PM-31xx CPS 系列
精巧型 CANopen 電力表
使用手冊

保固條款

所有泓格科技製造的產品，泓格科技皆提供對產品本身一年的保固，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。

商標

在本書中所引用的商標及產品名稱均屬原公司所有，書中引用純屬介紹之用，並無侵害之意。

目錄

1. 介紹

1.1 概述

CANopen是一個基於智能領域匯流排(intelligent field bus，如CAN bus)之上的通訊協定。其被使用來發展具備高度彈性組態能力的標準嵌入式網路。

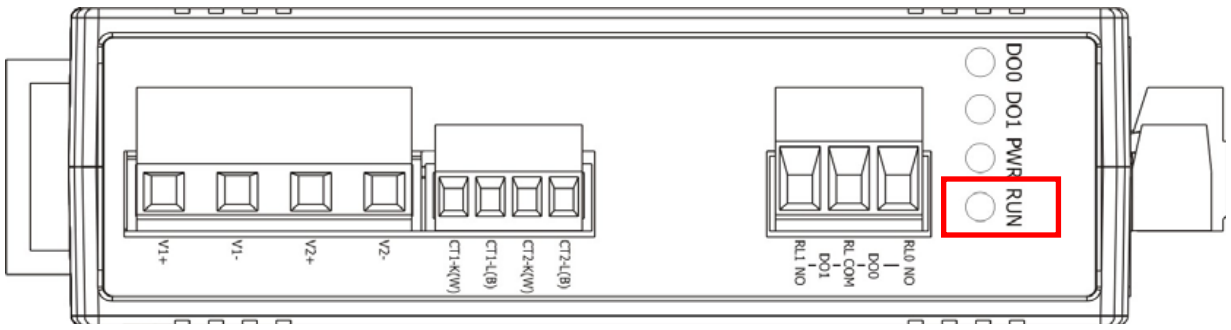
CANopen為了因應不同的需求，提供了多種標準化的通訊物件，像是適合用來傳輸即時(real time)資料的 Process Data Objects (PDO)、適合用來傳輸組態資料的Service Data Objects(SDO)、可進行網路管理的 Network Management Objects(如NMT訊息與錯誤控制)，以及其它具有特殊功能的通訊物件(如Time Stamp，SYNC與EMCY訊息)等…

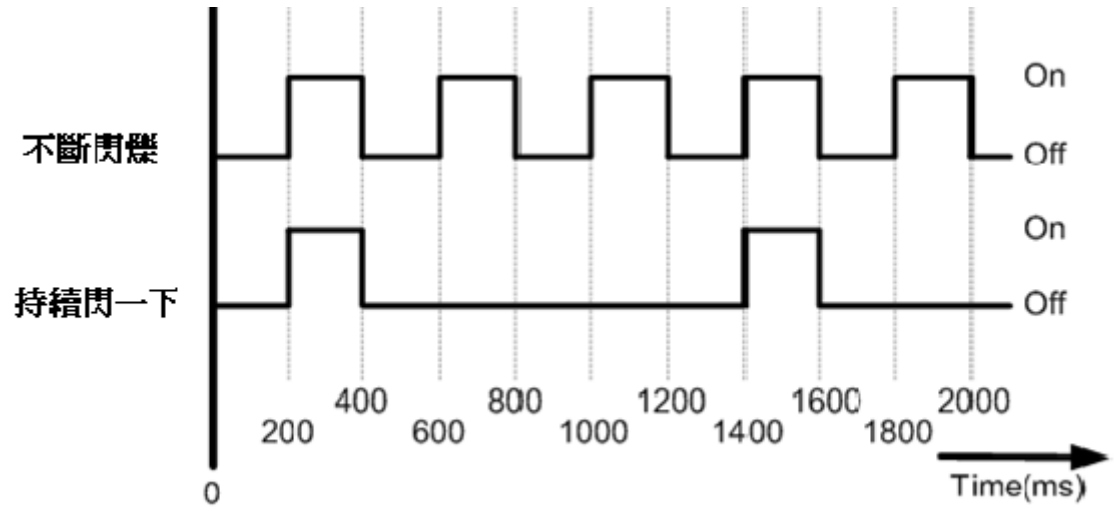
如今，CANopen被使用在各種不同的應用領域，像是醫療設備、工程車輛、航海電子、公眾傳輸與建築自動化等…

PM-31xx-CPS電表內建CANopen介面。可以讓使用者輕易地應用在任何的CANopen網路中。

1.2 LED 燈號狀態

PM-31xx-CPS上的CANopen運行指示燈是用來表示目前CANopen的狀態。下表敘述燈號所表示的狀態:





編號	CANopen運行指示燈	狀態	描述
1	不亮	沒有操作	電源故障或是沒有上電
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-operation)	裝置目前處於預操作狀態
4	恆亮	操作(Operation)	裝置目前處於操作狀態

表 2-3

2. 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。

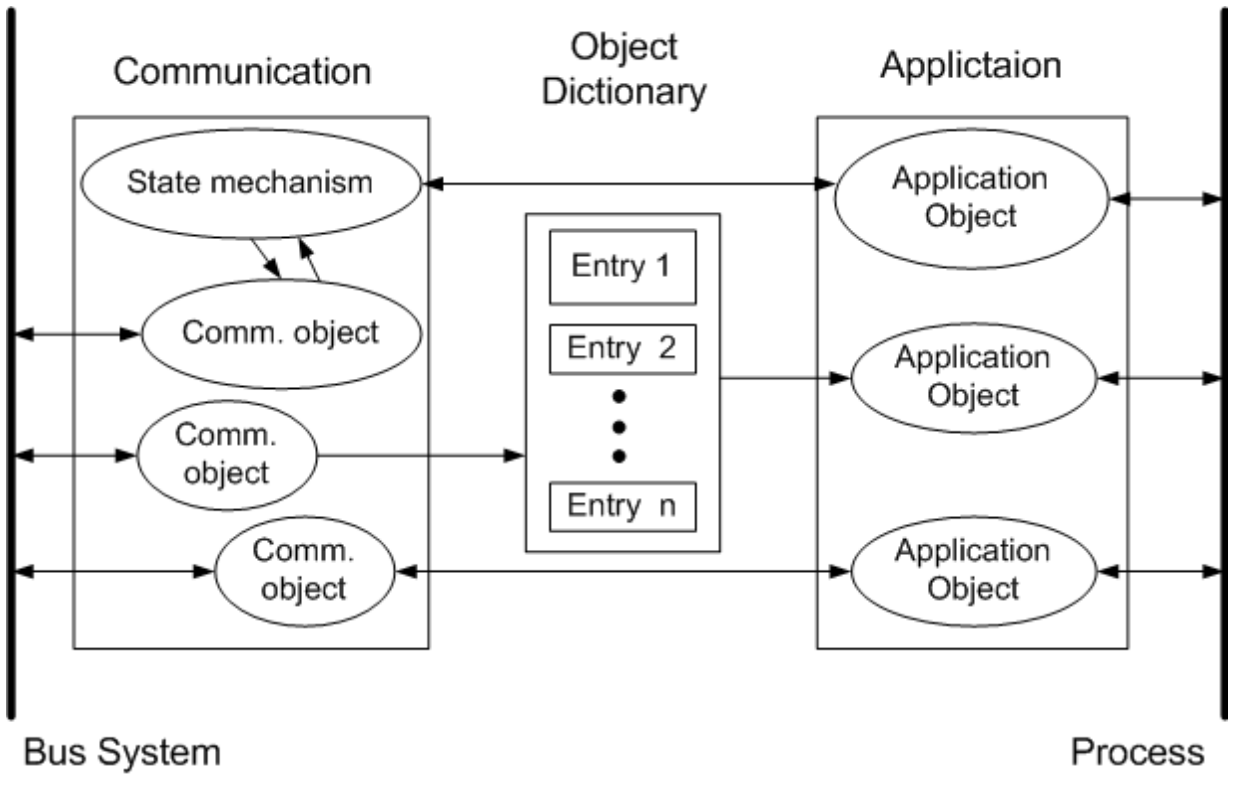
2.1 CANopen 介紹

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)標準機制與錯誤管理(error management)標準機制，來對CANopen的網路進行實作與偵錯。

任何CANopen的裝置均可有效率的存取I/O數值，或者查詢同一網路內其他裝置的節點狀態。一般來說，一個CANopen的裝置可以大略分為三個部分。

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

下圖為CANopen的裝置模型，其內每一部分的功能和概念，將於下描述。



通訊

通訊模型中的通訊部分內含數個不同功能的通訊物件(COB，Communication Object)，裝置與裝置之間就是使用這些COB來傳遞CANopne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SDO、PDO、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的通訊物件則被用來監控CANopen網路中，各節點的狀態，其遵循了主從式(master/slave)的架構 (詳見3.4節)。不管使用哪種COB來進行訊息的傳輸，其格式都會如同下圖。

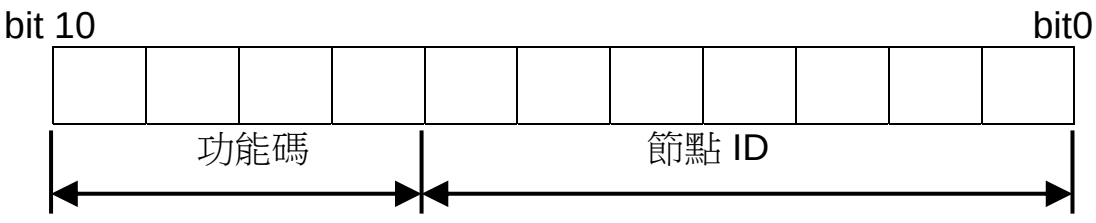
ID	RTR	資料長度	8-byte 資料

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欄位僅有1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(Small 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，也用來區分訊息的用途。另外也決定了網路中的每一點 ，在傳輸訊息時的優先等級。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規範，有部分的COB-ID被保留給特定的通訊物件，或者留作更進一步的用途，使用者不能隨意的使用這些COB-ID。下表整理出被保留的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PDO
601~67F	預設用來接收PDO
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，使用者可以依實際需求，使用剩下來的COB-ID。此處將CANopen協定中所有COB預設的COB-ID列於下表:

(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	節點 ID	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

註: PM-31xx-CPS支援時戳和RxPDO以外的所有物件。

物件字典

物件字典內收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，另外這些項目可以透過一事先定義的(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8bit的整數、8bit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。

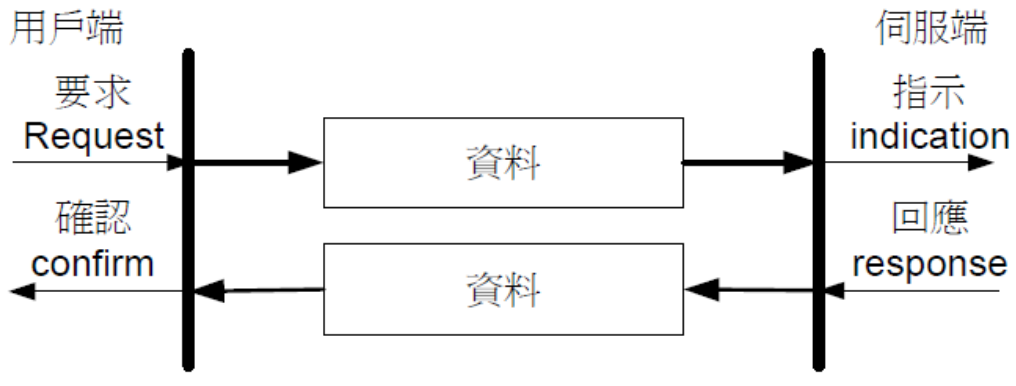
物件在物件字典中的位置，乃是透過一16bit的主索引來定址，如果某個物件內有很多個項目，那這些項目還必需透過一個8bit的子索引來定址，如果某物件內只有一個項目，那用主索引便可對這個項目定址。

