給CAN-2000C模組的時間間隔必須在1秒之內。

步驟7：若CAN-2000C模組收到守衛要求訊息的時間間隔超過1秒，CAN-2000C模組就會發生生存守衛事件，對外傳送EMCY訊息，此時輸出通道的數值會根據物件字典中主索引為0x6206、0x6207、0x6443和0x6444的內容作改變。如果此處停止傳送守衛要求訊息給CAN-2000C模組，它就會發出以下訊息。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	0	0	1	0	0	0	0	1	0	1	0	8	30	81	11	07	00	00	00	00

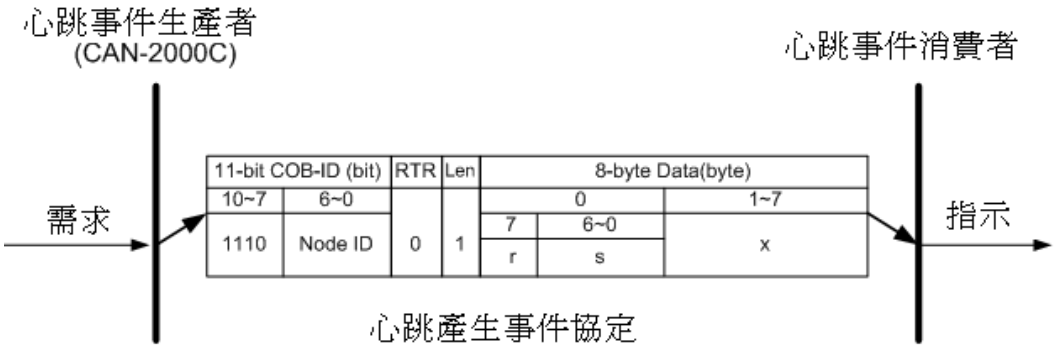
EMCY 消費者 ← **EMCY 生產者 (CAN-2000C)**

EMCY-msg : 30 81 11 07 00 00 00 00

前2 bytes“30 81”，代表緊急錯誤碼。第3個byte“11”代表CAN-2000C模組錯誤暫存器的數值。後5 bytes“07 00 00 00 00”為製造商特定錯誤區域的數值。根據緊急錯誤碼，我們可以知道這個EMCY訊息被傳送是因為發生了生存守衛事件。

心跳事件產生協定 (Heartbeat Protocol)

在物件字典之中，主索引為 0x1017 的物件記錄了心跳事件產生時間(heartbeat time)，且心跳事件產生協定是一個不需使用遠程遠程幀的錯誤控制協定。當主索引為 0x1017 的物件記錄值不為 0 時，CAN-2000C 模組便會循環的發送心跳事件訊息。心跳事件的通訊如下圖所示:



r : 保留 (數值永為 0)

s : NMT 僕端的狀態

4: 停止(STOPPED)狀態

5: 操作(OPERATIONAL)狀態

127: 預操作(PRE-OPERATIONAL)狀態

心跳產生事件協定範例 (Heartbeat Protocol Example)

底下以一個節點 ID 被設為 1 的 CAN-2000C 模組為範例。步驟如下:

步驟1 ：移除CAN-2000C的電源，再重新把CAN-2000C接上電源。這時候CAN-2000C會自動進入預操作狀態。

步驟2：此處透過初始SDO 下載協定，把心跳事件產生時間設定為250。心跳事件產生時間位於物件字典中主索引為0x1017 之處。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0	0	1	0	8	2B	17	10	00	FA	00	00	00

SDO 用戶端 → **SDO 伺服端 (CAN-2000C)**

ccs : 1
n : 2
e : 1
s : 1
m : 17 10 00
d : FA 00 00 00

步驟 3：如果沒有錯誤的話，CAN-2000C 會回覆以下訊息以結束初始 SDO 下載。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	0	1	0	4	60	17	10	00	---	---	---	---

SDO 用戶端 ← **SDO 伺服端 (CAN-2000C)**

scs : 3
m : 17 10 00

步驟4：使用者可以接收到如下訊息，其內記錄了CAN-2000C模組所回報的NMT狀態。心跳事件產生時間的設定為250，使用者可以隨時進行更改。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
Func Code				Node ID																
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	1	0	0	0	0	0	0	0	1	0	1	7F	---	---	---	---	---	---	---

