



CAN Protocol

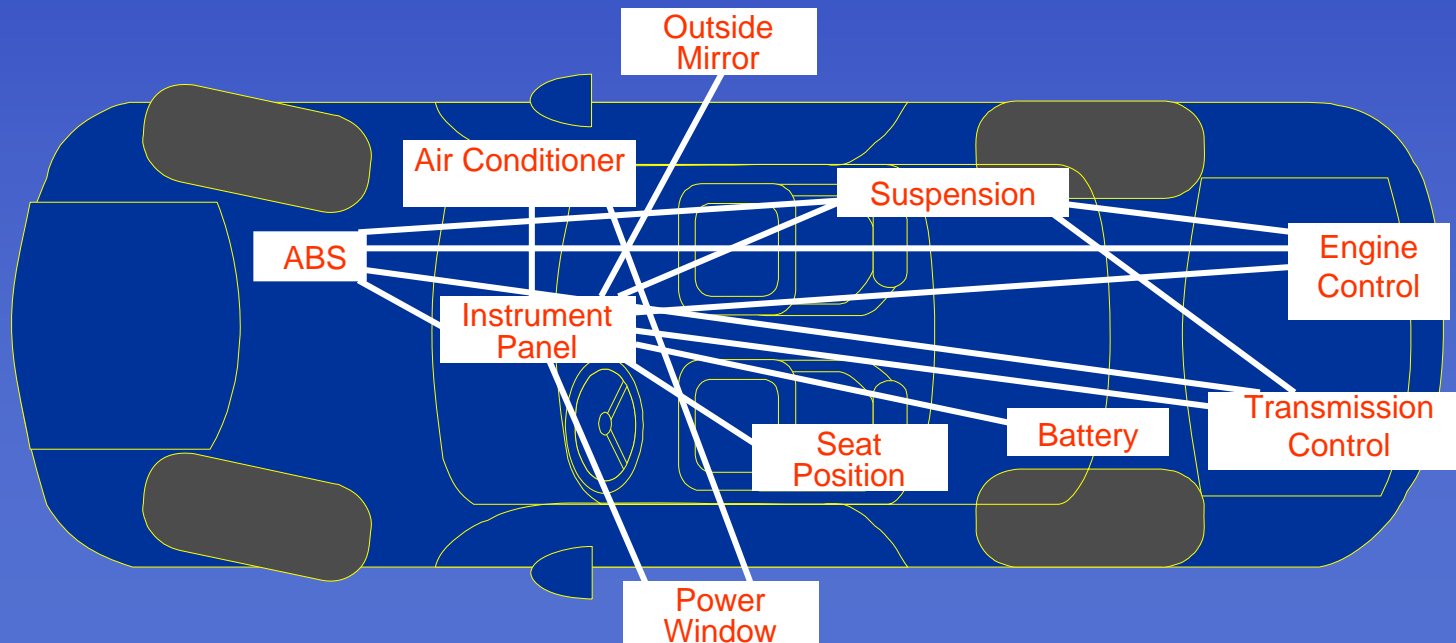


使用 **CAN bus** 來實現
高速而可靠的通信

CAN Bus 概要

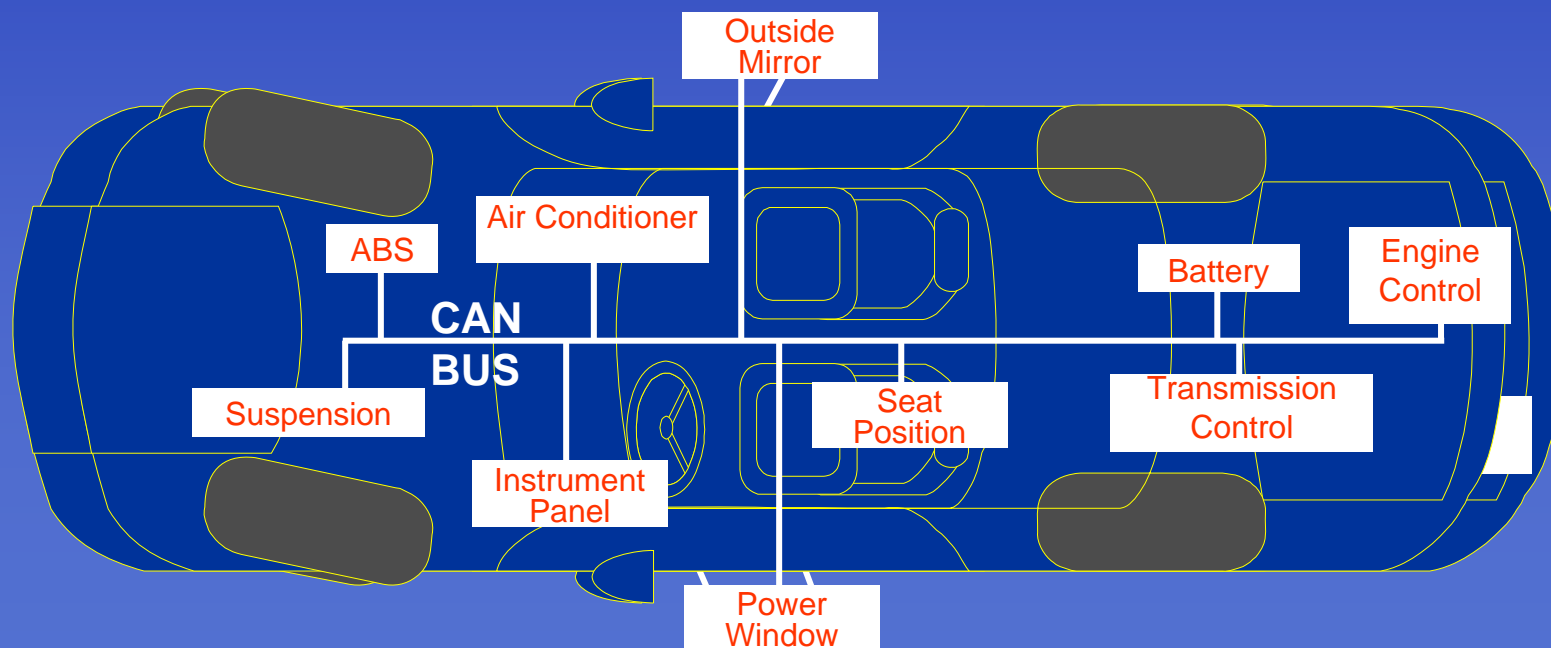
CAN (Controller Area Network) 通信協定於 80 年代由 Bosch 首先發展, 爲的是因應使用於新型汽車上不斷增加的電子裝置. 這些裝置使汽車增加許多功能與附加價值, 也增加控制系統的複雜度