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	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(Device Profile Specific Static Data Types)
0x0080 – 0x009F	裝置描述文件特定的複合資料型別(Device Profile Specific Complex Data Types)
0x00A0 – 0x0FFF	保留作更進一步的用途
0x1000 – 0x1FFF	通訊描述文件區域(Communication Profile Area)
0x2000 – 0x5FFF	製造商特定描述文件區域(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

2.2 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，若使用者要傳送一個SDO給PM-31xx-CPS，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PM-31xx-CPS希望對外傳送一個SDO訊息，則此SDO對裝置而言就是TxPDO。這個訊息的COB-ID就必須是裝置的TxPDO 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。

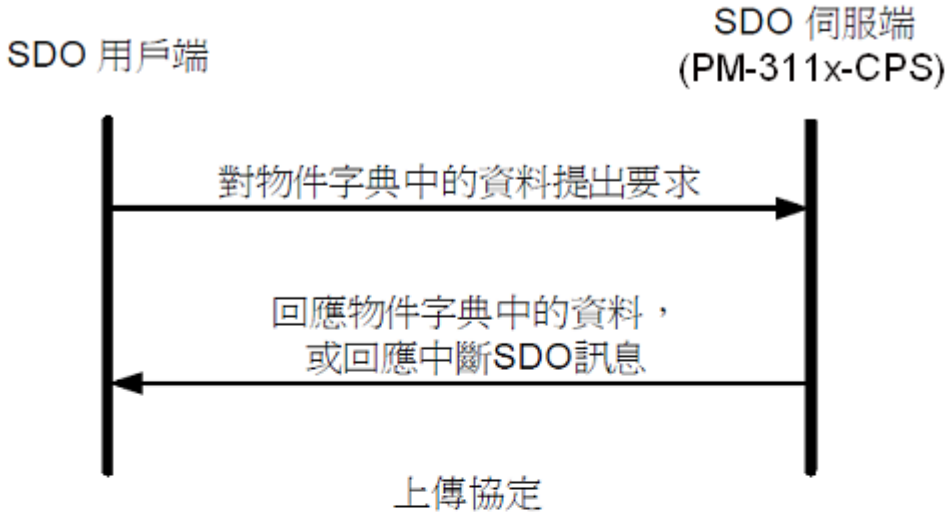
如果SDO用戶端必須要由SDO伺服端上的物件字典獲得資訊，則需要使用上傳SDO協定(upload SDO protocol)或SDO區塊上傳協定(SDO block upload protocol)。前者適合用來傳輸較少量的資料，後者則適合用來傳輸較大量的資料。

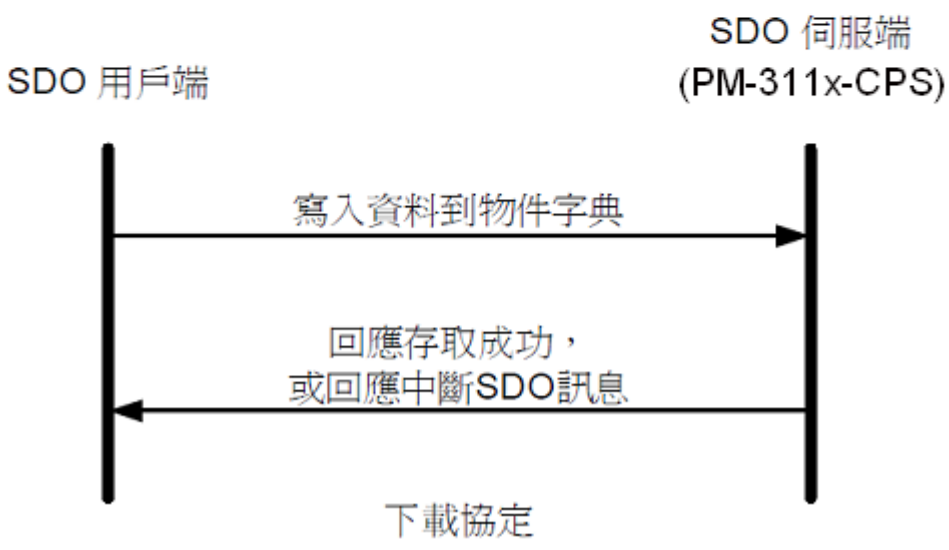
當SDO用戶端想要更改SDO伺服端的物件字典內容時，則可以透過下載SDO協定(download SDO protocol)或SDO區塊下載協定(SDO block download protocol)。這兩者之間的差異就如同於上傳SDO協定與SDO區塊上傳協定之間的差異。

此外，因為物件字典中不同的項目會有不同的存取類別，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。

PM-31xx-CPS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傳輸。有關PM-31xx-CPS的上傳協定和下載協定，其大致上的概念可參考下圖:





2.3 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…由於CAN bus的傳輸資料格式限制，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的通訊物件更有效率。

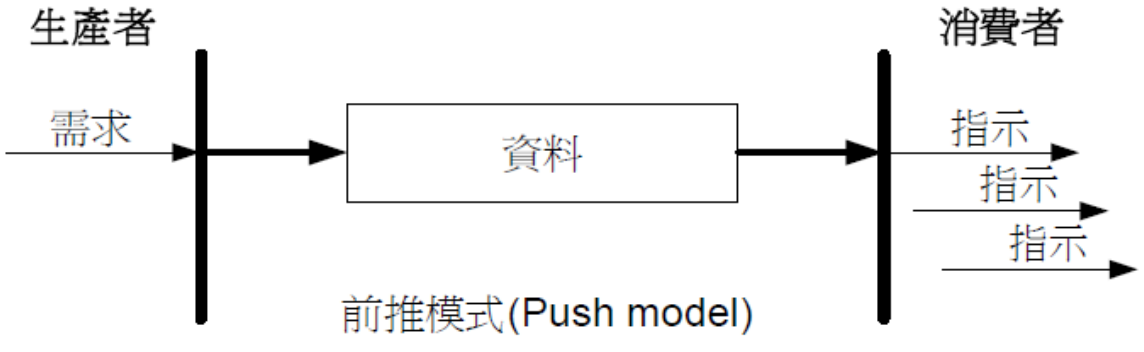
PDO的通訊模式(Communication Mode)

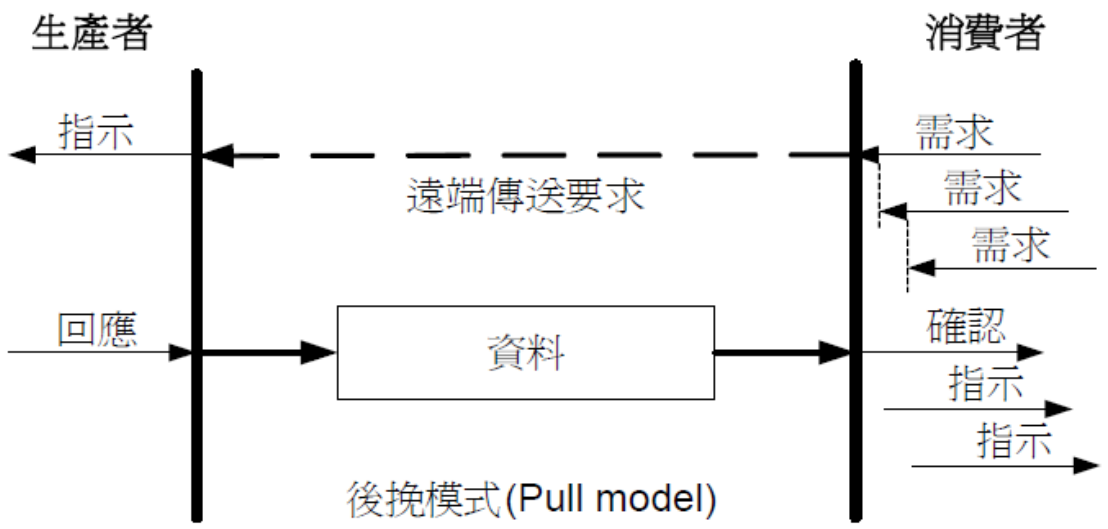
PDO 的傳送與接收通訊模型，乃是依據生產者/消費者(producer/consumer)的架構所實作的。這種通訊模式也可稱為推挽式(push/pull)模型。

當PDO前推模式(push mode)開始進行時，必須有一個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訊息必須要先在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。並決定接受或者是丟棄這個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)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CAN bus上的每一個PDO消費者。

PDO的通訊架構圖如下所示:





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的DI/AI通道的數值。而在裝置用來接收資料的PDO就是RxPDO，其常被用來改變裝置上DO/AO通道的數值。

以PM-31xx-CPS為例，PM-31xx-CPS只支援TxPDO。當有PDO消費者傳送遠端傳送要求給PM-31xx-CPS時，此一訊息的COB-ID就必須使用TxPDO COB-ID。因為對 PM-31xx-CPS來說回覆遠端傳送要求的動作是屬於TxPDO的傳送。

PDO的觸發模式(Trigger Modes)

對PDO生產者而言，PDO訊息傳輸的觸發條件可分為三種。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。

註: PM-31xx-CPS不支援此觸發條件。

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通訊參數內的事件記錄器，則當經過了事件記錄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，PM-31xx-CPS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 (PDO Transmission Types)

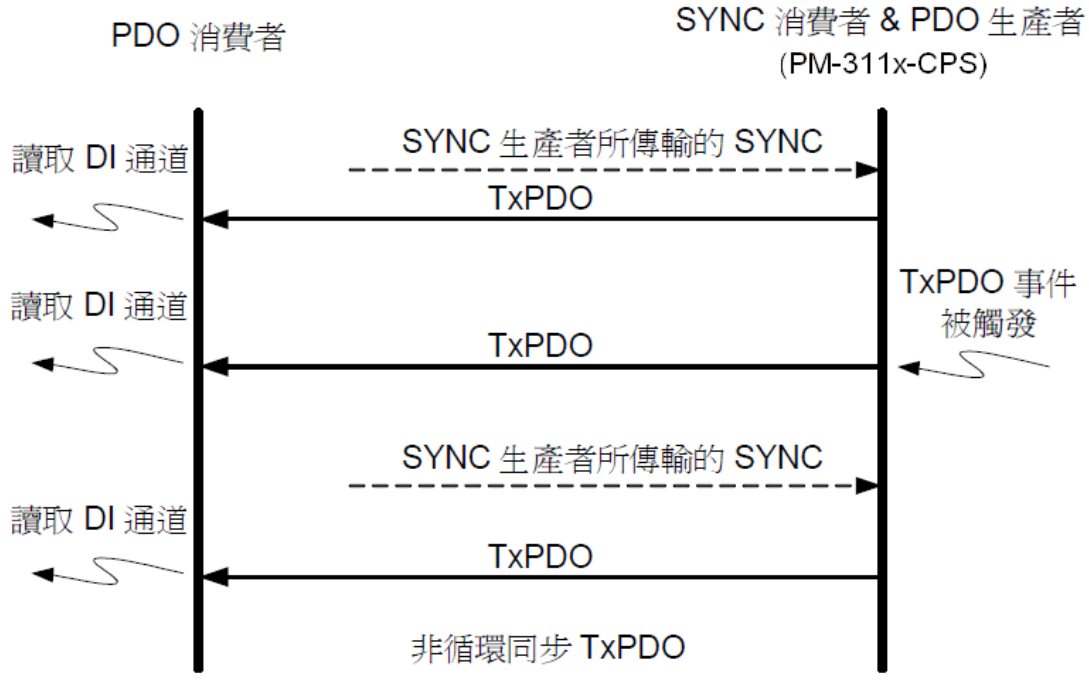
一般來說，PDO的傳輸型態可大略分為兩種。分別為同步和非同步。在同步傳輸型態下，必須要藉由SYNC訊息的接收，才會觸發PDO的收送。在非同步傳輸型態下，PDO傳輸的觸發是與SYNC物件相互獨立的。

