遠端串列通訊在工業控制上之應用及錯誤檢查原理分析

Application of Remote IO Communication in Industrial Control System and Analysis of the Principle of Communication Error Check

關鍵詞

EDIO (Exquisite Digital Inputs/Outputs) ASIC 數位輸出入控制晶片
Distributed IO Control 分散式輸出入控制
Serial Interface 串列介面
CRC：Cyclic Redundancy Check 循環冗餘檢查

摘要

在工業控制應用中，對於低成本、高速傳輸和高可靠度通訊品質的要求也不斷成長以滿足這些應用，目前流行的通訊一般採用串列或平行模式，而串列模式應用更為廣泛。常用的整合串列匯流排是通用非同步接收器傳輸匯流排(UART)、同步週邊設備介面(SPI)、內部積體電路(I^2C)、Microwire 及 1-Wire 等五種常用的串列匯流排。這些匯流排在速度、實體介面要求和通訊方法上都有所不同。

EDIO 832-02 這顆 3 萬 Gates 的 ASIC 為一專屬數位輸出入控制晶片，主要應用在產業機械上，只要加上簡單的週邊線路並配合遠端輸出入模組，就可以形式一個具有 832 點的控制系統，由於大幅簡化了配線及降低系統的複雜度，除了節省成本外，亦提高了系統的可靠度及維護性。

通訊的目的是要把資料即時可靠地傳送給對方，因此要求一個通訊系統傳輸資料必須可靠且快速，在數位通訊系統中可靠與快速往往是互相矛盾。為了解決可靠性，通訊系統都採用了錯誤檢測。CRC(Cyclic Redundancy Check)則被廣泛應用於資料傳輸過程中的錯誤檢測，具有很強的檢錯能力。

Abstract

In application of industrial control system, transmit the request with high reliability communication quality to the low cost , at a high speed to grow up in order to satisfy the application constantly too, the present popular communication generally adopts serial or parallel mode, and application of serial mode uses more extensive. Commonly used integrated serial buses is Universal Asynchronous Receiver/Transmitter (UART), Serial Peripheral Interface (SPI), Inter Integrated Circuit (I2C), Microwire and 1-Wire, five kinds of commonly used serial buses. The requests of these buses in the speed , physical interface and means of communication are different to some extent.

The EDIO 832-02 30,000 Gates ASIC is an exclusive pure digital input/output

control chip, mainly applied on the input/output points of ordinary industrial machinery. In application, only add simply driving and receiving Master Mode and in conjunction with remote input/output Slave Mode, it will form a 832 points input/output control system. Since it greatly simplified wiring and reduced the complexity of control system, it allows the reduction of manufacturing and maintenance cost and promotes system reliability and maintainability.

The purpose of the communication is to convey the information to the other side reliably immediately, so it must reliable and fast to require a piece of communication system to transmit the information, it is reliable and fast in the digital communication system it is often contradictory each other. In order to solve reliability, the communication system has adopted the error check. CRC (Cyclic Redundancy Check) is widely used in the error check in the transmission process of the information, have very strong examining ability of error check.

前言

現今工業控制應用中，對於低成本、高速傳輸和高可靠度通訊品質的要求也不斷成長以滿足這些應用，其結果是越來越多的處理器和控制器用不同類型的匯流排整合在一起，實現與 PC 軟體、開發系統或與網路中的其它設備進行通訊。目前流行的通訊一般採用串列或平行模式，而串列模式應用更為廣泛。在多種情況下需要使用串列介面，最為常見的是在開發或現場使用時必須與 PC 進行介面通訊。多數 PC 帶有與週邊相接的串列介面匯流排，只有少數例外，對於必須與通用電腦介面通訊而言，使用串列介面常常比 ISA 或 PCI 擴展匯流排更為方便。串列通訊只需一個 I/O 引腳便可進行通訊，而平行通訊則需要 8 個或更多，串列相較於平行的主要優點是要求的線數較少，較少的線意味著所需要的控制器接腳較少。整合在一個控制器中的平行匯流排一般需要 8 條或更多的線，線數的多少取決於設計中地址和數據的寬度，所以整合一個平行匯流排的晶片至少需要 8 個接腳來與外部元件介面，這增加了晶片的總體尺寸；相反地，使用串列匯流排可以將同樣的晶片整合在一個較小的封裝中，而在 PCB 板設計中平行匯流排需要更多的線來與其它週邊設備介面連接，因此 PCB 板面積更大、更複雜，也增加了硬體成本。

為了因應需求，機械所已開發完成的一顆專屬數位輸出入控制晶片－EDIO ASIC，這顆專用晶片只要加上簡單的週邊線路並配合遠端輸出入模組，就可以形成一個具有 832 點的控制系統。由於大幅簡化了配線及降低系統的複雜度，除了節省成本外，亦提高了系統的可靠度及維護性。本文以介紹上述專用晶片之功能為主，期望該晶片對於國內自動化工業有所助益。

在數位通訊系統中，可靠與快速往往是一對矛盾；若要求快速，則必然使得每個資料位元所佔的時間縮短、波形變窄、能量減少，從而在受到干擾後產生錯

誤的可能性增加，傳送資訊的可靠度下降；若要求可靠，則使得傳送消息的速率變慢。因此，如何合理地解決可靠及速度之間的互相矛盾，是正確設計一個通訊系統的關鍵問題之一。為了保證傳輸過程的正確性，則需要對通訊過程進行錯誤控制。錯誤控制最常用的方法是自動請求重發方式(ARQ)、向前糾錯方式(FEC)和混合糾錯方式(HEC)。在傳輸過程誤碼率(bit-error-ratio，BER)比較低時，用 FEC 方式比較理想；在傳輸過程誤碼率較高時，採用 FEC 容易出現〝亂糾〞現象。HEC 方式則是 ARQ 和 FEC 的結合。在許多數位通訊中，廣泛採用 ARQ 方式，此時的錯誤控制只需要檢錯功能。實現檢錯功能的錯誤控制方法很多，傳統的有：同位檢查(Parity Check)、總和檢查(Checksum)、水平冗餘檢查、縱向冗餘檢查…等，這些方法都是增加資料的冗餘量，將校驗碼和資料一起發送到接受端。接受端對接受到的資料進行相同校驗，再將得到的校驗碼和接受到的校驗碼比較，如果二者一致則認為傳輸正確。但這些方法都有各自的缺點，誤判的機率比較高。

循環冗餘檢查(Cyclic Redundancy Check，CRC)是一種錯誤檢查技術，它使用數學多項式來檢查資料的正確性，通常用於資料傳輸。當資料接收或使用時，在每一個固定大小或一個區塊讀取後，由框架接收者計算此訊框內容被一質因數所除後的餘數，並將此值和傳送端以相同方法產生之餘數相比較，以確認資料是否正確。由於實現簡單，檢錯能力強，被廣泛使用在各種資料校驗應用中。佔用系統資源少，用軟硬體均能實現，是進行資料傳輸錯誤檢測的一種很好的方法。