在 CAN Bus 尚未被使用之前, 控制系統與各感測器之間大多透過點對點的電纜連接以完成控制及資料交換



CAN Bus 概要

在 CAN Bus 尚未被使用之前, 控制系統與各感測器之間大多透過點對點的電纜連接以完成控制及資料交換
使用 CAN bus, 透過簡單的串列界面即可完成對整個控制系統的連接及控制



CAN 的概要

CAN 是個先進的串列通信協定, 能有效率地支援分散式控制系統
它由以下兩個國際組織, International Standards Organization (ISO) and Society of Automotive Engineers (SAE) 制定標準

CAN 被廣泛的應用於汽車及工業控制上, 其他的應用尚包含船舶, 火車, 大樓自動化以及醫療儀器

目前已有大約 120 million 的各種 CAN Node 被使用於各種產品上.
預估在公元 2002 年將達到 170 million

CAN Bus 概要

如同大多數的網路系統 (Ethernet, USB, CAN, etc.), 分層的系統架構被使用來達成以下功能

- 使來自於不同製造商的產品能互相溝通
- 將相近的功能集合在一起
- 做為發展網路協定的骨架

ISO/OSI Seven Layer 網路參考模型定義了這種有系統的組織架構

WHAT IS CAN?

Controller Area Network

- 最早是由德國車廠 Robert Bosch GmbH 使用於歐洲的汽車中
- CAN 網路協定提供以下四大主要的優點
 - ◆ CAN 是經過標準化的通信協定
 - ◆ 通信造成的負擔將由系統的主要 CPU 轉移至智慧型的節點
 - ◆ 減少點對點需要的信號與控制連接線. (wiring Harness)
 - ◆ 符合廣大市場的需求

CAN Network Solutions

Bosch Specifications

- **Bosch Spec 2.0B (Active)**
 - ◆ CAN 通信協定の最新版本
 - ◆ 可包含 29 bit 的 Identifier (Extended Frame)
 - ◆ 完全地相容於較早期的版本 - Bosch spec 2.0A
 - ◆ 所有 Microchip 的 CAN devices 都與 CAN 2.0B 相容

CAN Network Solutions

Bosch Specifications

● Bosch Rev 2.0A

- ◆ Bosch 2.0A 的規格中共有 2 種不同的版本
- ◆ Rev2.0A – 此版本的通信協定只能傳送與接收“**standard frame message**”的資料 (11-bit Identifier) . 假設接收到一個 **extended frame message (29-bit Identifier)** 則將產生 “**error flag**”
- ◆ Rev2.0B Passive – 此版本的通信協定依然只能傳送及接收屬於 2.0A 規範的 message (Standard Frame)
 - ✦ 當接收到 “Extended Frame Message”, 它將會回應 acknowledge 但是會忽略此一 message !
 - 至少錯誤不再發生而且不會因為錯誤而干擾 CAN Bus

CAN / Network Solutions

CAN Bus 的能耐

- 資料長度
CAN 在單次傳送中可最多傳出 8 Byte 的資料
- 傳送速率 vs. 傳送距離

<u>Length (meters)</u>	<u>(Bit per Second)</u>
40m	1Mbps
500m	125Kbps
1,000m	50Kbps



- CAN Network

2.0B 版本的通信協定有能力支援最多達 2^{29} 個 node

CAN BUS LENGTH -v- BIT RATES

CAN BUS LENGTH -v- BIT RATES

- CAN belongs to a group of protocols known as - 'In Bit Response' protocols
 - ◆ Requires that bit time to travel from transmitting node to node farthest away shall take no longer than $\frac{2}{3}$ of total bit time
 - ✦ Result is a reduction in bus length with an increase in bit rate
 - ◆ Remaining $\frac{1}{3}$ of total bit time allows
 - ✦ Receiving node to perform bit wise arbitration
 - ✦ Switch into receive mode if arbitration lost

CAN BUS

LENGTH -v- BIT RATES

- Calculating Max. Transmission Distance for Bit Rate
 - ◆ Factors
 - ✦ Speed of electrical wave in copper is approx 2/3 of speed of light in a vacuum = $2/3 * 30\text{cm/ns} = 17\text{cm/ns}$ ($t_{\text{prop-line}}$)
 - ✦ Average propagation delay in transceiver = 25ns ($t_{\text{prop-tcvr}}$)
 - ◆ Equation - First step approximation
 - ✦ $2/3 t_{\text{bit}} \geq 4 * t_{\text{prop-tcvr}} + 2 * t_{\text{prop-line}}$
 - ◆ Results in 'rule of thumb' design rules
 - ✦ 40m at 1Mbit/s (CAN max. bit rate)
 - ✦ 400m at 100Kbit/s
 - ✦ 1000m at 40Kbit/s
 - Values may change depending on choice of transceiver

CAN BUS LENGTH -v- BIT RATES

- Example Bit Rate -v- Bus Length when utilizing high speed CAN controller e.g. PCA82C251