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。而非同步傳輸型態則可分為兩種。分別為唯遠端傳送要求非同步以及非同步。

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。或者裝置收到此TxPDO的RTR訊息，此時裝置也會對外傳送此TxPDO。

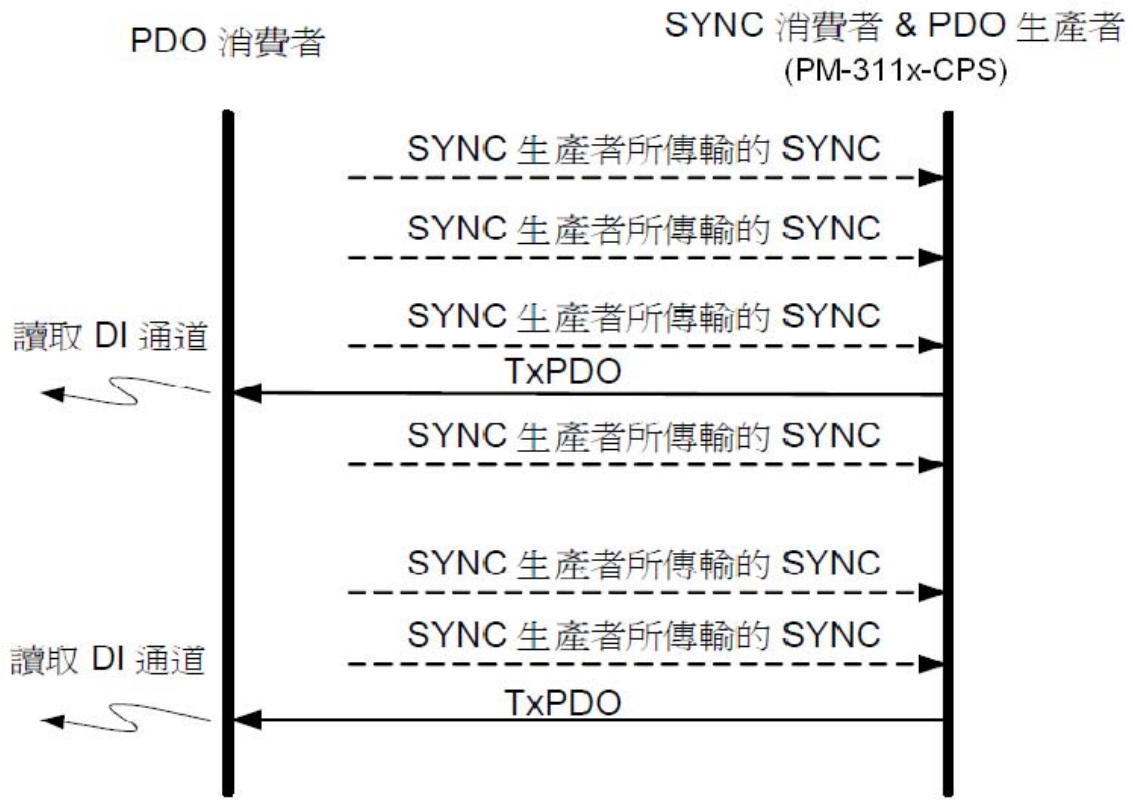
關於非同步循環的傳輸型態，可以參考下圖：



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PM-31xx-CPS將會在收到3個SYNC物件後，進行此TxPDO的傳送。此運作概念如下圖所示:



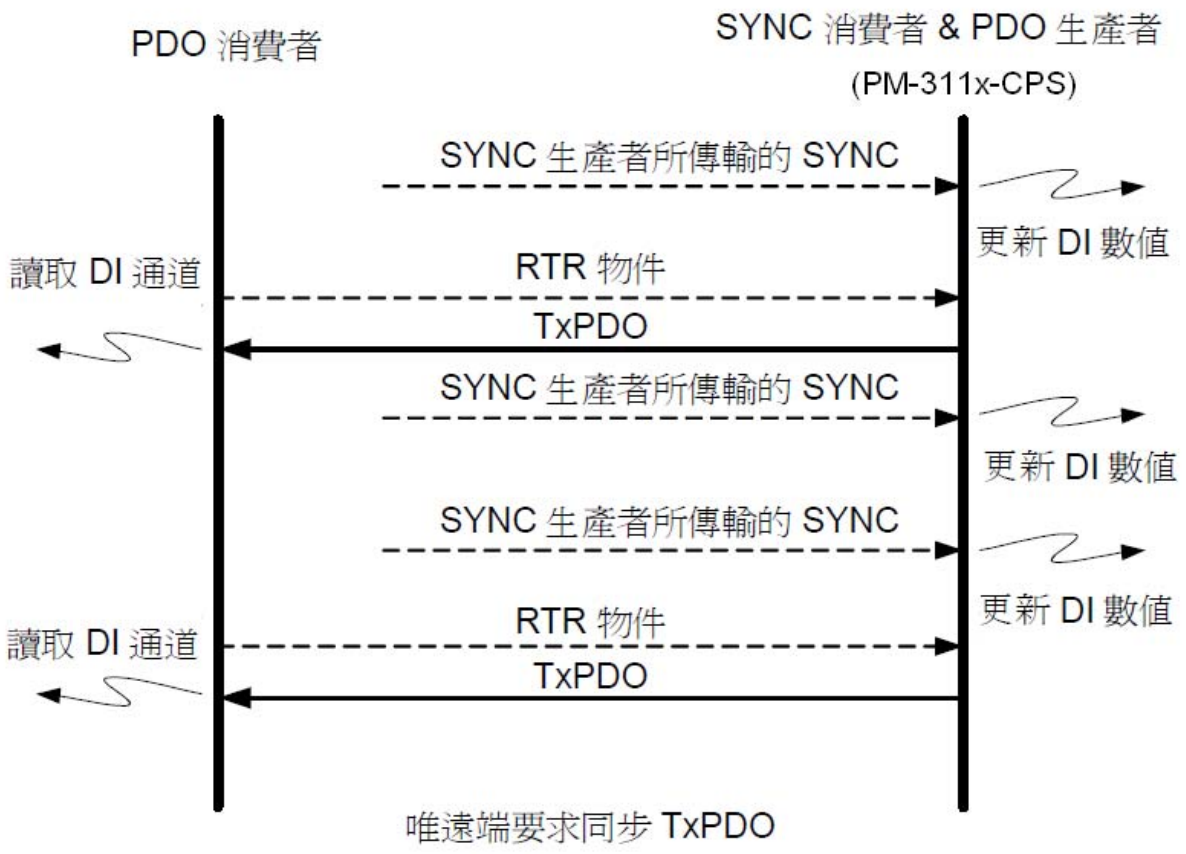
循環同步 TxPDO

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內紀錄的有可能是過時的資料。

只有TxPDO能被設定為這種傳輸型態。關於唯遠端傳送要求同步的概念，可以參考下圖:

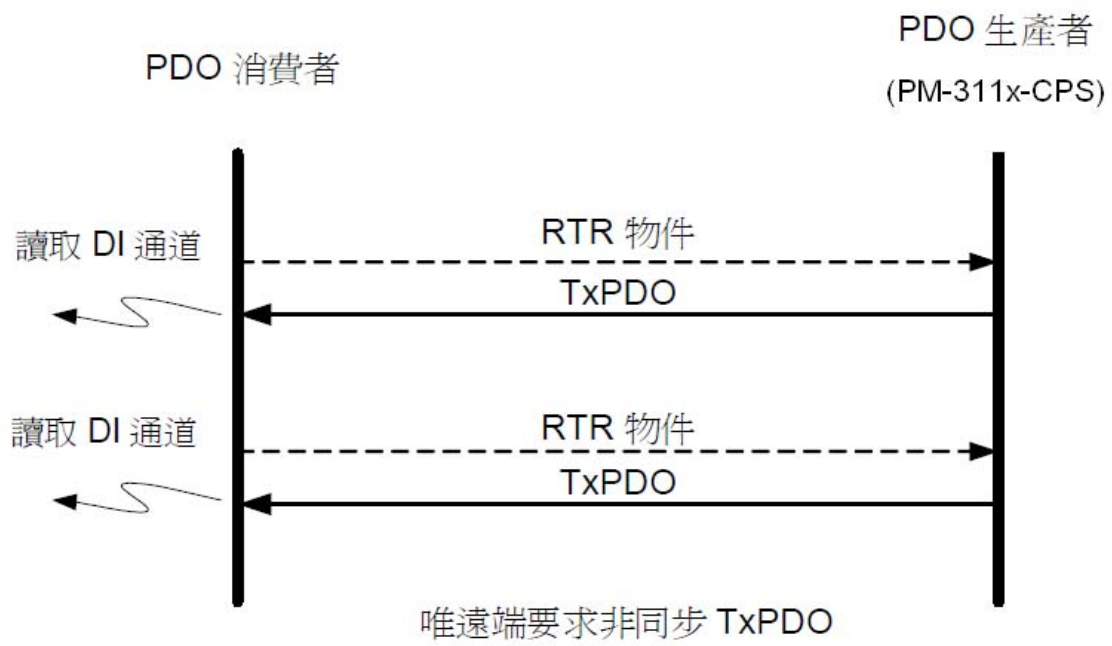
註: PM-31xx-CPS 不支援此種傳輸型態



唯遠端要求同步 TxPDO

唯遠端傳送要求非同步(RTR-only asynchronous)

只有TxPDO的傳輸型態可以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。使用者可以參考下圖:



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，裝置就會直接啟用RxPDO，不需要接收SYNC訊息。

註: PM-31xx-CPS不支援此傳輸型態

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上有兩個節點，其中一個想要傳送COB-ID為0x181的訊息，另一個想要傳送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的訊息必須等匯流排再次空閒(bus idle)，才能夠被傳送。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映射物件 (PDO Mapping Objects)

PDO映射物件提供了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映射物件的內容)，又可以透過SDO訊息來加以改變。所有的PDO映射物件均存放在通訊描述文件區域(Communication Profile Area)之中。

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TxPDO和4組RxPDO
- 第1組RxPDO和TxPDO的映射，分別對應到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該填入那些資料。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。