一、 串列通訊介面

在工業應用中常用的串列協議：

1. UART

UART 是一種通用串列數據匯流排，用於非同步通訊。該匯流排雙向通訊，可以實現全雙工傳送和接收。UART 首先將接收到的平行數據轉換成串列數據來傳輸，消息訊框從一個低位起始位元開始，後面是 7 個或 8 個數據位元、1 個可用的奇偶位元和 1 個或幾個高位停止位元，接收器發現起始位元時就知道數據準備發送，並嘗試與發送器時脈頻率同步。如果選擇了奇偶，UART 就在數據位元後面加上奇偶位元。奇偶位元可用來幫助錯誤校驗。在接收過程中，UART 從消息訊框中去掉起始位元和結束位元，對進來的位元組進行同位檢查，並將數據位元組從串列轉換成平行。UART 也產生額外的訊號來指示發送和接收的狀態。例如，如果產生一個奇偶錯誤，UART 就置位奇偶標誌。

數據傳輸可以首先從最低有效位元(LSB)開始。然而，有些 UART 允許靈活選擇先發送最低有效位元或最高有效位元(MSB)。微控制器中的 UART 傳送數據的速度範圍為每秒幾百位元到 1.5Mbps。例如，嵌入在 ElanSC520 微控制器中的高速 UART 通訊的速度可以高達 1.1152Mbps。UART 波特率還受發送和接收線對距離(線長度)的影響。目前，市場上有只支援非同步通訊和同時支援非同步與同步通訊

的兩種硬體適用於 UART。前者就是 UART 名字本身的含義，在摩托羅拉微控制器中被稱為串列通訊介面(SCI)；Microchip 微控制器中的通用同步非同步收發器(USART)和在富士通微控制器中的 UART 是後者的兩個典型例子。

2. 同步週邊設備介面

同步週邊設備介面(SPI)是由摩托羅拉公司開發的全雙工同步串列匯流排，該匯流排大量用在與 EEPROM、ADC、FRAM 和顯示驅動器之類的慢速週邊設備元件通訊。SPI 通訊基於主－從配置有以下 4 個訊號：

MOSI：主出/從入、MISO：主入/從出、SCK：串列時脈、SS：從屬選擇。

晶片上〝從屬選擇〞(slave-select)的接腳數決定了可連到匯流排上的元件數量。在 SPI 傳輸中，數據是同步進行發送和接收的。數據傳輸的時脈基於來自主處理器的時脈脈衝，摩托羅拉沒有定義任何通用 SPI 的時脈規格。然而，最常用的時脈設置基於時脈極性(CPOL)和時脈相位(CPHA)兩個參數，CPOL 定義 SPI 串列時脈的活動狀態，而 CPHA 定義相對於 SO-數據位元的時脈相位。CPOL 和 CPHA 的設置決定了數據取樣的時脈沿。SPI 傳輸串列數據時首先傳輸最高位元。波特率可以高達 5Mbps，具體速度大小取決於 SPI 硬體。例如，Xicor 公司的 SPI 串列元件傳輸速度能達到 5MHz。

SPI 與 UART 比較：SPI 通訊快於 UART 通訊，兩者都可以用在中等速度週邊設備的通訊中，例如非揮發性 EEPROM 記憶體。然而，SPI 更常用於 EEPROM 或數位類比變換器的通訊中。有些 UART 能支援 SPI 通訊，在這種情況下，會用一個通用 IO 作為從屬選擇接腳。

3. I^2C 匯流排

I^2C 是由飛利浦公司開發的雙線同步匯流排。像 SPI 一樣，該匯流排可用來與 EEPROM、ADC、DAC 和 LCD 這類慢速元件進行通訊。I^2C 是一個半雙工、多主匯流排，該匯流排網路有一個或幾個主控元件和很多個從元件。資訊由兩條串列線傳輸：串列數據線(SDA)和串列時脈線(SCL)。圖 1 顯示了使用兩個主控和三個從元件相連接的例子。網路中的每一個元件都預指定一個 7 位元或 10 位元的地址。飛利浦會為元件製造商分配地址，也有一個特定的地址用於高速通訊，以及一個通用呼叫地址用於與網路中所有元件的通訊。10 位元尋址的優點是允許更多的元件(高達 1024 個)佈置在網路中；然而，匯流排中元件的數目取決於匯流排的電容器量，必須限制在 400pF 以內。主控元件發起數據傳送，並提供用於通訊的時脈訊號。通訊開始於 SCL 為高電平時 SDA 由高到低的轉換，緊接著是一個 7 位元或 10 位元的從地址，一個數據方向位(R/W)，一個應答位和停止狀態。停止狀態定義為在時脈訊號為高時數據線電平由低到高的轉換。每一個數據位元組長度為 8 位元，單次傳送的位元組數並沒有限制。

由於 I^2C 是一個多主匯流排，因此可能有兩個或更多的主控元件同時試圖存取匯流排，在時脈訊號為高電平時在匯流排上置'1'的主控元件贏得匯流排仲裁。I^2C 有三種不同的執行模式：低速、快速和高速模式。在使用快速和高速模式時，可

能某個從屬元件不能像主控元件那麼快地處理數據。此時，從屬元件會將 SCL 線拉至低電平來保持匯流排，這迫使主控元件進入等待狀態，直至從屬元件準備就緒。數據傳輸首先從最高位元開始。I^2C 匯流排設計用於三種數據傳輸速度，每個都具有向下相容性：低速，數據傳輸率為 0 到 100kbps；快速，數據傳輸率可以高達 400kbps；高速，數據傳輸率可以高達 3.4Mbps。

I^2C 與 SPI 比較：I^2C 和 SPI 都能用於低速元件的通訊，而 SPI 的數據傳輸速率高於 I^2C。此外，SPI 具有一個內在地址功能，不需要設計一個額外的暫存器來測試地址，因而減少軟體和硬體的設計開銷。

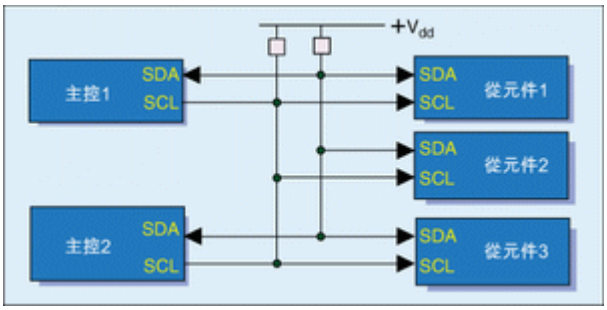

圖 1　I^2C 雙主控、三從元件配置

4. Microwire 匯流排

Microwire 是由美國國家半導體公司開發的一種三線同步介面，用於該公司的 COP8 處理器系列產品。與 SPI 相似，Microwire 是一種主／從匯流排，包括主設備發出的串列數據(SO)、主設備接收的串列數據(SI)及信號時脈(SK)等三路信號，分別對應於 SPI 的 MOSI、MISO 及 SCK。此外還有一個片選信號，其功能與 SPI 的/SS 相似。Microwire 是一種全雙工匯流排，速度可達到或超過 625Kbps(由其電容決定)。針對不同的數據需求，Microwire 元件遵循不同的協議。與基於 8 位元組的 SPI 不同，Microwire 的數據長度可變，同時還規定了一種'連續'位元流模式。