Bit Rate (Kbit/s)	Bus Length (m)
1000	30
500	100
250	250
125	500
62.5	1000

* source - CAN System Engineering - Lawrenz

CAN BUS LENGTH -v- BIT RATES

- Number of Nodes in a System
 - ◆ Factors
 - ✦ Electrical Characteristics of transceiver
 - ✦ Application specific drivers
 - ◆ Real Systems
 - ✦ Standard Differential Line Drivers - 32 Nodes
 - ✦ High Speed Diff. Line Drivers eg. PCA82C251 style - >110 Nodes
 - ✦ Application Specific Drivers - 64..128 Nodes

CAN Network Solutions

在汽車上的應用

- Driver Seat Control Unit
- Oil Condition and Level Sensor
- Diagnostics and Service Port
- Steering Position Sensor
- Engine Sensor Module
 - ◆ *Temperature Sensor*
 - ◆ *Air Intake Position Sensor*
 - ◆ *Pressure/Vacuum Sensors*
- Airbag Controller
- Environmental Controls
- Engine Control Applications
- Transmission Control
- Anti-Lock Brakes
- Suspension Control
- Dashboard/Instrumentation Control
- Power windows/Moon roof
- Power Mirrors

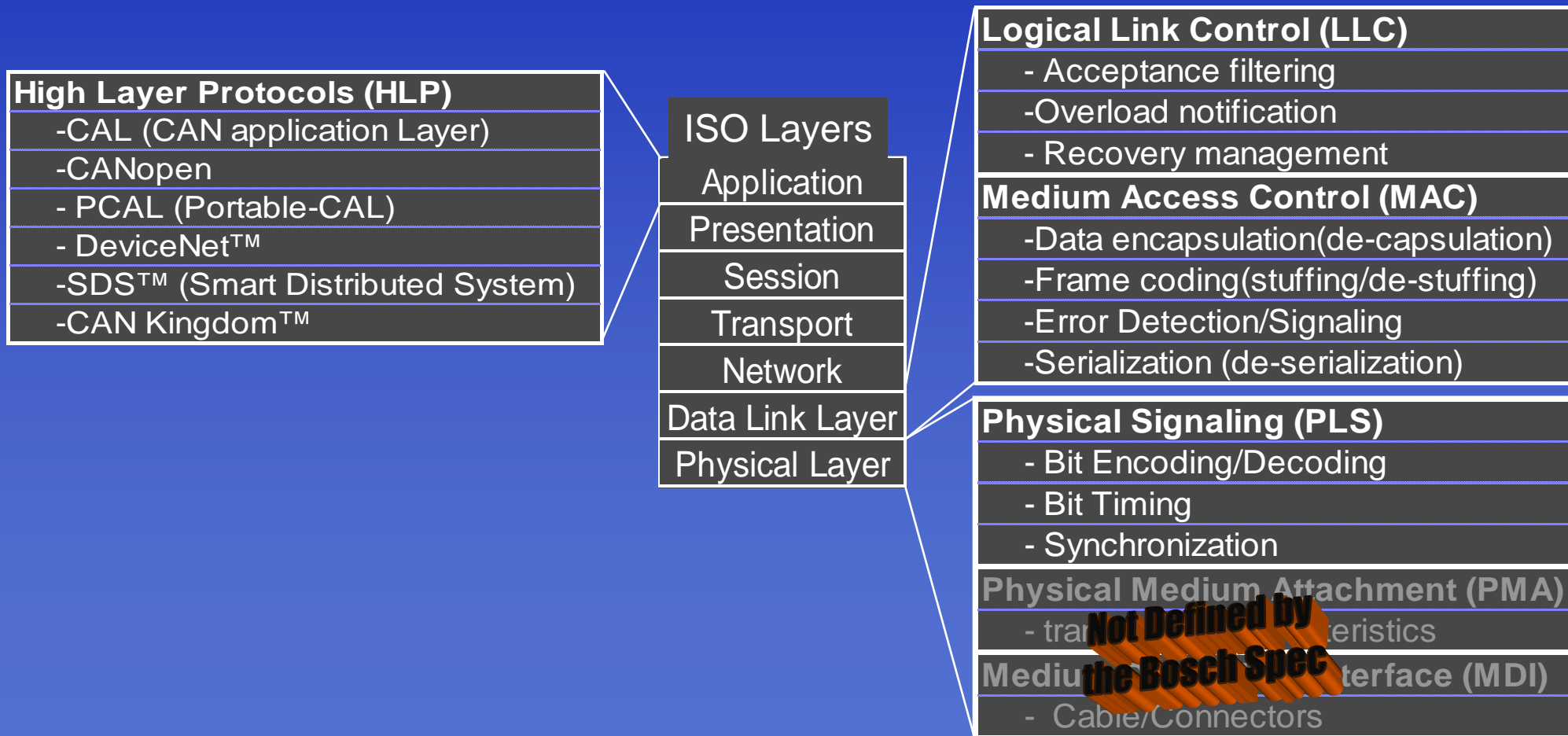
CAN Network Solutions

在非汽車相關產品的應用