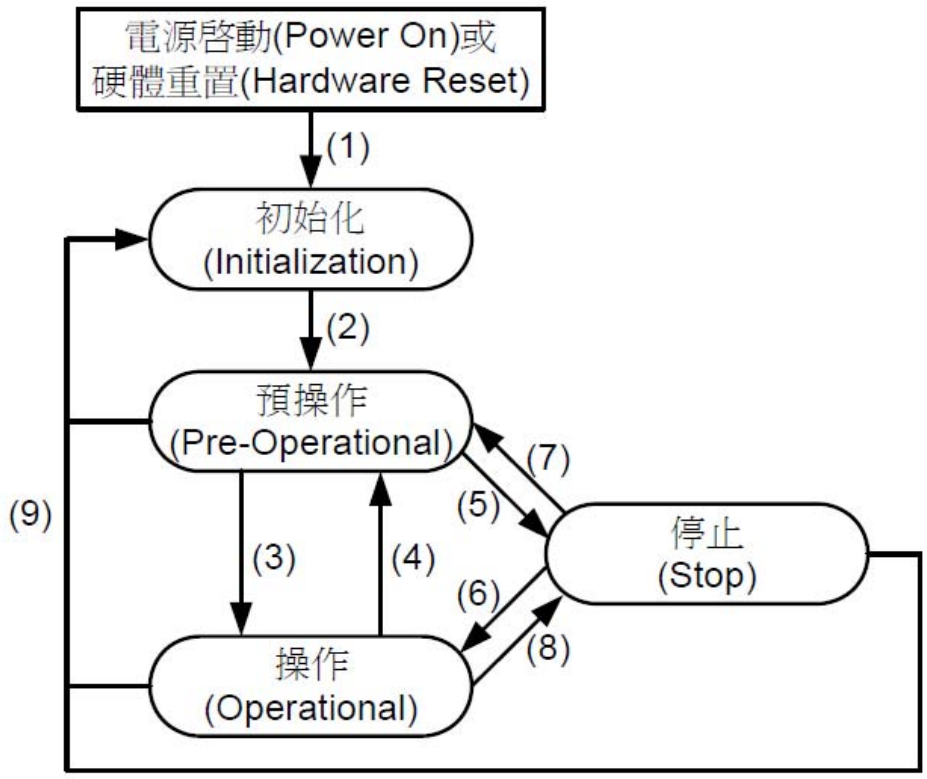
2.4 NMT 介紹

在CANopen內，所謂NMT通訊物件，其通訊模型乃依循著節點導向(node-oriented)以及主僕式(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，其它的節點則作為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，可以用來確認節點是否存活。

2.4.1 模組控制協定

在介紹模組控制協定(modules control protocol)之前，讓我們先把焦點放在NMT狀態機制(NMT state mechanism)的架構上。下圖列出了各NMT狀態(NMT state)之間的關係以及各NMT僕端(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)	“啟動遠端操作節點(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-operation)	操作 (Operational)	停止 (Stopped)
PDO			O	
SDO		O	O	
SYNC		O	O	
Time Stamp		O	O	
EMCY		O	O	
Boot-Up	O			
NMT		O	O	O

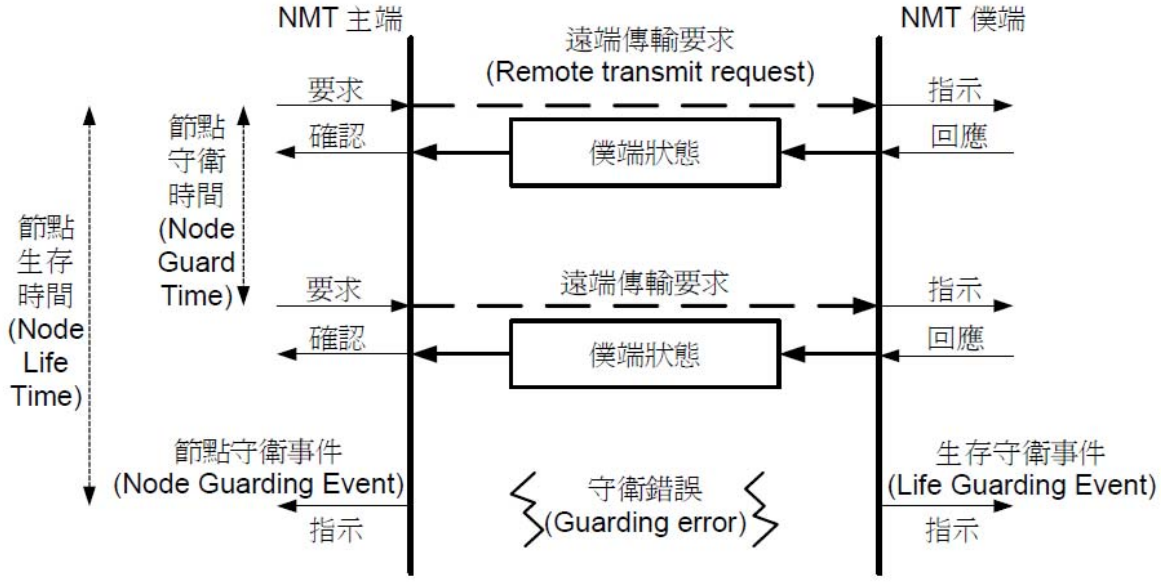
2.4.2 錯誤控制協定

錯誤控制協定共有兩種。分別為節點守衛協定(Node Guarding Protocol)與心跳產生協定(Heartbeat 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PM-31xx-CPS上啟用這項功能。

下面將針對此兩種錯誤控制協定作進一步介紹。

<u>節點守衛協定(Node Guarding Protocol)</u>

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:

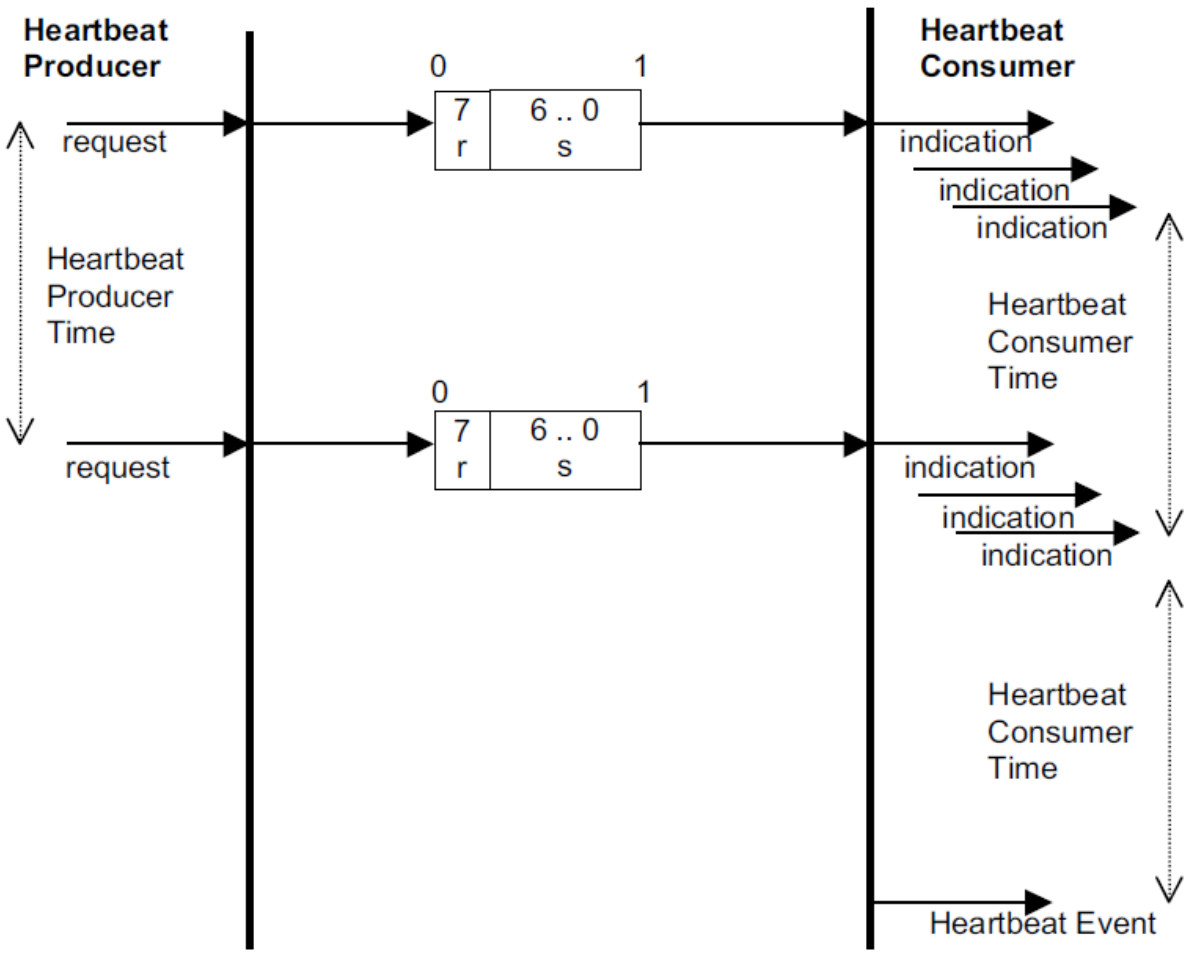


每隔一段固定的時間間隔，NMT的主端就會去對每個NMT僕端作輪詢(poll)的動作。此一時間間隔即所謂的節點守衛時間(node guard time)，使用者可以對不同的NMT僕端設定不同的節點守衛時間。

NMT僕端收到NMT主端的詢問之後，會回答NMT僕端目前所處的狀態，像是停止、操作或者是預操作等狀態。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。

心跳協定(Heartbeat Protocol)

心跳協定的通訊模型依循著生產者與消費者(Producer/Consumer)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:



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(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，心跳消費者就會產生一個心跳事件。

3. CANopen 通訊集

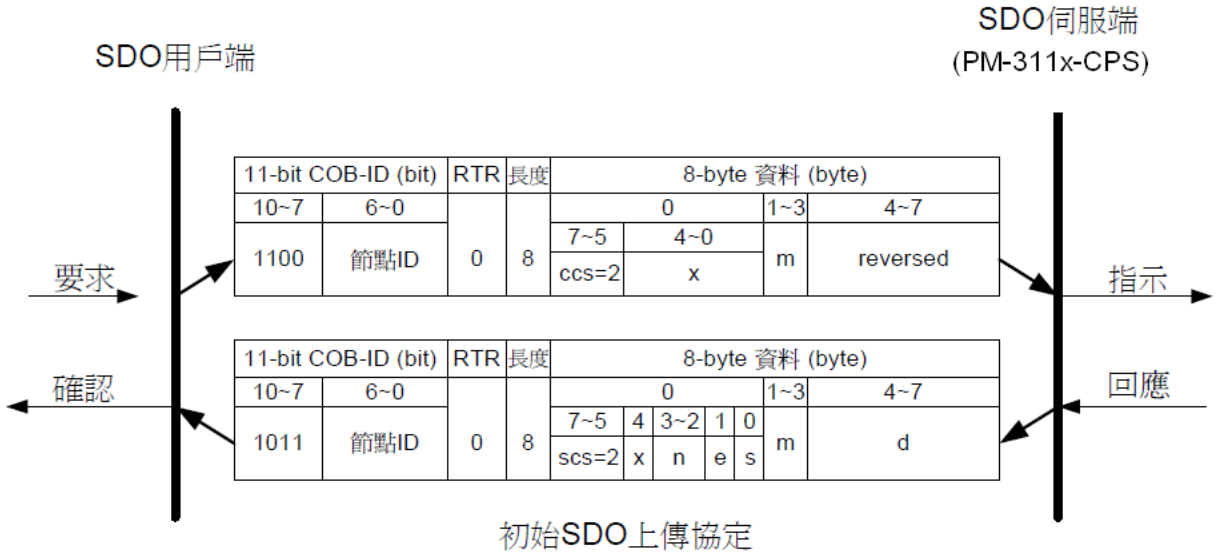
3.1 SDO 通訊集

3.1.1 上傳 SDO 協定

初始SDO上傳協定 (Initiate SDO Upload Protocol)

在傳輸SDO區段(SDO segments)之前，用戶端和伺服端必須先利用初始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伺服端上傳的資料長度小於或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)。



初始SDO上傳協定
(Initiate SDO Upload Protocol)