Heartbeat 消費者 ← **Heartbeat 生產者 (CAN-2000C)**

COB-ID : 0x701

t : 1

s 7F

數值 7F 代表 CAN-2000C 模組目前的 NMT 狀態為預操作狀態。

6 CAN-2000C 的物件字典

6.1 通訊描述文件區域

底下會列出在CAN-2000C模組的物件字典中，於通訊描述文件區域內的各個項目。為方便閱讀，此處將所有通訊描述文件區域內的項目區分為數個表格，分別為：“一般通訊項目”、“SDO通訊項目、RxPDO通訊項目”以及“TxPDO通訊項目”。在預設區裡面的“---”代表這個項目沒有定義預設值，或著在某些條件下，才會由CAN-2000C模組內建的韌體來指定預設值。在表格內，如果數字的後方帶有“h”字樣，即代表這個數字是以16進位來表示的。

一般通訊項目(General Communication Entries)

主索引	子索引	描述	型態	屬性	預設值
1000h	0h	裝置型態	UNSIGNED 32	唯讀	---
1001h	0h	錯誤暫存器	UNSIGNED 8	唯讀	---
1003h	0h	“預設錯誤區 ”子索引最大定址範圍	UNSIGNED 8	唯讀	0h
	1h	實際的錯誤 (最新的)	UNSIGNED 32	唯讀	---
	…	…	…	…	---
	5h	實際的錯誤 (最舊的)	UNSIGNED 32	唯讀	---
1005h	0h	SYNC 訊息的 COB-ID	UNSIGNED 32	可讀寫	80h
1008h	0h	製造商所定義的裝置名稱	VISIBLE_STRING	唯讀	---
1009h	0h	製造商所定義的硬體版本	VISIBLE_STRING	唯讀	---
100Ah	0h	製造商所定義的軟體版本	VISIBLE_STRING	唯讀	---
100Ch	0h	守衛時間	UNSIGNED 16	可讀寫	0h
100Dh	0h	生存時間係數	UNSIGNED 8	可讀寫	0h
1010h	0h	“儲存參數” 子索引最大定址範圍	UNSIGNED 8	唯讀	1h
1010h	1h	儲存所有參數(含 PDO 與硬體設定)	UNSIGNED 32	可讀寫	---
1010h	2h	儲存 PDO 通訊參數	UNSIGNED 32	可讀寫	---
1010h	3h	儲存硬體設定參數	UNSIGNED 32	可讀寫	---
1011h	0h	“存回預設參數” 子索引最大定址範圍	UNSIGNED 8	唯讀	1h
1011h	1h	存回所有預設參數(PDO 與硬體設定)	UNSIGNED 32	可讀寫	---
1011h	2h	存回預設 PDO 通訊參數	UNSIGNED 32	可讀寫	---
1011h	3h	存回預設硬體設定參數	UNSIGNED 32	可讀寫	---
1014h	0h	EMCY 訊息的 COB-ID	UNSIGNED 32	可讀寫	80h+Node-ID
1017h	0h	心跳事件產生時間	UNSIGNED 16	可讀寫	0
1018h	0h	“識別物件” 子索引最大定址範圍	UNSIGNED 8	唯讀	4

	1h	供應商的 ID	UNSIGNED 32	唯讀	---
	2h	產品序號	UNSIGNED 32	唯讀	---
	3h	改版版號	UNSIGNED 32	唯讀	---
	4h	序列編號	UNSIGNED 32	唯讀	---

註:

1. 主索引為 0x1000 物件內的項目，其資料格式如下：

附加資訊		一般資訊
bit 31~ bit 24	bit 23 ~ bit16	bit 15 ~ bit 0
特定功能碼	I/O 功能碼	裝置描述文件碼

在CAN-2000C模組上，特定功能碼的數值會保持0。I/O 功能碼則定義了CAN-2000C模組是哪一種性質的裝置，功能碼內的bit 16、17、18、19、20、21、22分別代表著裝置是否有DI、DO、AI、AO、 Counter、PWM、DIO的通道。舉例來說，如果bit16 為1，這就代表著CAN-2000C模組有DI 的通道，如果bit 16 和bit 17 為1，這就代表著CAN-2000C模組有DI 和DO 的通道。功能碼內的bit 20 到bit 19 的數值會保持為0。一般資訊內記錄的為裝置描述文件碼，CAN-2000C模組的裝置描述文件碼為