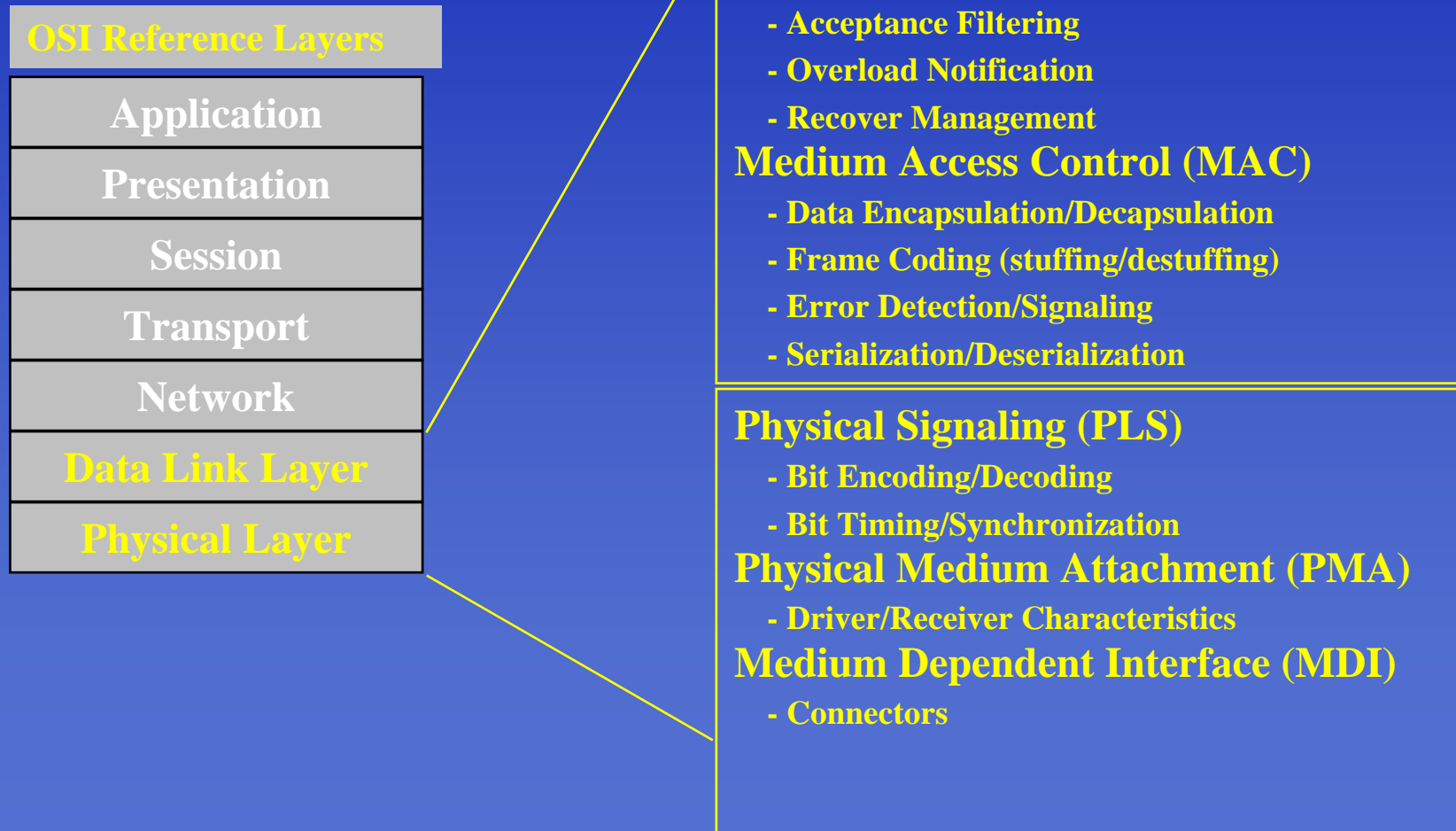
- Controller for Swimming Pool
- Industrial Lighting Controller
- Building Networks
 - ◆ Security System
 - ◆ Control Systems
 - ◆ Elevator Position Sensor
 - ◆ Elevator Control Panel
 - ◆ Lighting control
 - ◆ Emergency Lighting/Warning Systems
- Communication Protocol for industrial Copier
- RAID Storage Systems
- Hospital Bed Controller
- Intercept Seeker for Laser Targeting System
- Battery Charger on Forklift
- GPS navigation system for aviation
- Avionics Control for Model Rockets
- Secure Medicine Dispensing Machine
- Network system for Vending Machines

BASIC NETWORK OVERVIEW

ISO/OSI Network Layering Reference Model for CAN Bus



ISO/OSI 網路層 Reference Model



A Word on HLP's

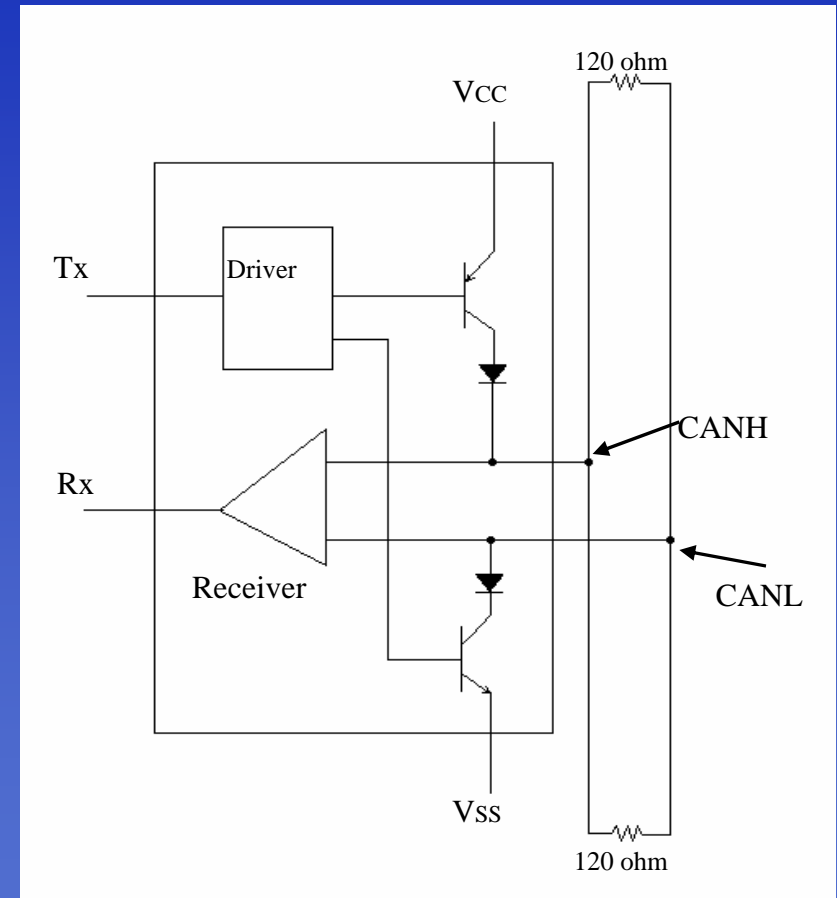
- 除了第一及第二層, 位於 ISO/OSI Model 的其他網路層則留給系統開發廠商來完成
 - ◆ 通常被稱為 Higher Layer Protocols (HLPs)
 - ◆ 例如：CanOpen, DeviceNet, SDS, Volcano, OSEK/VDX (European Auto Standard), Can Kingdom, CANopen/CAL
- HLPs 基本上對如何實現 7 層網路模型中的較高 5 層都有不同的定義方式. 每個系統開發廠商都選擇不同的方法來完成各網路層, 但是, 基本上它們都能履行各層所需的基本功能