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(initiate 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，及第8-n個byte到第7個byte內沒有節段資料(segment data)。

e : 傳輸型態(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SOD上傳協定即可傳送完畢。若**e**=0，就必須要進行上傳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表示主索引，後一個byte表示子索引。

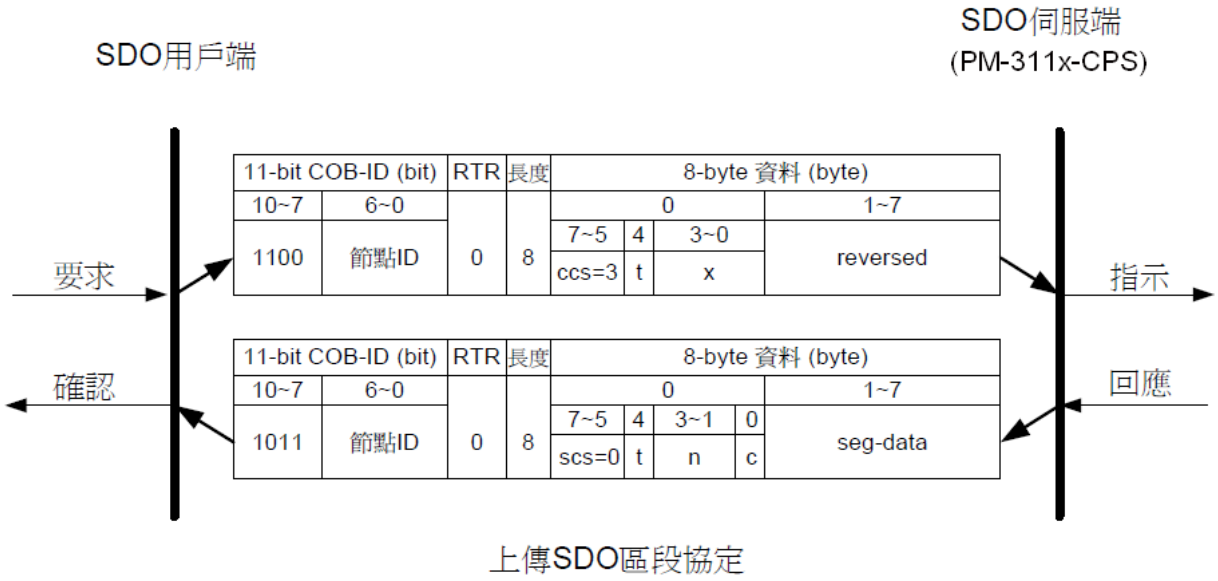
d : 資料
e=0, **s**=0: 表示**d**被保留，留待進一步使用。
e=0, **s**=1: **d**包含了欲上傳的byte 數目，其中byte 4內含最低有效位元(least significant bit)，byte 7內含最高有效位元(most significant bit)
e=1, **s**=1: **d**的內容為欲上傳的資料，其長度為**4-n**。資料編碼的方式取決於主索引和子索引在物件字典上參照項目的資料型別。
e=1, **s**=0: **d**的內容為未標示長度的上傳資料。

x : 沒有被使用，數值永為0

reserved : 保留待進一步使用，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圖所示。



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 request)

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: 還有節段等待被上傳
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。

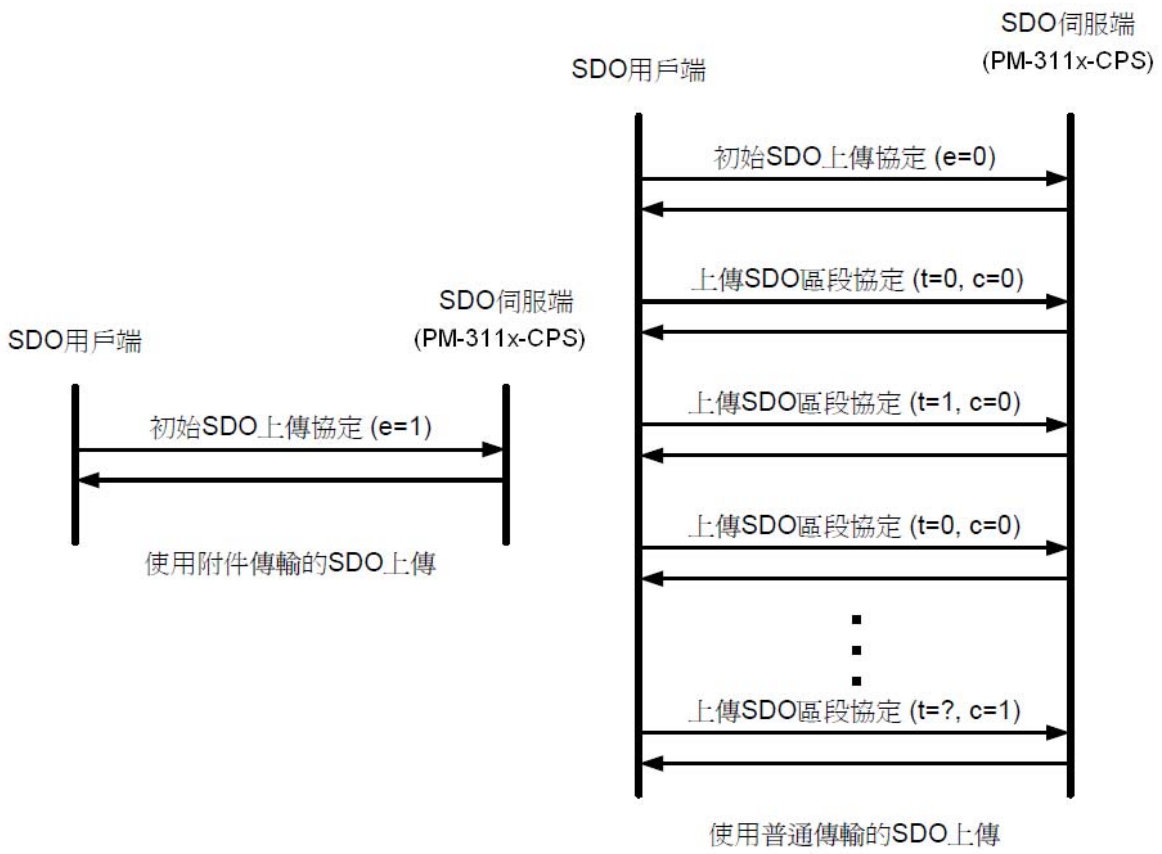
n ：n 內含**seg-data**欄位內沒有節段資料的byte數目，即第8-n個byte到第7個byte內，沒有節段資料。如果n=0，代表節段的大小沒有被指示。

x : 沒有被使用，數值永為0

reserved : 保留待進一步使用，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在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0x1800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伺服器上傳主索引為0x1800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0x1008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PM-31xx-CPS，以取得在物件字典的通訊描述文件區域內，主索引為0x1800 且子索引為00 的項目。有關此項目的作用，使用者可以參照第5章。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
功能碼				節點 ID																
10	9	8	7	6	5	4	3	5	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	18	00	00	00	00	00

SDO 用戶端 → **SDO 伺服端 (PM-31xx-CPS)**

ccs : 2
m : 00 18 00
因為低字組會先被傳送，所以第1個byte “00”代表0x1800的低字組，而第2個 byte “18”代表0x1800的高字組，至於最後一個byte “00”則代表子索引0x00。

步驟 2. PM-31xx-CPS將會回應主索引為0x1800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							
功能碼				節點 ID																
10	9	8	7	6	5	4	3	5	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	0	1	0	8	4F	00	18	00	05	00	00	00

SDO 用戶端 ← **SDO 伺服端 (PM-31xx-CPS)**

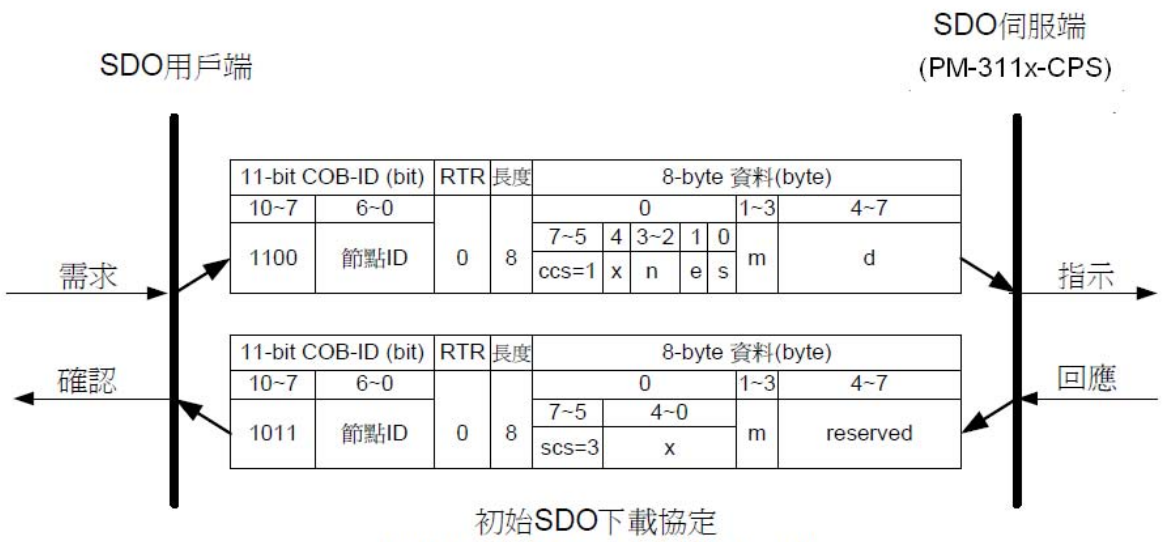
scs : 2
n : 3
e : 1
s : 1
m : 00 18 00
d : 05 00 00 00
因為**n**=3所以只有第4個byte會有資料，於此處其值為05。

下載SDO協定

初始SDO下載協定(Initiate SDO Download Protocol)

下載SDO協定與上傳SDO協定十分類似，其通訊協定僅有部分參數與上傳SDO協定不同。

下載SDO協定也可被分為兩個部分，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。



初始SDO下載協定
(Initiate SDO Download Protocol)

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

scs : 伺服端命令識別符(server command specified)
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。

e : 傳輸型態(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，就必須要進行下載SDO節段協定。

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 : **e**=0, **s**=0: 表示**d**被保留，留待進一步使用。
e=0, **s**=1: **d**包含了欲下載的byte 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**的內容為欲下載的資料，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上參照項目的資料型別。
e=1, **s**=0: **d**的內容為未標示長度的下載資料。

x : 沒有被使用，數值永為0

reserved : 保留等待進一步使用，數值永為0。

<u>下載區段協定(Download Segment Protocol)</u>

下載SDO區段協定
(Download SDO Segment Protocol)

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

scs : 伺服端命令識別符(server command specified)
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bytes內，沒有節段資料。如果n=0，則代表節段的大小還沒有被指示。