由於 Microwire 也具有多個從設備，需要多條片選線，因此它像 SPI 一樣同時兼具優缺點。有時候，SPI 元件可工作於 Microwire 匯流排，同樣，Microwire 元件也可以工作於 SPI 匯流排，儘管只能在單元件情況下。儘管在電容配置得當且速率較低時 SPI 和 Microwire 通訊距離可長達 10 英尺，但它們通常都局限於板內通訊，距離不超過六英寸。

5. 1-Wire 匯流排

Dallas Semiconductor 的 1-Wire 是一種非同步主／從式匯流排，沒有用於多個主設備的協議。與 I_2C 相似，1-Wire 採用半雙工通訊，在單一連線上採用一種開漏拓樸結構進行雙向數據傳輸。不過，1-Wire 匯流排的數據線也可以向從設備傳輸功率，儘管比較有限。1-Wire 的最高速率僅達 16Kbps，但在升壓電阻配置得當時，傳輸距離可達 1000 英尺。

二、 EDIO ASIC－數位輸出入控制晶片

圖 2 EDIO ASIC-數位輸出入控制晶片

EDIO 功能說明：

EDIO 的全名是 Exquisite Digital Inputs／Outputs。相較於傳統集中式控制平行配線的方式，EDIO 採用串列式介面，將遠端欲受控的輸出入點資料，透過同步雙工方式的特殊協定格式和控制器溝通。如此一來多達 768 點的資料，在控制器端只須要少數的通訊配線即可完成傳輸。除減輕系統硬體的負擔外，每 128 點(64 輸入，64 輸出)即有獨立的遠端資料擷取模組來負責資料的即時更新及傳送，(一顆操作在主動模式控制器側的 EDIO chip，可以控制最多 6 個遠端模組上操作在被動模式的 EDIO chips。)進而朝向分散式控制的目標努力。

EDIO 是一顆 30000 Gates 的 ASIC，以 160 pin PQFP 包裝。基本上它是我們 EPCIO 的精簡版本，使用相同的輸出入定址及傳輸協定，故可以有交互的替代性。(即 EPCIO 的主動模式可以控制 EDIO 的被動模式，反之 EDIO 的主動模式，亦可以控制 EPCIO 的被動模式。)，EDIO 本身即為一具有雙重操作模式的晶片組，透過外部接腳直接設定，就可表現主動(Master)或被動(Slave)兩種應用。表一是 EDIO 的主要功能及規格：

表一 EDIO 主要規格

匯流排介面	ISA Interface Data Length : 16bits IO Port Addresses : 32 occupied Base Address : Hardware Setup Wait State : Software Setup
工作時脈	8MHz ~ 40MHz
中斷	Total 42 Interrupt Channels Including : Each Remote Module 4 ch (*6) , Transmission Error 2 ch , Local Inputs 15 ch , Timer 1 ch
遠端輸出入	64 In 64 Out per Slave , Each master EDIO can control 2 independent remote IOs sets, (Each set can have 3 slaves)

	Maximum Inputs 384 (64*6) Maximum Outputs 384 (64*6)
近端輸出入	64 Local IOs for EDIO Master Mode : 32 Dedicated Inputs 32 (4*8) Programmable Inputs or Outputs
計時器	24 bits Timer 1 set
看門狗	16 bits Timer 1 set Programmable Reset Duration (24 bits)
串列傳輸	80kHz ~ 2.5MHz , Programmable Transmission Clock Data Error Check : CRC (Cyclic Redundancy Check) Protocol : Proprietary Definition

再進一步說明如下：

匯流排介面：

EDIO 本身已內建 ISA 的介面，故很容易地即可用來設計成 PC-BASED 控制器的附加功能卡。另外透過適當的橋接 IC，也可將 EDIO 應用在 PCI 介面上。輸出入定址的部分，為了節省位址空間，在實用上是以切換頁(Page)的方式執行，表二是列舉所有位址的輸出入暫存器。內部的詳細定義這裡並不討論，較進階的使用者,可以呼叫我們現有提供的驅動程式(Device Driver)來控制 EDIO 的各項功能。

表二：輸出入暫存器(共 4 個 Pages)

Page 0：Bus interface control

P.A.	Read		Write
0	Interrupt Index		Software Reset
1			ISA Bus Wait State
2			Interrupt Channel & period
3			Interrupt Mode
4			Software Clear Interrupt
5~14			
15	Read the present R/W page		Write the next R/W page register

Page 5：First set remote Digital IO

P.A.	Read		Write
0	Set 1 Remote slave 0 input of channel 15~0		Set 1 Remote slave 0 output of channel 15~0
1	Set 1 Remote slave 0 input of channel 31~16		Set 1 Remote slave 0 output of channel 31~16
2	Set 1 Remote slave 0 input of channel 47~32		Set 1 Remote slave 0 output of channel 47~32
3	Set 1 Remote slave 0 input of channel 63~48		Set 1 Remote slave 0 output of channel 63~48
4	Set 1 Remote slave 1 input of channel 15~0		Set 1 Remote slave 1 output of channel 15~0
5	Set 1 Remote slave 1 input of channel 31~16		Set 1 Remote slave 1 output of channel 31~16
6	Set 1 Remote slave 1 input of channel 47~32		Set 1 Remote slave 1 output of channel 47~32
7	Set 1 Remote slave 1 input of channel 63~48		Set 1 Remote slave 1 output of channel 63~48
8	Set 1 Remote slave 2 input of channel 15~0		Set 1 Remote slave 2 output of channel 15~0
9	Set 1 Remote slave 2 input of channel 31~16		Set 1 Remote slave 2 output of channel 31~16
10	Set 1 Remote slave 2 input of channel 47~32		Set 1 Remote slave 2 output of channel 47~32
11	Set 1 Remote slave 2 input of channel 63~48		Set 1 Remote slave 2 output of channel 63~48

12	Set 1 Remote IO status		Set 1 Remote IO clock divider & Slave 2 IO Interrupt Control
13	Set 1 Remote IO Interruption latch		Set 1 Remote IO maximum error、slave enable & Slave 1 IO Interrupt Control
14			Slave 0 IO Interrupt Control & Transmission Failure interruption control
15	Read the present R/W page		Write the next R/W page register