CAN Bus 界面

- 爲了要減輕彼此溝通時將遭遇的問題, ISO 以及 SAE 已經依照 CAN 標準的需求, 制定完成以下包含 PMA (Physical Medium Attachment) 和 MDI (Medium Dependent Interface) 等的標準
 - ◆ ISO11898 是個適用於高速通信的標準規範 (up to 1Mbps)
 - ◆ ISO11519 是個適用於較低速通信的標準規範 (up to 125kbps)
 - ◆ SAE J1939 則被制定於使用在卡車及巴士時的應用
- 所有以上的標準都指定以 5V 差動式匯流排 (Differential Bus) 來作爲實體的連接界面

CAN 5V Differential Physical Layer

- 通常被稱為 CAN transceiver (例如: 82C251)
- V_{CC} 由 4.5 to 5.5V
- 與 uC 界面的 Tx 和 Rx 皆為數位的輸出/輸入信號
- CANH 和 CANL 間差動信號的電壓為 0V 到 3.0V
 $\Delta V > 1.0V$ 稱為 dominant
 $\Delta V < 0.5V$ 稱為 recessive
- Up to +/-40V continuous capable on CANH and CANL pins
- +/- 200V transient capable on CANH and CANL
- 40m max cable length @ 1Mbps



CAN Protocol Basics

CAN 的主要優點與特性

- 使用如下的通信協定來增加傳輸的效能
Carrier Sense Multiple Access and Collision Detection with Collision Resolution (CSMA/CD-CR)
- 以 **Message** 為信息傳遞的依據而非位址 (**Address**)
 - ◆ CAN Node 可以要求對方立刻傳送資料出來(發出 **Remote Transmit Request** 的要求), 與一般傳送的資料封包格式完全一樣, 其差別只在於 **RTR** 位元的值為 “1”
- CAN 是一種快速, 可靠的通信方式

CSMA/CD-CR

- CS** ● **Carrier Sense** - 在每次開始資料的傳送之前, 每一個 CAN node 必需偵測 bus 上的動作並確定一段期間內無任何的信號活動
- MA** **Multiple Access** - 當持續性的一段時間內無信號活動, 每一個 node 都有相同的機會來傳送訊息到 CAN bus
- CD** **Collision Detection** - 如果兩個 node 在同一時間點送出訊息, 則衝突便會發生. CAN 必需能偵測出此衝突的事件
- CR** **Collision Resolution** - 使用非破壞式的逐位元仲裁