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

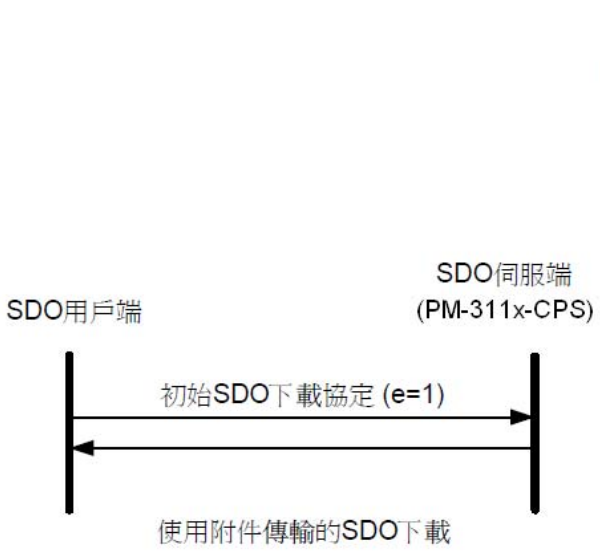
t ：交替位元(toggle bit)
對每一連續的下載節段而言，每一個節段的交替位元均需與其上一個或下一個節段的交替位元不同。而第1個節段的交替位元必須設定為0，另外，要求訊息和回應訊息的交替位元必須相同。

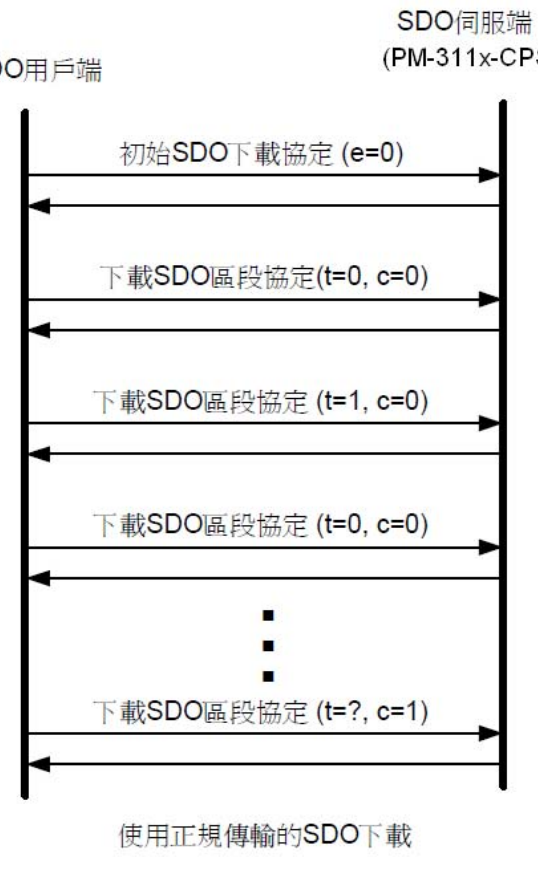
x : 沒有被使用，數值永為0

reserved : 保留待進一步使用，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PM-31xx-CPS 內，所有可被寫入的物件項目，其資料長度均不超過4 bytes，底下僅示範如何使用附件傳輸。

- **附件傳輸(expedited transfer)範例**

步驟 1. 傳送 RxSDO訊息給 PM-31xx-CPS，以更改在通訊描述文件區域內，主索引為0x1800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PM-31xx-CPS的節點ID被設定為1。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
功能碼				節點 ID																
10	9	8	7	6	5	4	3	5	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	18	02	05	00	00	00

SDO 用戶端 → **SDO 伺服端 (PM-311x-CPS)**

ccs : 1
n : 3
e : 1
s : 1
m : 00 18 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. PM-31xx-CPS將會回覆此訊息以結束資料的下載。

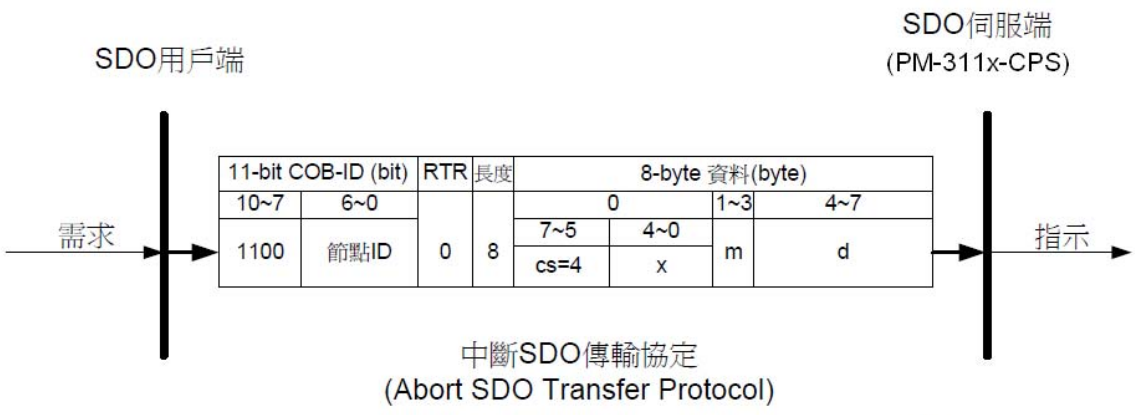
11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
功能碼				節點 ID																
10	9	8	7	6	5	4	3	5	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	0	1	0	8	60	00	18	02	00	00	00	00

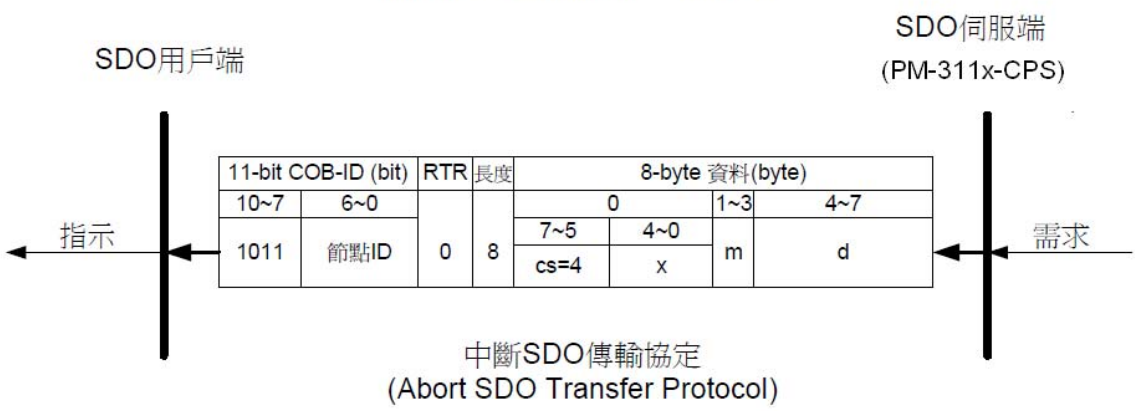
SDO 用戶端 ← **SDO 伺服端 (PM-31xx-CPS)**

scs : 3
m : 00 18 02

3.1.2 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，或者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他理由不想完成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，傳輸中斷訊息以中斷SDO的傳輸。底下是中斷SDO傳輸協定(Abort SDO Transfer Protocol)的介紹。





cs	:	命令識別符(command specified) 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(multiplexer) 其代表欲中斷SOD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，後一個代表子索引。
d	:	內含 4-byte的中斷碼(Abort Code)，代表其中斷原因。

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) 無效或者無法解釋。
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	基於硬體錯誤所導致的存取失敗。
0607 0010h	基於服務參數(service parameter)長度不符所引起的資料型別不符。
0607 0012h	基於服務參數(service parameter)長度過長所引起的資料型別不符。
0607 0013h	基於服務參數(service parameter)長度過短所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	一般性錯誤。
0800 0020h	資料無法傳送或者儲存到應用程式(application)。
0800 0021h	因為本地端控制(local control)，資料無法傳送或者儲存到應用程式。
0800 0022h	因為目前的裝置狀態，資料無法傳送或者儲存到應用程式。
0800 0023h	物件字典在動態產生時發生錯誤，或者物件字典不存在(若物件字典的產生必須藉由載入檔案來完成，萬一檔案載入時發生錯誤，那麼物件字典就無法被建立)。

<u>中斷SDO傳輸範列</u>

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PM-31xx-CPS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上載協定，傳送RxSDO訊息給PM-31xx-CPS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							
功能碼				節點 ID																
10	9	8	7	6	5	4	3	5	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 (PM-31xx-CPS)**

ccs : 2
m : 08 10 01

步驟 2. PM-31xx-CPS會回應中斷SDO訊息給用戶端。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
功能碼				節點 ID																
10	9	8	7	6	5	4	3	5	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 (PM-31xx-CPS)**

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”。

3.2 PDO 通訊集

3.2.1 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 objects)，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 29 bits COB-ID 內的第 28-11 bits
10-0(LSB)	X	COB-ID 內的第 10-0 bits。

註: PM-31xx-CPS 僅支援 CAN 2.0A 規範

此處將預設的PDO COB-ID整理如下表:

PDO 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COB-ID使用者不可以拿來自行定義之外，其它的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。

3.2.2 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	----------Reserved----------				
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物件才能觸發TxPDO的傳送。
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，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。

4. PM-31xx-CPS不支援RxPDO。

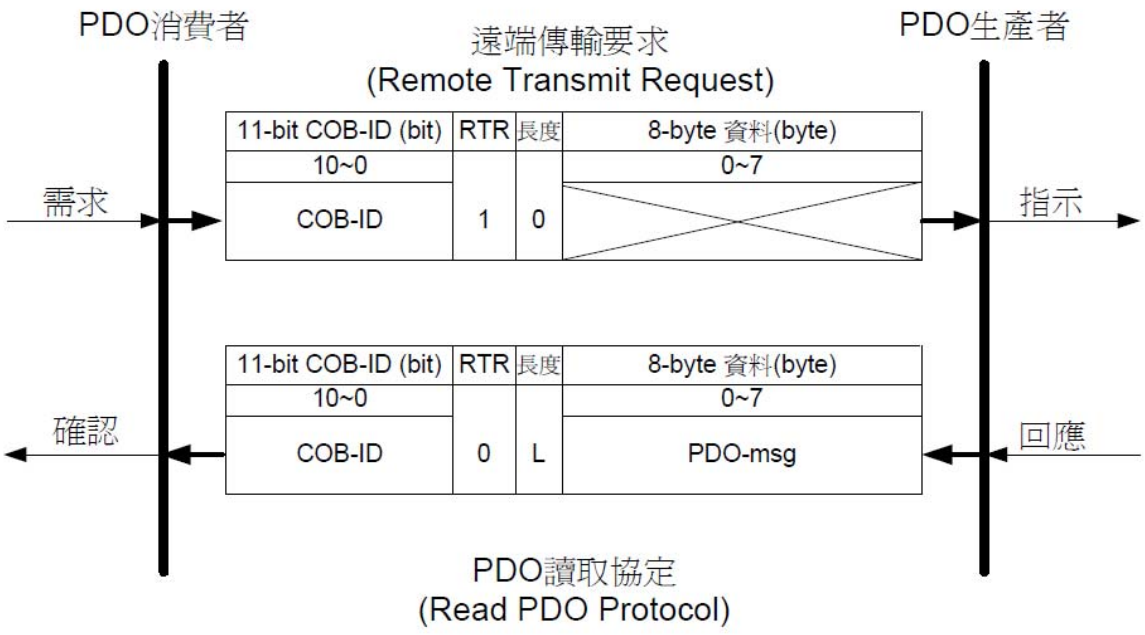
3.2.3 PDO 通訊規則

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為0x1400到0x1BFF之間。而在PM-31xx-CPS內， 沒有使用到RxPDO物件。TxPDO的通訊參數存放於物件字典主索引0x1800到0x1813之間。 其TxPDO的映射參數存放於物件字典主索引0x1A00到0x1A13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。

更進一步來說，譬如第1組TxPDO通訊參數存放於物件字典主索引為0x1800的地方，而其相對應的映射參數便會存放於物件字典主索引為0x1A00的地方，可依序推得主索引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PM-31xx-CPS，要求裝置改變NMT狀態為操作狀態。詳細的內容可以將於3.4節內作介紹。