Page 6：Second set remote Digital IO

P.A.	Read		Write
0	Set 2 Remote slave 0 input of channel 15~0		Set 2 Remote slave 0 output of channel 15~0
1	Set 2 Remote slave 0 input of channel 31~16		Set 2 Remote slave 0 output of channel 31~16
2	Set 2 Remote slave 0 input of channel 47~32		Set 2 Remote slave 0 output of channel 47~32
3	Set 2 Remote slave 0 input of channel 63~48		Set 2 Remote slave 0 output of channel 63~48
4	Set 2 Remote slave 1 input of channel 15~0		Set 2 Remote slave 1 output of channel 15~0
5	Set 2 Remote slave 1 input of channel 31~16		Set 2 Remote slave 1 output of channel 31~16
6	Set 2 Remote slave 1 input of channel 47~32		Set 2 Remote slave 1 output of channel 47~32
7	Set 2 Remote slave 1 input of channel 63~48		Set 2 Remote slave 1 output of channel 63~48
8	Set 2 Remote slave 2 input of channel 15~0		Set 2 Remote slave 2 output of channel 15~0
9	Set 2 Remote slave 2 input of channel 31~16		Set 2 Remote slave 2 output of channel 31~16
10	Set 2 Remote slave 2 input of channel 47~32		Set 2 Remote slave 2 output of channel 47~32
11	Set 2 Remote slave 2 input of channel 63~48		Set 2 Remote slave 2 output of channel 63~48
12	Set 2 Remote IO status		Set 2 Remote IO clock divider & Slave 2 IO Interrupt Control
13	Set 2 Remote IO Interruption latch		Set 2 Remote IO maximum error、 slave enable & Slave 1 IO Interrupt Control
14			Slave 0 IO Interrupt Control & Transmission Failure interruption control
15	Read the present R/W page		Write the next R/W page register

Page 8：LIO Control

P.A.	Read		Write
0	Local Digital IO LDIO[15:0]		Local Digital Output LDO[15:0]
1	Local Digital IO LDIO[31:16]		Local Digital Output LDO[31:16]
2	Local Double Function Digital Input DFI[15:0]		Enable Local Digital Output, Timer and Watch Dog Enable
3	Local Double Function Digital Input DFI[31:16]		Local Digital Input Interrupt Control
4	Local Digital Input & Timer Interrupt Latch		Local Double Function Input & Timer Interrupt Control
5			Timer Value low word
6			Timer Value high word
7			Watch Dog Reset duration low word
8			Watch Dog Reset duration high word
9			Watch Dog Timer value
10			Refresh Watch Dog
11~14			
15	Read the present R/W page		Write the next R/W page register

中斷：

EDIO 本身僅佔用 1 個系統的中斷輸入(可軟體設定)，但是中斷的來源卻有多達 42 個 Channels，故可藉由軟體檢查中斷指標(Index)的方式，確定中斷發生時中斷向量的位置。其中遠端模組(每一個 Slave)提供 4 個輸入點的中斷 channels，近端(Master)提供多達 15 個輸入點的中斷 channels，上述的中斷點均可規劃成正或負緣或正負緣同時觸發。另外內部計時器提供一個可設定週期的中斷 channel，可做為軟體固定時間的參考訊號。最後是 EDIO 所控制兩個 Sets(共 6 個 Slaves)的串列傳

輸發生失誤時的中斷 channels，使用者可規劃最大容忍的失敗次數。

遠端輸出入：

圖 3 為 EDIO 模組系統連接圖，Host PC 透過 ISA/PCI Bus 與 EPCIO/EDIO 控制母板做資料的存取，包括輸入點的讀取，輸出點的控制，及中斷訊號處理。總共 768 個輸出入點，分散在遠端的六個被動模組上，模組的設計可因使用者的應用而異，下圖二是我們目前使用的遠端模組 EDIO-S003。(操作在被動模式的 EDIO 最大可控制 64IN，64 OUT)。

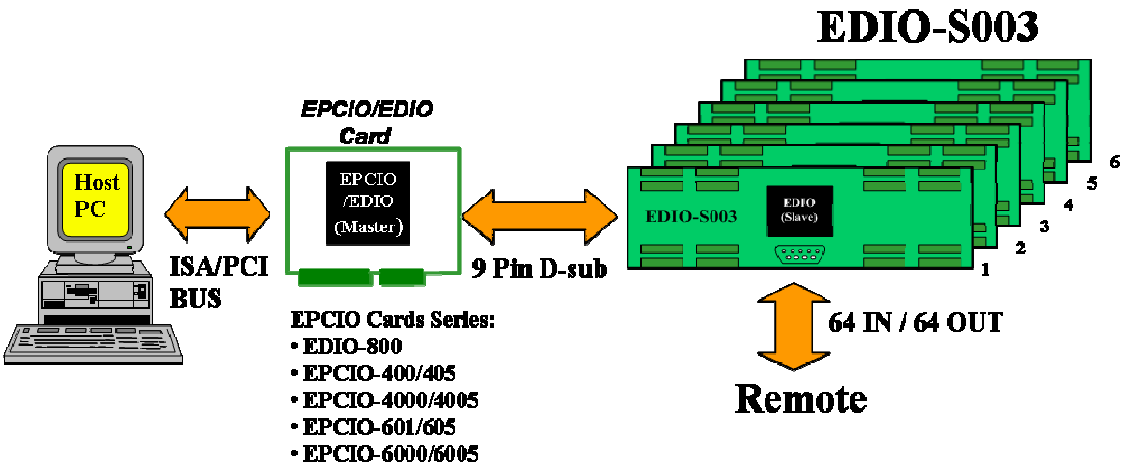

圖 3　EDIO 模組系統連接圖

圖上的 EPCIO/EDIO ASIC(Master Mode)負責在固定的時間內把遠端 IO 點的狀態做讀取與更新的工作，其中輸出部分先經過驅動電路轉成差動訊號後由 9Pin 連接器輸出至遠端模組，輸入部分則接收由連接器出入之差動訊號，經光偶合器後送至 ASIC 輸入。

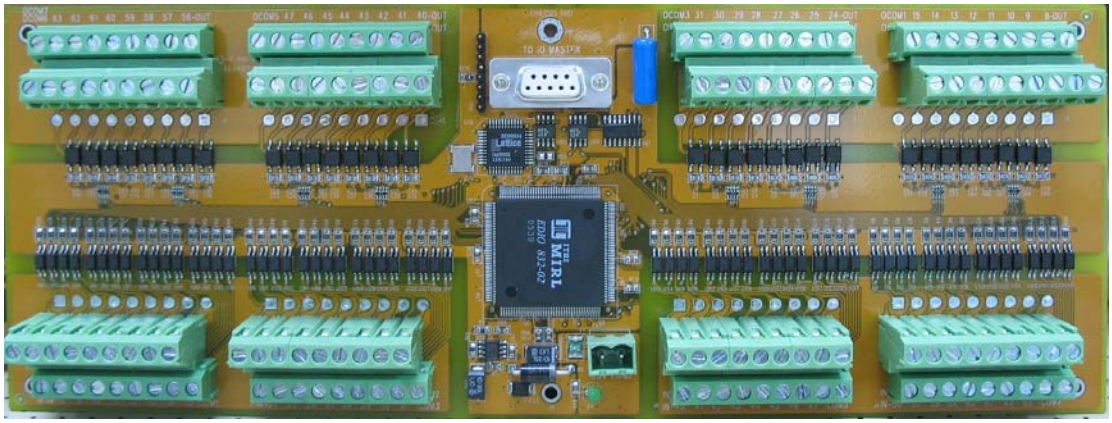

圖 4　遠端模組 EDIO-S003 (64IN/64OUT)