2. 有關主索引 0x1001 和主索引 0x1003 的物件，請參照 5.3.2 節。

3. 主索引0x1005 的物件裡面存放的是SYNC 的COB-ID。如果要傳SYNC 訊息給CAN-2000C模組， 就必須使用裝置上所記錄的SYNC COB-ID，其格式如下表所示：

Bit 編號	值	代表的意義
31 (MSB)	x	這個 bit 沒有意義，無須理會
30	0	裝置不會產生 SYNC 訊息
	1	裝置會產生 SYNC 訊息
29	0	11-bit ID (CAN 2.0A)
	1	29-bit ID (CAN 2.0B)
28-11	0	如果 bit 29=0，則這 18 個 bit 就為 0
	x	如果bit 29=1，則這18 個bit 就是COB-ID 的bit 28 到bit 11
10-0 (LSB)	x	是 COB-ID 的 bit 10 到 bit 0

CAN-2000C模組不支援產生SYNC 訊息，也不支援29 bits 的COB-ID，所以bit 29、30、31 的數值就會保持為0。

4. 物件字典內，主索引0x1008、0x1009、0x100A 的物件，記錄的是製造商於物件字典內記錄的產品資訊。使用者可以對照ASCII 碼，利用查表的方式來瞭解裝置的產品資訊 。

5. 有關主索引0x100C、0x100D 的物件，請參照5.4.2 節。主所引0x100C 的物件，其數值範圍為0~32767。

主所引0x100D 的物件，其數值範圍為0~255。

6. 主索引 0x1010 和 0x1011 的物件可以儲存模組的設定值 。

7. 有關主索引 0x1014 的物件，請參照 5.3.1 節。

8. 主索引 0x1017 的物件儲存心跳事件的時間。 如果沒有使用心跳事件的話，時間值為 0。此時間值單位為 ms。詳細說明請參考 5.4 節。

SDO 通訊項目 (SDO Communication Entries)

主索引	子索引	描述	型態	屬性	預設值
1200h	0h	伺服 SDO 參數”子索引最大定址範圍	UNSIGNED 8	唯讀	2
	1h	RxSDO 的 COB-ID。(用戶端到伺服端)	UNSIGNED 32	唯讀	600h+Node-ID
	2h	TxSDO 的 COB-ID。(伺服端到用戶端)	UNSIGNED 32	唯讀	580h+Node-ID

RxPDO 通訊項目 (RxPDO Communication Entries)

主索引	子索引	描述	型態	屬性	預設值
1400h	0h	第 1 組 “RxPDO 通訊參數” 子索引的最大定址範圍	UNSIGNED 8	唯讀	2
	1h	第 1 組 RxPDO 的 COB-ID	UNSIGNED 32	可讀寫	200h+Node-ID
	2h	第 1 組 RxPDO 的傳輸型態	UNSIGNED 8	可讀寫	FFh
1401h	0h	第 2 組 “RxPDO 通訊參數” 子索引的最大定址範圍	UNSIGNED 8	唯讀	2
	1h	第 2 組 RxPDO 的 COB-ID	UNSIGNED 32	可讀寫	300h+Node-ID
	2h	第 2 組 RxPDO 的傳輸型態	UNSIGNED 8	可讀寫	FFh
1402h	0h	第 3 組 “RxPDO 通訊參數” 子索引的最大定址範圍	UNSIGNED 8	唯讀	2
	1h	第 3 組 RxPDO 的 COB-ID	UNSIGNED 32	可讀寫	400h+Node-ID
	2h	第 3 組 RxPDO 的傳輸型態	UNSIGNED 8	可讀寫	FFh
1403h	0h	第 4 組 “RxPDO 通訊參數” 子索引的最大定址範圍	UNSIGNED 8	唯讀	2
	1h	第 4 組 RxPDO 的 COB-ID	UNSIGNED 32	可讀寫	500h+Node-ID
	2h	第 4 組 RxPDO 的傳輸型態	UNSIGNED 8	可讀寫	FFh
1404h	0h	第5 組 “RxPDO 通訊參數” 子索引的最大定址範圍	UNSIGNED 8	唯讀	2
	1h	第 5 組 RxPDO 的 COB-ID	UNSIGNED 32	可讀寫	80000000h