順帶一提，透過PDO來傳送訊息時，PDO內的資料長度必須和其對應的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，其餘部分則丟棄。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。



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

PDO 通訊範例

在開始舉例示範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)進行。

步驟0: PM-31xx-CPS接收到以下訊息時，若無異常，便會改變其NMT 狀態為NMT操作狀態。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
功能碼				節點 ID																
10	9	8	7	6	5	4	3	5	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 (PM-31xx-CPS)**

cs : 1
節點 ID : 1

步驟 1. 使用者可以透過傳送如下的第2組TxPDO訊息，來讀取PM-31xx-CPS的資料。

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

PDO
生產者 → **PDO**
消費者

COB-ID : 0x281

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
功能碼				節點ID																
10	9	8	7	6	5	4	3	5	1	0			0	1	2	3	4	5	6	7
0	1	0	1	0	0	0	0	0	0	1	0	8	91	64	8A	BC	76	20	06	C0

PDO
生產者 ← **PDO**
消費者

COB-ID : 0x281
L : 8
PDO-msg : 91 64 8A BC 76 20 06 C0

- **事件計時器(Event Timer)功能展示**

步驟 2. 此處利用SDO存取物件字典內主索引為0x1801且子索引為5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							
功能碼				節點ID																
10	9	8	7	6	5	4	3	5	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0	0	1	0	8	2B	01	18	05	E8	03	00	00

SDO 用戶端 → **SDO 伺服端**
(PM-31xx-CPS)

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**等於2，所以最後2 bytes的內容“00 00”不具任何意義。

步驟 3. 如果SOD傳輸成功，PM-31xx-CPS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							
功能碼				節點ID																
10	9	8	7	6	5	4	3	5	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	0	1	0	8	60	01	18	05	00	00	00	00

SDO 用戶端 ← **SDO 伺服端 (PM-31xx-CPS)**

scs : 3
m : 01 18 05

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。底下是連續3 次接收到的第2 組TxPDO 訊息，每個訊息接收到的間隔時間是1秒。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
功能碼				節點ID																
10	9	8	7	6	5	4	3	5	1	0			0	1	2	3	4	5	6	7
0	1	0	1	0	0	0	0	0	0	1	0	8	58	84	8A	BC	1F	46	06	C0

PDO 生產者 ← **PDO 消費者**

COB-ID : 0x281
L : 8
PDO-msg : 58 84 8A BC 1F 46 06 C0

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
功能碼				節點ID																
10	9	8	7	6	5	4	3	5	1	0			0	1	2	3	4	5	6	7
0	1	0	1	0	0	0	0	0	0	1	0	8	A9	2F	8A	BC	33	46	06	C0

PDO
生產者

PDO
消費者

COB-ID : 0x281
L : 8
PDO-msg : A9 2F 8A BC 33 46 06 C0

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
功能碼				節點ID																
10	9	8	7	6	5	4	3	5	1	0			0	1	2	3	4	5	6	7
0	1	0	1	0	0	0	0	0	0	1	0	8	31	1F	8A	BC	47	46	06	C0

PDO
生產者

PDO
消費者

COB-ID : 0x281
L : 8
PDO-msg : 31 1F 8A BC 47 46 06 C0

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

SDO 用戶端 → **SDO 伺服端 (PM-31xx-CPS)**

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							
功能碼				節點ID																
10	9	8	7	6	5	4	3	5	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	0	1	0	8	60	01	18	05	00	00	00	00

SDO 用戶端 ← **SDO 伺服端 (PM-31xx-CPS)**

scs : 4
m : 01 18 05

- **自訂PDO COB-ID**

步驟 4. 使用者可以自訂PDO的COB-ID來存取第5組之後的TxPDO。下面的例子是以設定0x182(當站號為1時，PM-31xx-CPS中沒有使用到這個COB-ID)為第5組TxPDO的COB-ID來存取資料。自訂PDO COB-ID前，需確認COB-ID的bit 31是否為0，COB-ID bit 31必須為0才可更改。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
功能碼				節點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	23	05	18	01	82	01	00	00

SDO 用戶端 → **SDO 伺服端 (PM-31xx-CPS)**

ccs : 1
n : 0
e : 1
s : 1
m : 05 18 01
d : 82 01 00 00

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
功能碼				節點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	60	05	18	01	00	00	00	00

SDO 用戶端 ← **SDO 伺服端 (PM-31xx-CPS)**

SCS : 4
m : 05 18 01

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
功能碼				節點ID																
10	9	8	7	6	5	4	3	5	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 (PM-31xx-CPS)**

cs : 1
節點ID : 1

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
功能碼				節點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	0	2	1	0	00	00	00	00	00	00	00	00

PDO
生產者 → **PDO**
消費者

COB-ID : 0x182
L : 0

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
功能碼				節點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	0	2	0	8	3F	55	DC	42	3F	CA	66	3E

PDO
生產者 ← **PDO**
消費者

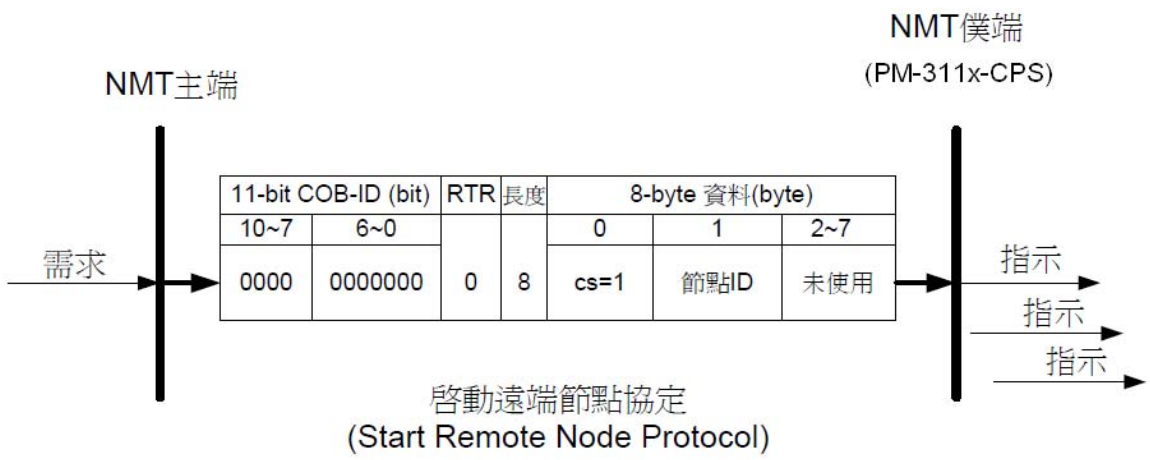
COB-ID : 0x182
L : 8
PDO-msg : 3F 55 DC 42 3F CA 66 3E

3.3 NMT 通訊集

3.3.1 模組控制協定

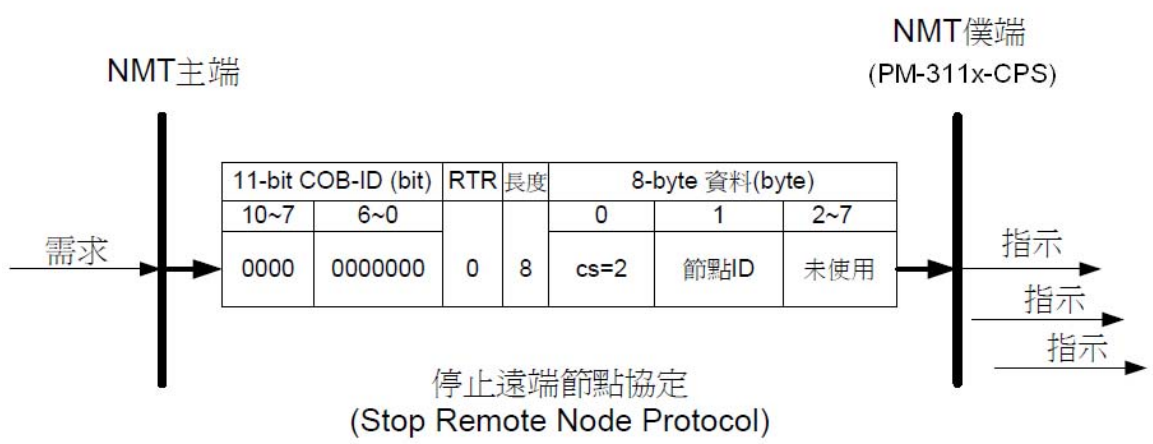
NMT主端可以利用模組控制協定來改變NMT僕端的NMT狀態，底下將詳細介紹如何改變 PM-31xx-CPS 的NMT狀態。

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



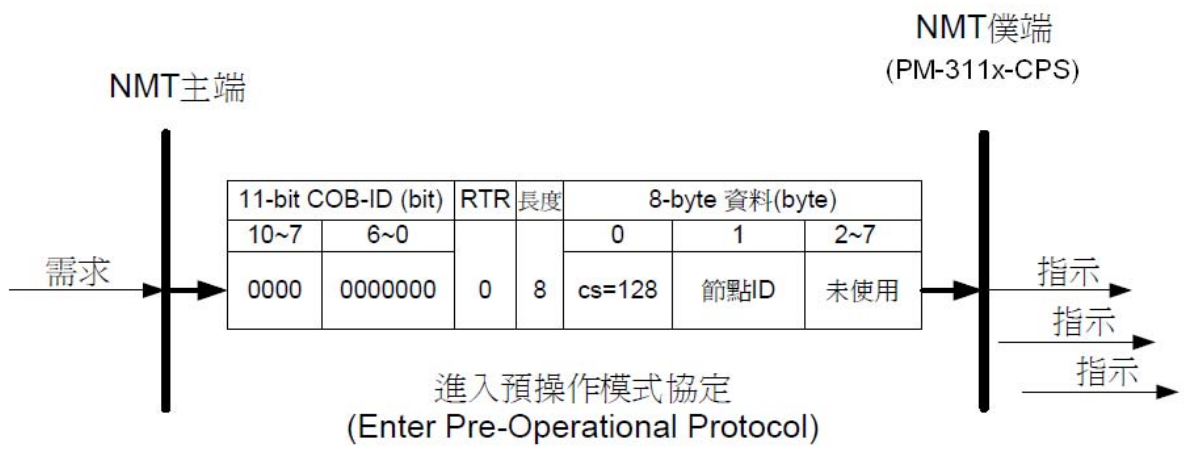
cs : NMT命令識別符
1: 啟動(start)
節點 ID : NMT僕端裝置的節點ID