圖 4 為 EDIO-S003 遠端模組，EDIO ASIC(Slave Mode)負責接受來自於 EPCIO/EDIO 控制板的資料並輸出及讀回實際 IO 接點狀態。整個架構採省配線化，遠端控制模式，所有控制電路整合入 ASIC 內部，以提高產品可靠度及穩定性。該模組透過一 D 型 9 pin 的接頭，和控制器上設於主動模式的 EDIO 溝通，使用者

除了節省控制器的配線成本外，各個輸出入模組可以就近在各個致動器(Actuator)或感測器(Sensor)旁作控制，系統的穩定性得以提昇。而且可有效利用串列介面在長距離傳輸的經濟性，配合適當的電磁干擾防制措施，我們已得到如下的傳輸距離表現：(見表三)

表三　遠端輸出入模組傳輸距離

1 Slave	Clock Rate	Distance	Data Update Time
64IN/64OUT	650KHz	100 meters	150 us
64IN/64OUT	3.4MHz	15 meters	30 us
64IN/64OUT	4MHz	1.5 meters	25 us

測試條件：使用線材編號

E146924 AWM 2464 VW-1 80C 300V 24AWG

LL101096 CSA AWM A/B I/II 80℃ 300V FT1 24AWG GEI TAI

具有隔離網的 9 芯 RS232 傳輸線。

近端輸出入：

為了充分利用操作在主動模式的 EDIO 晶片，我們把大部份的接腳(除了匯流排及串列介面要用到的外)規劃為一般的輸出入點用，即所謂的 Local IO。總共有 64 點，其中 32 點只作輸入點用，另外的 32 點以每 4 點為一組，可規劃為輸出或輸入用。對於在控制器附近的受控元件，可以提供系統更大的應用彈性，而且這些輸出入點因為透過 EDIO 直接由 ISA 和 PC 的 CPU 作存取，理論上可以實現高速控制(例如 500 nsec update time)，唯仍須考慮輸出入介面整體設計的響應頻寬。

計時器／看門狗：

以 40MHz 的晶片工作時脈來估，下表是使用者能規劃的設定範圍(表四)：

表四　計時器/看門狗 設定範圍 (40 MHz clock)

Function	Minimum Period	Maximum Period	Step
Timer (24 bits)	50 ns	420 msec	25 ns
Watch Dog Strobe(16 bits)	25 ns	27525 sec	Depends on timer period setup
Watch Dog Reset Duration (24 bits)	25 ns	420 ms	25 ns

其中計時器主要負責產生看門狗計數器所使用的固定時基，並且允許發生一週期性的中斷，而看門狗的用處，是可以提供系統一個自我保護的機制，在規劃的 Strobe 時間內，必須對 EDIO 作一輪詢，以便確保系統仍處於清醒的狀態，否則 EDIO 便會發出重置的信號。當使用此重置信號於外部硬體週邊時，可配合 EDIO 原有提供的重置暫存器功能，上層便可以有絕對的重置權，以求系統的強韌性。

串列傳輸：

EDIO 內建有兩組獨立的串列介面給兩個 Sets 的六個遠端輸出入模組使用，對

每一個被動模式 EDIO 而言，它須要 Transmission Clock，Chip Select，Data In，Data Out，等四條信號線作溝通，整體的通訊協定格式可以下圖表示(圖 5)：

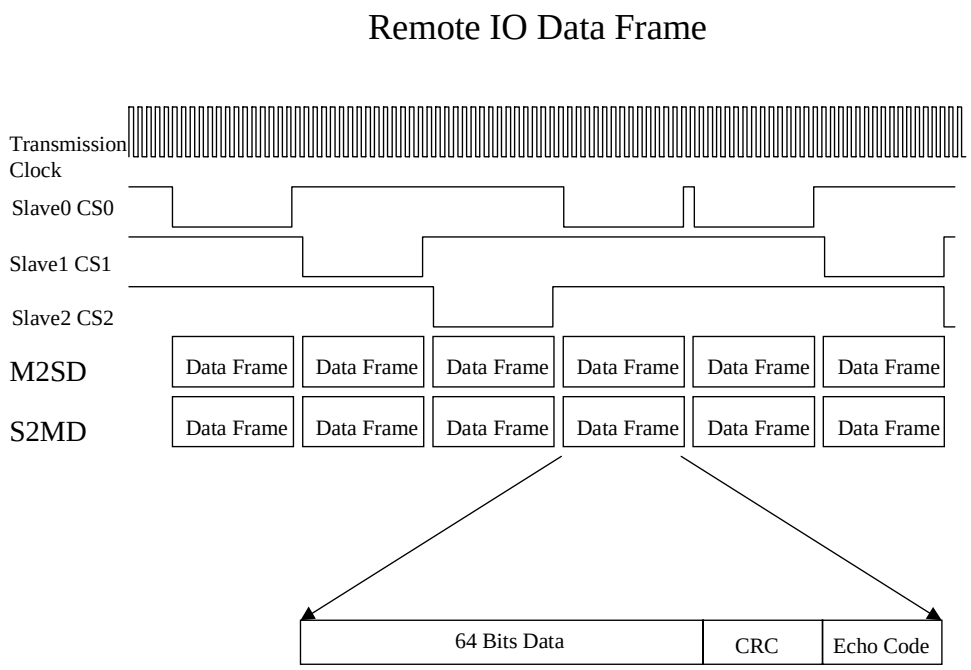

圖 5　串列介面傳輸協定

由圖上得知，Slave 0，1，2 是共用 Data In，Out 及 Clock 線的，而由 Chip Select 0，1，2 來做解碼。隨著傳輸 Clock 頻率動作，Data In 及 Data Out 同時被 Shift 進入及 Shift 移出兩邊的 EDIO Chip。接著 64 bits 的資料碼後，緊跟著的是 16 bits 的 CRC 碼以及 16 bits 的 Echo 碼。這些附加碼是為了偵測傳送數據的錯誤而定的。

CRC 是 Cyclic Redundancy Check 的縮寫，即循環冗餘檢查，基本上 CRC 將資料視為一訊息多項式，將此式除以預先決定之生成多項式後，得到的結果就是檢核碼。將此碼附加在資料後一起傳送，而接收端也利用相同之生成多項式進行除算確認，如果餘數為零就可判斷資料沒有問題。常見的錯誤檢查法除了 CRC 外，就是同位檢查法(Parity Check)，使用上的選擇以錯誤漏檢率大小為考量，而後者也是我們之前 EIO-80 板上所使用的檢查機制。錯誤偵測的最終目的乃是要校正傳輸的錯誤，目前 EDIO 的作法是要求信號重送，而重送的次數會累積起來，和使用者預設的最大容忍次數作比較，以便產生傳輸失敗的中斷信號。