	2h	第 5 組 RxPDO 的傳輸型態	UNSIGNED 8	可讀寫	FFh
…	…	…	…	…	…
1409h	0h	第10 組 “RxPDO 通訊參數” 子索引的最大定址範圍	UNSIGNED 8	唯讀	2
	1h	第 10 組 RxPDO 的 COB-ID	UNSIGNED 32	可讀寫	8000 0000h
	2h	第 10 組 RxPDO 的傳輸型態	UNSIGNED 8	可讀寫	FFh

TxPDO 通訊項目 (TxPDO Communication Entries)

主索引	子索引	描述	型態	屬性	預設值
1800h	0	第1 組 "TxPDO 通訊參數" 子索引的最大定址範圍	UNSIGNED 8	唯讀	5
	1	第 1 組 TxPDO 的 COB-ID	UNSIGNED 32	可讀寫	180h+Node-ID
	2	第 1 組 TxPDO 的傳輸型態	UNSIGNED 8	可讀寫	FFh
	3	第 1 組 TxPDO 的抑制時間	UNSIGNED 16	可讀寫	0
	4	此項目被保留	---	---	---
	5	第 1 組 TxPDO 的事件計時器	UNSIGNED 16	可讀寫	0
1801h	0	第 2組 "TxPDO 通訊參數" 子索引的最大定址範圍	UNSIGNED 8	唯讀	5
	1	第 2 組 TxPDO 的 COB-ID	UNSIGNED 32	可讀寫	280h+Node-ID
	2	第 2 組 TxPDO 的傳輸型態	UNSIGNED 8	可讀寫	FFh
	3	第 2 組 TxPDO 的抑制時間	UNSIGNED 16	可讀寫	0
	4	此項目被保留	---	---	---
	5	第 2 組 TxPDO 的事件計時器	UNSIGNED 16	可讀寫	0
1802h	0	第3 組 "TxPDO 通訊參數" 子索引的最大定址範圍	UNSIGNED 8	唯讀	5
	1	第 3 組 TxPDO 的 COB-ID	UNSIGNED 32	可讀寫	380h+Node-ID
	2	第 3 組 TxPDO 的傳輸型態	UNSIGNED 8	可讀寫	FFh
	3	第 3 組 TxPDO 的抑制時間	UNSIGNED 16	可讀寫	0
	4	此項目被保留	---	---	---
	5	第 3 組 TxPDO 的事件計時器	UNSIGNED 16	可讀寫	0
1803h	0	第 4組 "TxPDO 通訊參數" 子索引的最大定址範圍	UNSIGNED 8	唯讀	5
	1	第 4 組 TxPDO 的 COB-ID	UNSIGNED 32	可讀寫	480h+Node-ID
	2	第 4 組 TxPDO 的傳輸型態	UNSIGNED 8	可讀寫	FFh
	3	第 4 組 TxPDO 的抑制時間	UNSIGNED 16	可讀寫	0
	4	此項目被保留	---	---	---

	5	第 4 組 TxPDO 的事件計時器	UNSIGNED 16	可讀寫	0
1804h	0	第 5 組 "TxPDO 通訊參數" 子索引的最大定址範圍	UNSIGNED 8	唯讀	5
	1	第 5 組 TxPDO 的 COB-ID	UNSIGNED 32	可讀寫	80000000h
	2	第 5 組 TxPDO 的傳輸型態	UNSIGNED 8	可讀寫	FFh
	3	第 5 組 TxPDO 的抑制時間	UNSIGNED 16	可讀寫	0
	4	此項目被保留	---	---	---
	5	第 5 組 TxPDO 的事件計時器	UNSIGNED 16	可讀寫	0
…	…	…	…	…	…
1809h	0	第10 組 "TxPDO 通訊參數" 子索引的最大定址範圍	UNSIGNED 8	唯讀	5
	1	第 10 組 TxPDO 的 COB-ID	UNSIGNED 32	可讀寫	80000000h
	2	第 10 組 TxPDO 的傳輸型態	UNSIGNED 8	可讀寫	FFh
	3	第 10 組 TxPDO 的抑制時間	UNSIGNED 16	可讀寫	0
	4	此項目被保留	---	---	---
	5	第 10 組 TxPDO 的事件計時器	UNSIGNED 16	可讀寫	0

6.2 標準化裝置描述文件區域

每個 CAN-2000C 模組有不同的功能。因此關於模組的標準化裝置描述文件區域，請參閱各模組的使用手冊。