<u>停止遠端節點協定 (Stop Remote Node Protocol)</u>



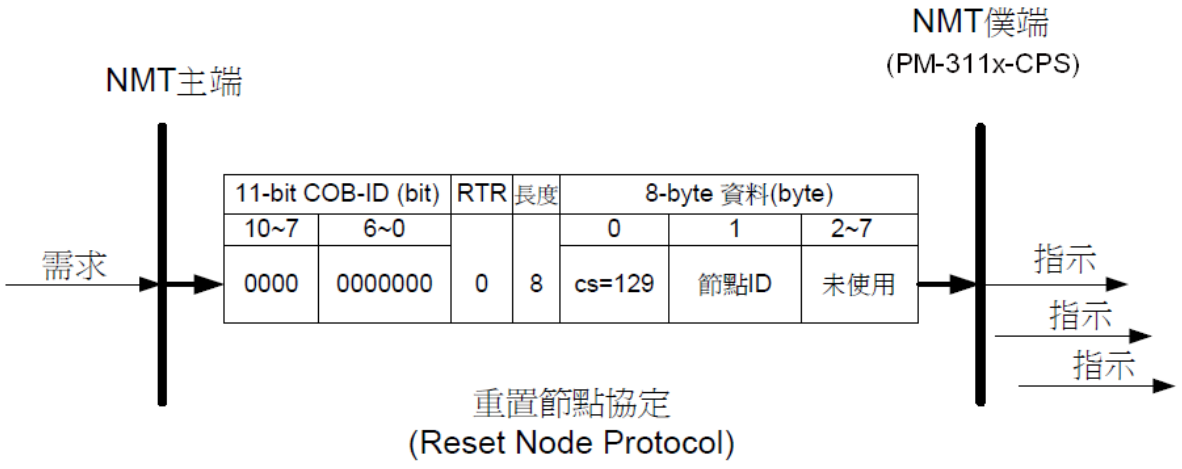
cs : NMT命令識別符
2: 停止(stop)
節點 ID : NMT僕端裝置的節點ID

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



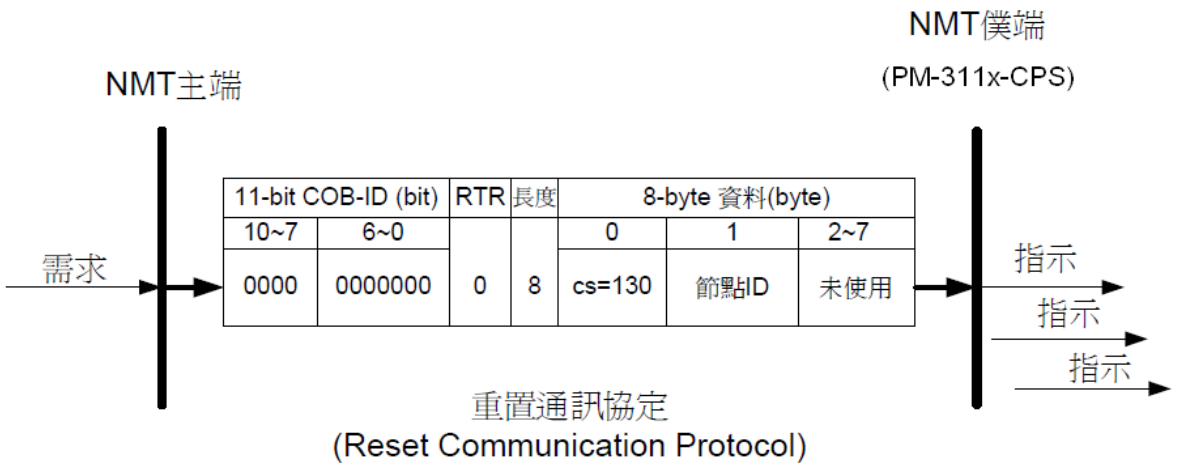
cs ：NMT命令識別符
128: 進入預操作(PRE-OPERATIONAL)狀態
節點 ID： NMT僕端裝置節點ID

重置節點協定 (Reset Node Protocol)



cs : NMT命令識別符command specified
129: 重置(Reset)節點
節點 ID : NMT僕端裝置節點ID

重置通訊協定 (Reset Communication Protocol)



11-bit COB-ID (bit)		RTR	長度	8-byte 資料(byte)		
10~7	6~0			0	1	2~7
0000	0000000	0	8	cs=130	節點ID	未使用

cs : NMT命令識別符command specified
130: 重置通訊(Reset_Communication)
節點 ID : NMT僕端裝置節點ID

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

假設PM-31xx-CPS 的節點 ID 被設定為 5 。

步驟1. 關閉 PM-31xx-CPS.

步驟2. 啟動PM-31xx-CPS。若無異常，PM-31xx-CPS在結束初始化(initialization)後，會自動進入預操作模式(Pre-Operational)。使用者可以看到PM-31xx-CPS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PM-31xx-CPS，透過NMT模組控制協定要求PM-31xx-CPS進入操作模式(operation state)。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
功能碼				節點 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

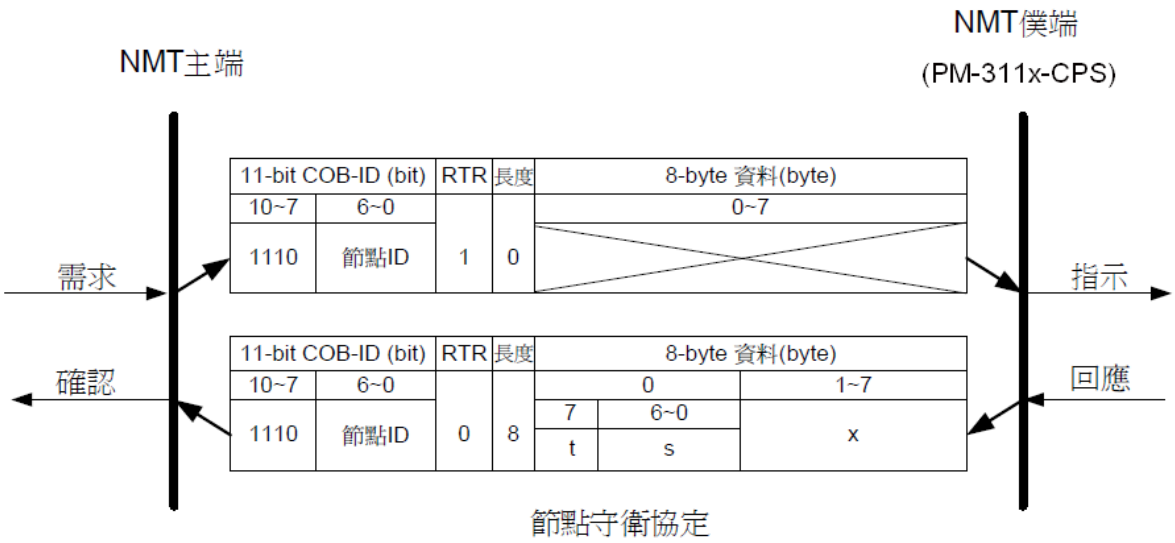
NMT 主端 → **NMT 僕端 (PM-31xx-CPS)**

cs : 1
節點 ID : 5

3.3.2 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(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。

PM-31xx-CPS 在接收到具有特定COB-ID的遠端要求訊息後，便會根據節點守衛時間開始倒數。若裝置沒有在此時間內再次收到守衛要求訊息，裝置便會對外發EMCY訊息。錯誤控制協定的細節如下所示:



節點守衛協定
(Node Guarding Protocol)

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

<u>錯誤控制協定範例</u>

步驟 1. 關閉PM-31xx-CPS的電源，再重新打開電源，PM-31xx-CPS就自動進入預操作模式。

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							
功能碼				節點 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 (PM-31xx-CPS)**

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

步驟 3. PM-31xx-CPS會回覆以下訊息結束初始SDO下載。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
功能碼				節點 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	60	0C	10	00	00	00	00	00

SDO 用戶端 ← **SDO 伺服端 (PM-31xx-CPS)**

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							
功能碼				節點 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 (PM-311x-CPS)**

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

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
功能碼				節點 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	60	0D	10	00	00	00	00	00

SDO 用戶端 ← **SDO 伺服端 (PM-311x-CPS)**

scs : 3
m : 0D 10 00

步驟 5. 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							
功能碼				節點 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	00	00	00	00	00	00	00	00

NMT 主端 → **NMT 僕端 (PM-311x-CPS)**

COB-ID : 0x701

步驟 6. 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							
功能碼				節點 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	8	7F	00	00	00	00	00	00	00

NMT 主端 → **NMT 僕端 (PM-311x-CPS)**

COB-ID : 0x701
t : 1
s : 7F
數值 7F代表PM-31xx-CPS目前的狀態為預操作狀態。

3.4 PM-31xx-CPS 的特殊功能

3.4.1 電表資料表

PM-31xx-CPS在製造商設定描述文件區域中定義了一些物件來供電表量測到的資料使用。這些物件存放於主索引0x3200到主索引0x3208之間。以PM-3114-CPS為例，詳細資料內容可以參考下表:

No.(PDO)	COB-ID	Data Length	D0~D3	D4~D7
1	0x180+節點 ID	8	kW(Kw_a)	kWh_a
2	0x280+節點 ID	8	kW(Kw_b)	kWh_b
3	0x380+節點 ID	8	kW(Kw_c)	kWh_c
4	0x480+節點 ID	8	kW(Kw_d)	kWh_d
5	---	8	Volt(V_a)	Amp(I_a)
6	---	8	Volt(V_b)	Amp(I_b)
7	---	8	Volt(V_c)	Amp(I_c)
8	---	8	Volt(V_d)	Amp(I_d)
9	---	8	kvar(kvar_a)	kVA(Kva_a)
10	---	8	kvar(kvar_b)	kVA(Kva_b)
11	---	8	kvar(kvar_c)	kVA(Kva_c)
12	---	8	kvar(kvar_d)	kVA(Kva_d)
13	---	8	PF_a	kVAh_a
14	---	8	PF_b	kVAh_b
15	---	8	PF_c	kVAh_c
16	---	8	PF_d	kVAh_d
17	---	4	kvarh_a	---
18	---	4	kvarh_b	---
19	---	4	kvarh_c	---
20	---	4	kvarh_d	---

使用者可以使用預設的前 4 組預設的 COB-ID 來讀取 PDO1 到 PDO4 的資料。以下例子假設節點 ID 設定為 1:

11-bit COB-ID (bit)											RTR	Data	8-byte 資料							
功能碼				節點 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 (PM-31xx-CPS)**

cs : 1
節點 ID : 1

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
功能碼				節點 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	0	1	1	0	00	00	00	00	00	00	00	00

PDO 生產者 → **PDO 消費者**

COB-ID : 0x181

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
功能碼				節點 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	0	1	0	8	B1	CD	8C	BC	6A	1A	F0	BF

PDO 生產者 ← **PDO 消費者**

COB-ID : 0x181
L : 8
PDO-msg : B1 CD 8C BC 6A 1A F0 BF
D0到D3的資料表示量測到Kw_a的值而D4到D7則是kWh_a的值。

如果要讀取 PDO5 到 PDO20 的值，就必須動態自訂 COB-ID 給 PDO。下面是讀取第 9 組 PDO 的例子。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
功能碼				節點 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	23	08	18	01	82	01	00	00

SDO 用戶端 → **SDO 伺服端 (PM-31xx-CPS)**

ccs : 1
n : 0
e : 1
s : 1
m : 08 18 01
d : 82 01 00 00

步驟 3. 如果沒有異常，PM-31xx-CPS會回覆以下訊息，結束通訊。

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
功能碼				節點 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	60	08	18	01	00	00	00	00

SDO 用戶端 ← **SDO 伺服端 (PM-31xx-CPS)**

scs : 3
m : 08 18 01

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
功能碼				節點 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 (PM-31xx-CPS)**

cs : 1
節點 ID : 1

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
功能碼				節點 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	0	2	1	0	00	00	00	00	00	00	00	00

PDO
生產者 → **PDO**
消費者

COB-ID : 0x182

11-bit COB-ID (bit)											RTR	資料長度	8-byte 資料							
功能碼				節點 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	0	2	0	8	E0	26	CC	3C	65	42	95	3C

PDO
Producer ← **PDO**
Consumer

COB-ID : 0x182
L : 8
PDO-msg : E0 26 CC 3C 65 42 95 3C
D0到D3 是Kvar_a的值而D4到D7是kva_a的值。