介面電路：

這裡再簡單介紹有關串列介面及輸出入介面的電路，以便於日後的溝通。圖 6 是我們之前所討論 EDIO 串列介面所使用的標準線路。因為採用光耦合器隔離，它所造成額外的時間延遲對於規劃傳輸 Clock 的快慢有很大關係。

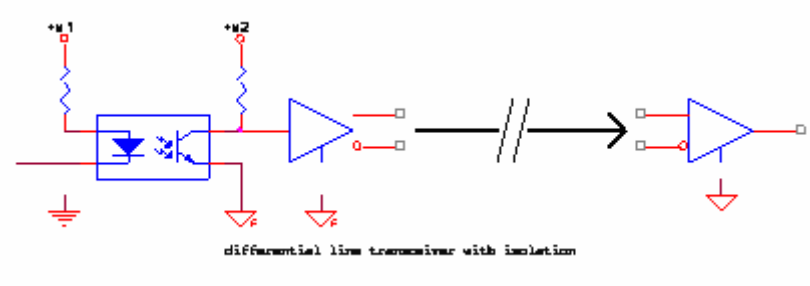

圖 6　介面電路說明

應用說明：

早期的 EIO-80 板是將上述類似的功能以 FPGA 的 IC 來實現，因為受控點局限在 40 In，40 Out，對於遠端輸出入模組的應用貢獻不大。是故將主動模式的 FPGA IC 和被動模式的 FPGA IC 設計於同一片板上，利用中間串列介面的少數信號線來達成光耦合隔離的效果。試想在一般的情形下，這片板子上就必須要使用多達 80 顆的光耦合器 IC，大大增加了硬體設計的負擔。

至於 EDIO 的出現，我們可以看看下圖 7，典型產業機台的控制器，從人機介面，機台感測器，到運動控制用馬達及驅動器的溝通，藉由遠端輸出入模組的設計，一切變得非常的簡潔。對於控制器本身而言，只須要一條串列傳輸用的電纜即可做到。

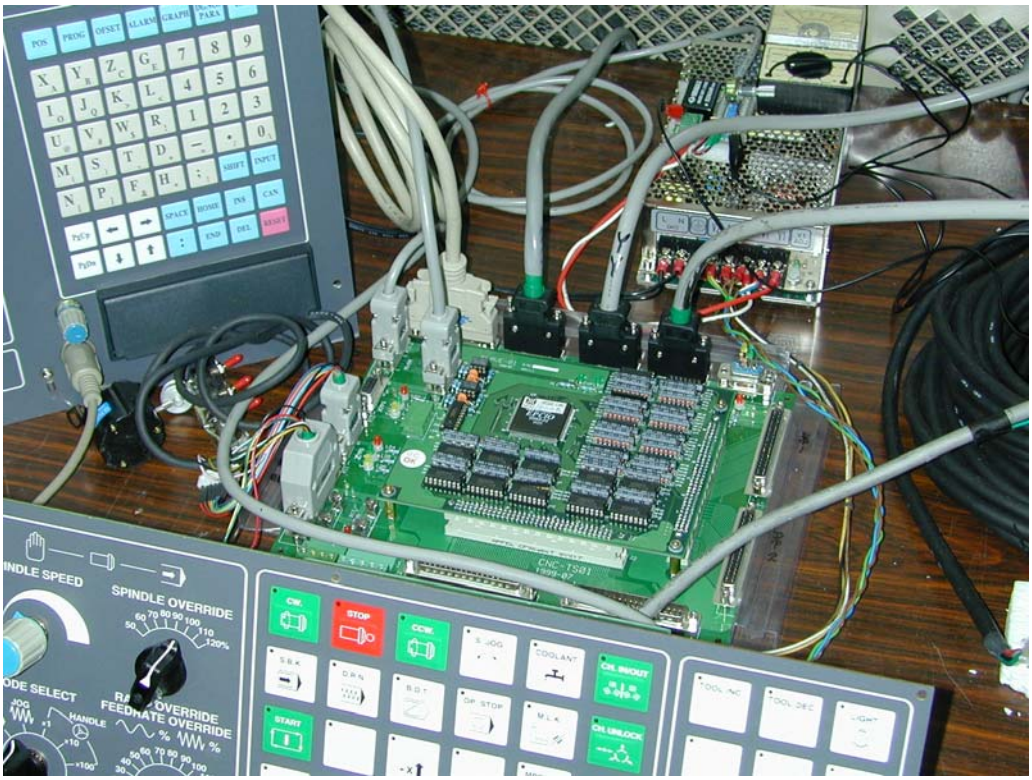

圖 7　遠端輸出入模組的應用－車銑床控制器(由數控部提供)

三、 通訊錯誤檢測原理

在資料通訊中，接收端需要檢測在傳輸過程中是否發生錯誤，常用的技術有同位檢查，總和檢查和 CRC。它們都是發送端對消息按照某種演算法計算出校驗碼，然後將校驗碼和消息一起發送到接收端。接收端對接收到的消息按照相同演算法得出校驗碼，再與接收到的校驗碼比較，以判斷接收到消息是否正確。

同位檢查只需要 1 位校驗碼，其計算方法也很簡單。以奇同位檢查為例，發送端只需要對所有消息位元進行異或運算，得出的值如果是 0，則校驗碼為 1，否則為 0；接收端可以對消息進行相同計算，然後比較校驗碼，也可以對消息連同校驗碼一起計算，若值是 0 則有錯誤，否則校驗通過。通常說同位檢查可以檢測出 1 位錯誤，實際上它可以檢測出任何奇數位錯誤。

總和檢查的思想也很簡單，將傳輸的消息當成 8 位元(或 16/32 位元)整數的序列，將這些整數加起來而得出校驗碼，該校驗碼也叫校驗和。校驗和被用在 IP 協議中，按照 16 位元整數運算，而且其最高有效位元(Most Significant Bit，MSB)的進位被加到結果中。顯然，同位檢查和總和檢查都有明顯的不足。同位檢查不能檢測出偶數位錯誤。對於總和檢查，如果整數序列中有兩個整數出錯，一個增加了一定的值，另一個減小了相同的值，這種錯誤就檢測不出來。

1. CRC 演算法的基本原理

CRC 演算法的是以 GF(2)(2 元素伽羅瓦域)多項式算術為數學基礎的，聽起來很恐怖，但實際上它的主要特點和運算規則是很好理解的。GF(2)多項式中只有一個變數 x，其係數也只有 0 和 1，如：

1*x^7 + 0*x^6 + 1*x^5 + 0*x^4 + 0*x^3 + 1*x^2 +1*x^1 + 1*x^0

即：

x^7 + x^5 + x^2 + x + 1　　(x^n 表示 x 的 n 次冪)