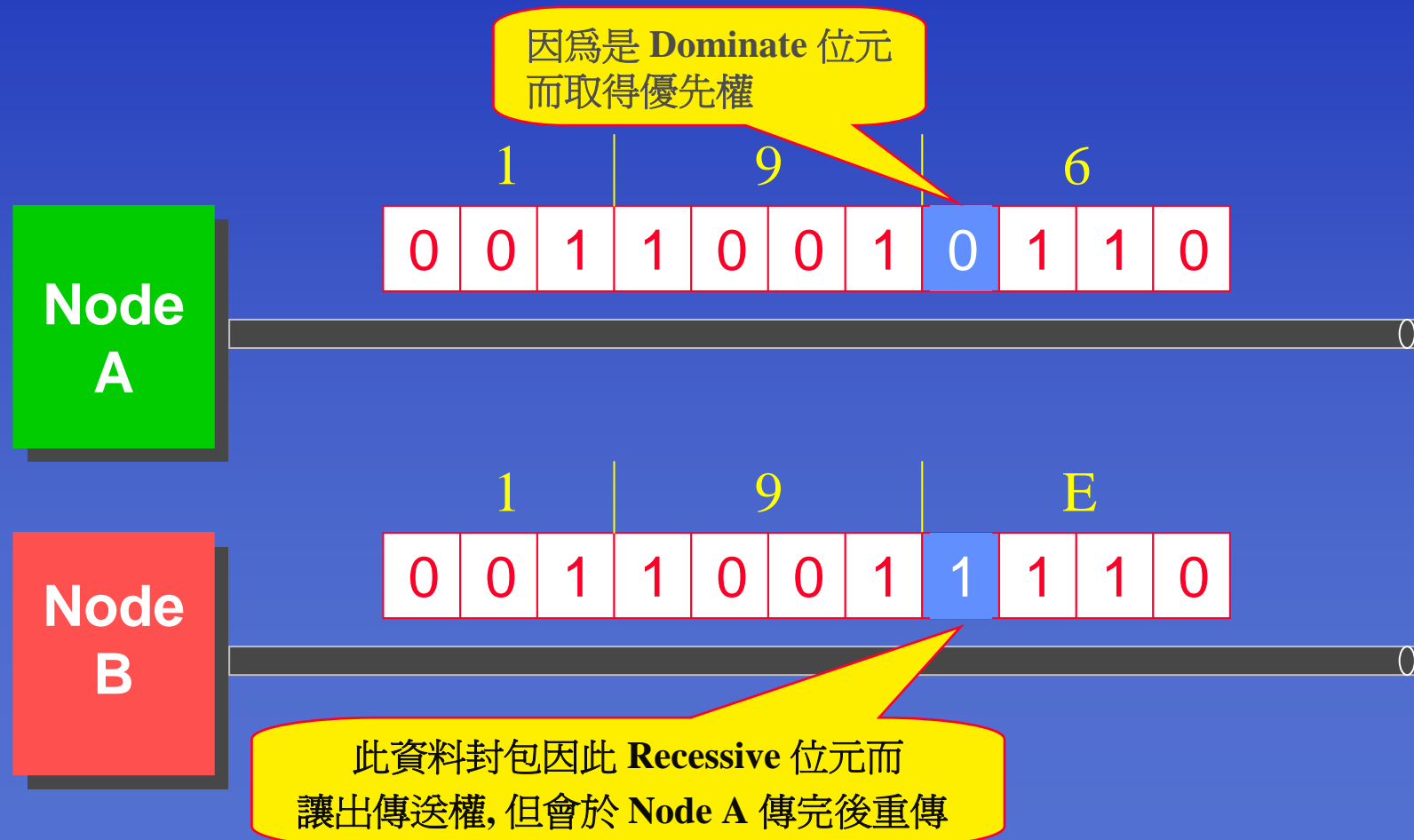
CSMA/CD-CR

CR Collision Resolution - 非破壞性的逐位元仲裁

- 即使資料碰撞發生, 所要發送的 **Messages** 仍然保持其完整性
- 所有仲裁行為發生時都不會對具有最高優先權的資料造成任何延遲或者錯誤
- 任何因為優先權較低而無法在第一時間傳送的資料封包會自動於下一個可傳送的時間自動的被傳送
- 要達到上述功能的需求:
 - ◆ **Dominant** 以及 **recessive** 位元的狀態必需被事先定義
 - ◆ 每個 **node** 必須不斷的監視 **bus** 上的資料流, 比較自己送出的資料與真正出現在 **bus** 上的資料是否相符!

CSMA/CD-CR

CR Collision Resolution - 非破壞性仲裁的例子



CAN is Message-Based (Not Address-Based)

- Messages 並非以位址為依據, 由一個節點被傳送到另一個節點
- 在傳送的訊息中包含了資料本身以及優先等級等相關資訊
- 所有的節點 (nodes) 對每個傳輸的資料都會接收, 並在收到正確的封包後回應 ACK (內建於 CAN 周邊的硬體功能中)
- 各個 Node 自行決定要處理此資訊或置之不理 (可經由軟體或設定適當的 Masks 以及 Filters 來達成)
- 因為使用的是 message-based 的架構, node-to-node 以及 broadcast (欲由一個以上的 node 接收並處理) 的訊息皆可經由相同的訊息 (message) 來傳送
- 新的 nodes 可隨時被加入系統中, 其他 nodes 並不需因為此新 node 的加入而必需更新內部資訊

CAN 是以 Message 資訊基礎的通信 (非傳統以 Address 的點對點方式)



Message Frames

● Data Frame

- ▶ 最爲普遍的類型
- ▶ 資料被從一個節點傳送到所有的節點

● Remote Frame

- ▶ 與 Data Frame 極爲相似 (只有 RTR bit 被設爲 1)
- ▶ 被用來向其它的 CAN node 主動要求資料之用

● Error Frame

- ▶ 用來表示在 CAN bus 上有錯誤狀況被偵測到

● Overload Frame

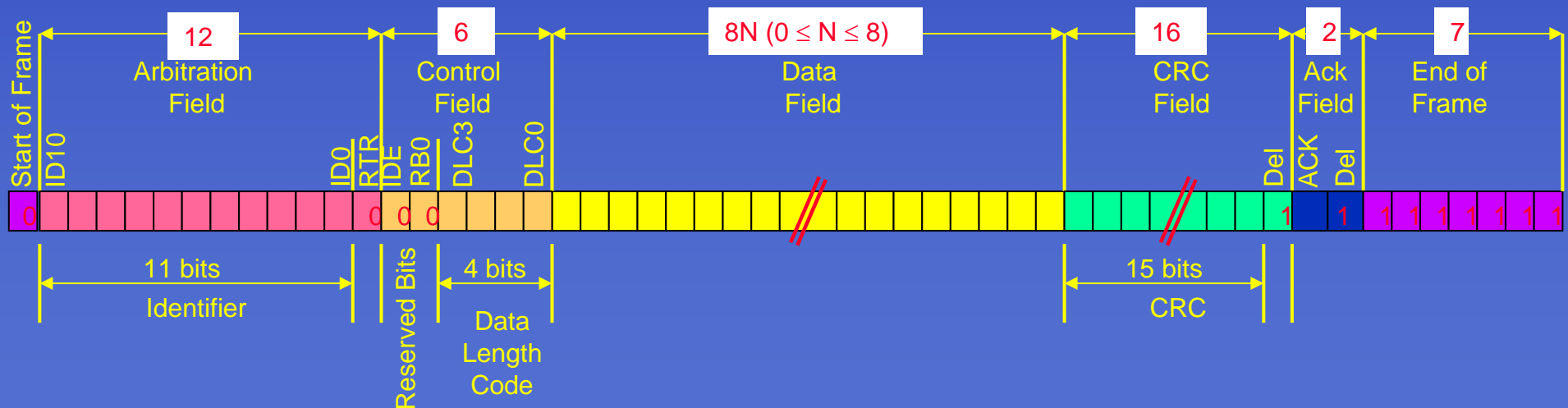
- ▶ 用來告知其他的 CAN nodes, 必須有更多的時間來處理收到的資料. 主要的功能是延遲下一個封包被傳送的時間