GF(2)多項式中的加減用模 2 算術執行對應項上係數的加減，模 2 就是加減時不考慮進位和借位，即：

0 + 0 = 0　　0 − 0 = 0

0 + 1 = 1　　0 − 1 = 1

1 + 0 = 1　　1 − 0 = 1

1 + 1 = 0　　1 − 1 = 0

顯然，加和減是一樣的效果(故在 GF(2)多項式中一般不出現"-"號)，都等同於異或運算。例如 P1 = x^3 + x^2 + 1，P2 = x^3 + x^1 + 1，則 P1 + P2 為：

```
  x^3 + x^2     + 1
+ x^3           + x + 1
-----------------------
        x^2 + x
```

GF(2)多項式乘法和一般多項式乘法基本一樣，只是在各項相加的時候，按照模 2 算術進行，例如 P1 * P2 為：

```
    (x^3 + x^2 + 1)(x^3 + x^1 + 1)
  = (x^6 + x^4 + x^3+ x^5 + x^3 + x^2+ x^3 + x + 1)
  =  x^6 + x^5 + x^4 + x^3 + x^2 + x + 1
```

GF(2)多項式除法也和一般多項式除法基本一樣，只是在各項相減的時候，按照模 2 算術進行，例如 P3 = x^7 + x^6 + x^5 + x^2 + x，P3 / P2 為：

```
                  x^4 + x^3           + 1
                  ----------------------------------------
x^3 + x + 1 )x^7 + x^6 + x^5 +              x^2 + x
              x^7       + x^5 + x^4
              --------------------
                    x^6       + x^4
                    x^6       + x^4 + x^3
                    --------------------
                                  x^3 + x^2 + x
                                  x^3       + x + 1
                                  ----------------
                                        x^2     + 1
```

CRC 演算法將長度為 m 位元的消息對應一個 GF(2)多項式 M，比如對於 8 位元消息 11100110，如果先傳輸 MSB，則它對應的多項式為 x^7 + x^6 + x^5 + x^2 + x。發送端和接收端約定一個次數為 r 的 GF(2)多項式 G，稱為生成多項式，比如 x^3 + x + 1，r = 3。在消息後面加上 r 個 0 對應的多項式為 M'，顯然有 M' = Mx^r。用 M'除以 G 將得到一個次數等於或小於 r - 1 的餘數多項式 R，其對應的 r 位數值則為校驗碼。如下所示：

```
          11001100
        ------------
1011 )11100110000
       1011.......
       ----.......
        1010......
        1011......
        ----......
           1110...
           1011...
           ----...
```

1010. .
1011. .
----
100 <---校驗碼

發送端將 m 位元消息連同 r 位元校驗碼(也就是 M' + R)一起發送出去，接收端按同樣的方法算出收到的 m 位元消息的校驗碼，再與收到的校驗碼比較。接收端也可以用收到的全部 m + r 位元除以生成多項式，再判斷餘數是否為 0。這是因為，M' + R = (QG + R) + R = QG，這裏 Q 是商。顯然，它也可以像發送端一樣，在全部 m + r 後再增加 r 個 0，再除以生成多項式，如果沒有錯誤發生，餘數仍然為 0。

2. 生成多項式的選擇

很明顯，不同的生成多項式，其檢錯能力是不同的。如何選擇一個好的生成多項式需要一定的數學理論，這裏只從一些側面作些分析。顯然，要使用 r 位校驗碼，生成多項式的次數應為 r。生成多項式應該包含項'1'，否則校驗碼的 LSB(Least Significant Bit)將始終為 0。如果消息(包括校驗碼)T 在傳輸過程中產生了錯誤，則接收端收到的消息可以表示為 T + E；若 E 不能被生成多項式 G 除盡，則該錯誤可以被檢測出。考慮以下幾種情況：

(1) 1 位錯誤，即 E = x^n = 100...00，n >= 0。只要 G 至少有 2 位 1，E 就不能被 G 除盡。這是因為 Gx^k 相當於將 G 左移 k 位，對任意多項式 Q，QG 相當於將多個不同的 G 的左移相加。如果 G 至少有兩位 1，它的多個不同的左移相加結果至少有兩位 1。

(2) 奇數位錯誤，只要 G 含有因數 F = x + 1，E 就不能被 G 除盡。這是因為 QG =Q'F，由 1)的分析，F 的多個不同的左移相加結果 1 的位數必然是偶數。

(3) 爆炸性錯誤，即 E = (x^n + ... + 1)x^m = 1...100...00，n >= 1，m >= 0，顯然只要 G 包含項"1"，且次數大於 n，就不能除盡 E。

(4) 2 位錯誤，即 E = (x^n + 1)x^m = 100...00100...00，n >= 0。設 x^n + 1 = QG + R，則 E = QGx^m + Rx^m，由 3)可知 E 能被 G 除盡當且僅當 R 為 0。因此只需分析 x^n + 1，根據[5]，對於次數 r，總存在一個生成多項式 G，使得 n 最小為 2^r - 1 時，才能除盡 x^n + 1。稱該生成多項式是原始的(primitive)，它提供了在該次數上檢測 2 位錯誤的最高能力，因為當 n = 2^r - 1 時，x^n + 1 能被任何 r 次多項式除盡。[5]同時指出，原始生成多項式是不可約分的，但不可約分的的多項式並不一定是原始的，因此對於某些奇數位錯誤，原始生成多項式是檢測不出來的。

3. CRC 多項式規範

表五為常用 CRC（按照 ITU-IEEE 規範），表格中略去了「初始值(Initial value)」、「反射值(Reflected value)」以及「最終異或值(Final XOR value)」。對於一些複雜的校驗和來說這些十六進位數值是很重要的，如 CRC-32 以及 CRC-64。

通常小於 CRC-16 的 CRC 不需要使用這些值。通常可以通過改變這些值來得到各自不同的校驗和，但是校驗和演算法機制並沒有變化。

表五　CRC polynomials in common use (in ITU-IEEE syntax)

Name	Polynomial	Representations: Normal or Reverse (Normal of Reciprocal)
CRC-1	x + 1 (use: hardware; also known as parity bit)	0x1 or 0x1 (0x1)
CRC-5-CCITT	$x^5 + x^3 + x + 1$ (ITU G.704 standard)	0x0B or 0x1A (0x15)
CRC-5-USB	$x^5 + x^2 + 1$ (use: USB token packets)	0x05 or 0x14 (0x9)
CRC-7	$x^7 + x^3 + 1$ (use: telecom systems, MMC)	0x09 or 0x48 (0x11)
CRC-8-ATM	$x^8 + x^2 + x + 1$ (use: ATM HEC)	0x07 or 0xE0 (0xC1)
CRC-8-CCITT	$x^8 + x^7 + x^3 + x^2 + 1$ (use: 1-Wire bus)	0x8D or 0xB1 (0x63)
CRC-8-Dallas/Maxim	$x^8 + x^5 + x^4 + 1$ (use: 1-Wire bus)	0x31 or 0x8C (0x19)
CRC-8	$x^8 + x^7 + x^6 + x^4 + x^2 + 1$	0xD5 or 0xAB (0x57)
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x + 1$	0x233 or 0x331 (0x263)
CRC-12	$x^{12} + x^{11} + x^3 + x^2 + x + 1$ (use: telecom systems)	0x80F or 0xF01 (0xE03)
CRC-15-CAN	$x^{15} + x^{14} + x^{10} + x^8 + x^7 + x^4 + x^3 + 1$	0x4599 or 0x4CD1 (0x19A3)
CRC-16-Fletcher	Not a CRC; see Fletcher's checksum	Used in Adler-32 A & B CRCs
CRC-16-CCITT	$x^{16} + x^{12} + x^5 + 1$ (XMODEM,X.25, V.41, Bluetooth, PPP, IrDA; known as "CRC-CCITT")	0x1021 or 0x8408 (0x0811)

CRC-16-IBM	$x^{16} + x^{15} + x^2 + 1$ (USB, many others; also known as "CRC-16")	0x8005 or 0xA001 (0x4003)
CRC-24-Radix-64	$x^{24} + x^{23} + x^{18} + x^{17} + x^{14} + x^{11} + x^{10} + x^7 + x^6 + x^5 + x^4 + x^3 + x + 1$	0x864CFB or 0xDF3261 (0xBE64C3)
CRC-32-Adler	Not a CRC; see Adler-32	See Adler-32
CRC-32-MPEG2	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$	0x04C11DB7 or 0xEDB88320 (0xDB710641) Also used in IEEE 802.3
CRC-32-IEEE 802.3	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ (V.42)	0x04C11DB7 or 0xEDB88320 (0xDB710641)
CRC-32C (Castagnoli)[1]	$x^{32} + x^{28} + x^{27} + x^{26} + x^{25} + x^{23} + x^{22} + x^{20} + x^{19} + x^{18} + x^{14} + x^{13} + x^{11} + x^{10} + x^9 + x^8 + x^6 + 1$	0x1EDC6F41 or 0x82F63B78 (0x05EC76F1)
CRC-64-ISO	$x^{64} + x^4 + x^3 + x + 1$ (use: ISO 3309)	0x000000000000001B or 0xD800000000000000 (0xB000000000000001)
CRC-64-ECMA-182	$x^{64} + x^{62} + x^{57} + x^{55} + x^{54} + x^{53} + x^{52} + x^{47} + x^{46} + x^{45} + x^{40} + x^{39} + x^{38} + x^{37} + x^{35} + x^{33} + x^{32} + x^{31} + x^{29} + x^{27} + x^{24} + x^{23} + x^{22} + x^{21} + x^{19} + x^{17} + x^{13} + x^{12} + x^{10} + x^9 + x^7 + x^4 + x + 1$ (as described in ECMA-182 p.63)	0x42F0E1EBA9EA3693 or 0xC96C5795D7870F42 (0x92D8AF2BAF0E1E85)
CRC-128	IEEE-ITU Standard. Use superseded by hashes MD5 & SHA-1	
CRC-160	IEEE-ITU Standard. Use superseded by hashes MD5 & SHA-1	

採用 CRC 檢查時，發送端和接收端用同一個生成多項式 G(x)，並且 G(x)的首位和最後一位的係數必須為 1。CRC 的處理方法是：發送端以 G(x)去除 T(x)，得到餘數作為 CRC 校驗碼；檢查時，以計算的校正結果是否為 0 為依據，判斷資料是否有錯誤。CRC 校驗可以百分之百地檢測出所有奇數個隨機錯誤和長度小於等於 k（k 為 G(x)的階數）的突發錯誤。所以 CRC 的生成多項式的階數越高，那麼誤判的機率就越小。

從 CRC 的編碼規則可以看出，CRC 編碼實際上是將代發送的 m 位元二進位多項式 T(x)轉換成了可以被 G(x)除盡的 m+r 位元二進位多項式，所以解碼時可以用接受到的資料去除 G(x)，如果餘數為零，則表示傳輸過程沒有錯誤；如果餘數不為零，則在傳輸過程中肯定存在錯誤。許多 CRC 的硬體解碼電路就是按這種方式進行檢錯的。同時 可以看做是由 T(x)和 CRC 校驗碼的組合，所以解碼時將接收到的二進位資料去掉尾部的 r 位元資料，得到的就是原始資料。

四、 結語

不同的應用在選擇匯流排的標準方面並不同，下面是一些進行匯流排選擇的通用參考準則：

1. 評估使用不同串列匯流排在網路上連接各種元件的系統成本。
2. 在效率、速度和可靠性方面確定對你最重要的性能。
3. 確定在網路上將連接多少元件，以及匯流排將可能具有的電容器量。有些串列匯流排對連在網路上的元件數目有限制。
4. 注意元件間的距離，有些串列匯流排只支援短距離通訊。

如上所述，面對大量串列通訊匯流排標準。在選擇一種串列匯流排時，不應只考慮到滿足產品的當前需要，而且還應考慮到是否能用於產品的整個生命周期才是長遠的考量。

目前市面已有許多專屬輸出入控制晶片，因此我們更要以戰戰兢兢的態度，對於相關的測試項目及合作廠商要求的應用項目做仔細的驗證，而業界各位先進不吝的指正，才會是我們最大的收穫。對於未來的發展，以下是幾個可以努力的方向：

1. 串列介面朝向節省時鐘信號的同步式多工化技術研究，更精簡配線。
2. 殊殊編碼，資料壓縮，解壓縮技術研究，以因應日後大量的資料傳輸要求。
3. 光纖或射頻(無線)傳輸取代傳統電氣信號在工業控制用的可行性研究。
4. 主動式資料傳輸錯誤校正的研究，使其接收端具有更正，修復資料的能力。
5. 遠端輸出點安全失效設計，讓使用者具有設定輸出安全值，並且在適當時機強迫執行安全輸出的能力。
6. 具中斷功能的輸入點，增加使用者可規劃的數位濾波頻寬要求。用來免除不必要的外部被動式濾波電路設計。

參考文獻：

[1] 馮元掄、陳文泉”分散式輸出入控制晶片－EDIO ASIC 介紹”，機械工業雜誌 205 期，p.98。

[2] 工研院機械所，”EDIO-S003 使用手冊”。

[3] Neelima Chaurasia，”工業和汽車應用中多種串列匯流排特性及比較”，電子工程專輯網站，http://www.eettaiwan.com/。

[4] John Patrick，”嵌入式系統設計中串列匯流排的選擇策略”，電子工程專輯網站，http://www.eettaiwan.com/。

[5] P. E. Boudreau, W. C. Bergman and D. R. lrvin, "Performance of a cyclic redundancy check and its interaction with a data scrambler", IBM J. Res Develop., vol.38 no.6, November 1994.

[6] “CRC 從原理到實現”， http://blog.csdn.net/。

[7] “關於 CRC”， http://blog.csdn.net/。

[8] “循環冗餘校驗 CRC 的算法分析和程式實現”， http://blog.csdn.net/。

[9] http://en.wikipedia.org/wiki/Cyclic_redundancy_check/。

[10] http://www.epcio.com.tw/。

作者：

黃靜宜，

工業技術研究院 機械與系統研究所

智慧機械技術組 機電控制整合部