Data Frame

有兩種型式的 Data Frame, 依據不同版本的 CAN 協定規範而有差異, 但新版本可相容於舊的版本

□Standard Data Frame

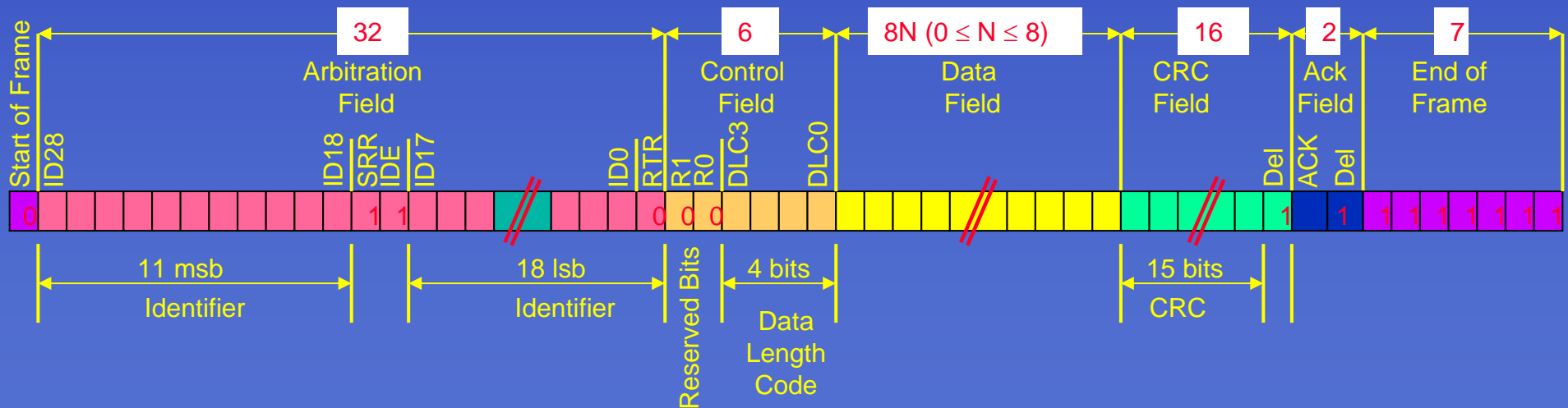
➡依據較早的 CAN 規範所訂定 (versions 1.0 及 2.0A). 定義成具有11 個位元的辨識欄位(identifier field)



Data Frame

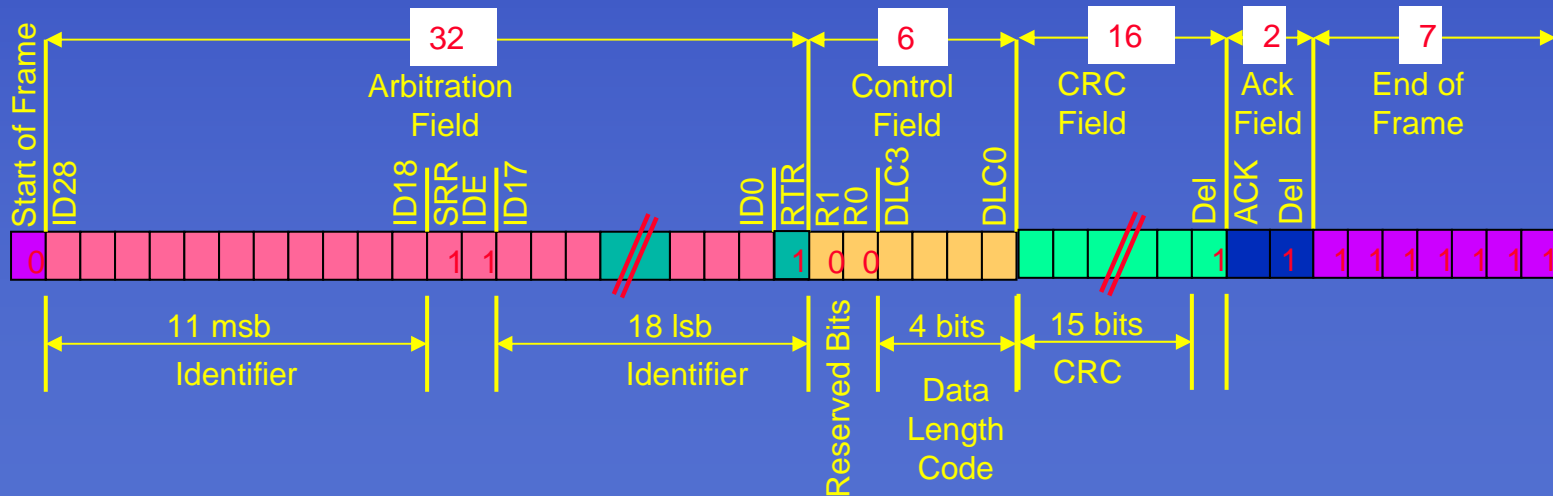
● Extended Data Frame

- ✦ 依據最新的 CAN 協定規範所定義 (version 2.0B). 辨識欄位(identifier field) 被擴展為 29 位元; 增加了可用 Message 的數量



Remote Frame

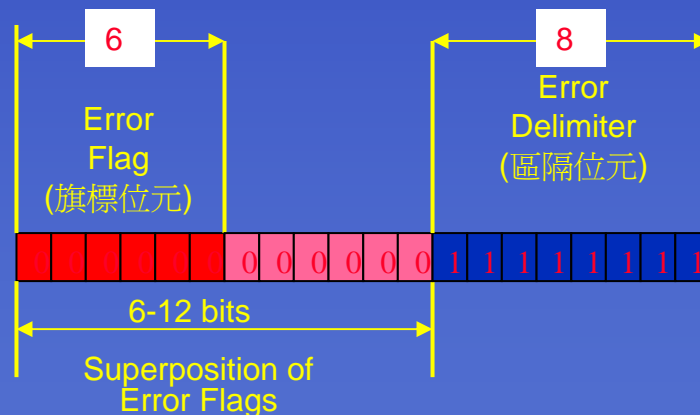
- 與 **Data Frame** 一樣, 有兩種型式的 **Remote Frame**, 依據不同版本的 **CAN** 協定規範而有差異, 但新版本可相容於舊的版本
- 除了不具備資料與 **RTR** 位元被設定為 “**recessive**” 狀態外, 與 **Data Frame** 幾乎完全相同



Error Frame

● Error Frame

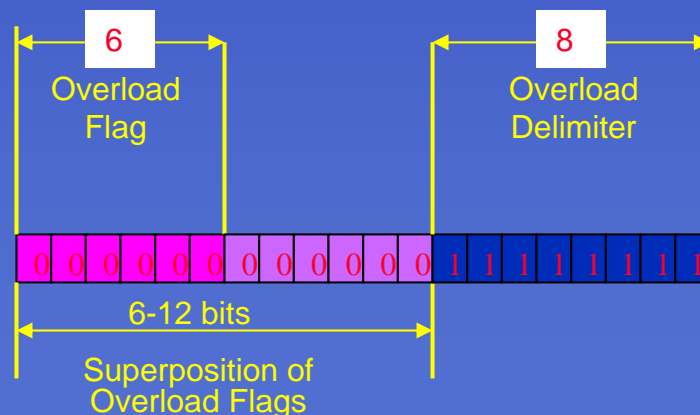
- ◆ 當 CAN node 偵測到 CAN 所定義的眾多型態的錯誤時, 將產生 Error Frame
- ◆ Error Frames 將在後續的說明中描述



Overload Frame

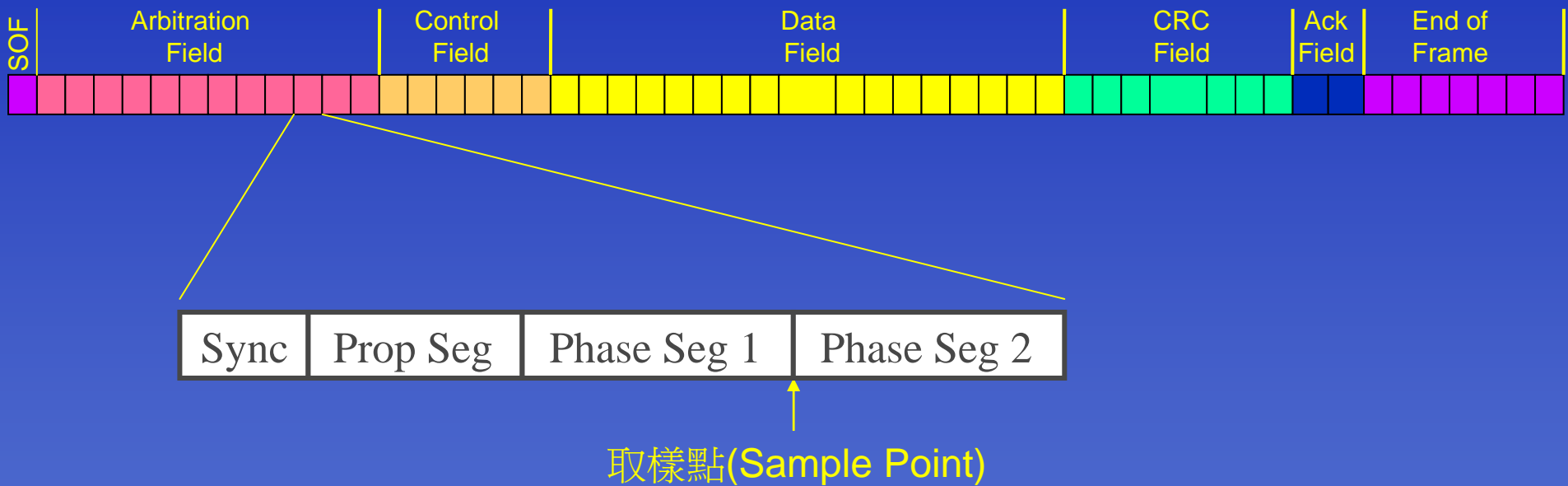
● Overload Frame

- ◆ 當一個 node 需要較多的時間來處理所接收的資料時，便送出 Overload Frame
- ◆ 當間隔位元少於規定值時 (最少要 3 個 recessive bits) 也會發生
- ◆ 間隔位元 (Intermission) 使 CAN node 有時間做內部資料處理



CAN Message 的位元結構

CAN Message



每個 CAN 的 message bit 是由 4 個 Segment 所組成

CAN Message 的位元結構

- 每一個 Bit Timing Segment 是由整數個單位時間所組成, 稱為 Time Quanta (TQ)



TQ 被硬體設定為 $2 \cdot (\text{BRP} + 1) \cdot (\text{Tosc}) \rightarrow (\text{Tosc} = 1/\text{Fosc})$
更改 Baud Rate 的預除器 (BRP) 將變更 TQ 的時間
Min = 1:1, Max = 1:64

CAN Message 的位元結構

Sync	Prop Seg		Phase Seg 1			Phase Seg 2		
TQ	TQ	TQ	TQ	TQ	TQ	TQ	TQ	TQ
1TQ	1-8TQ		1-8TQ			1-8TQ		

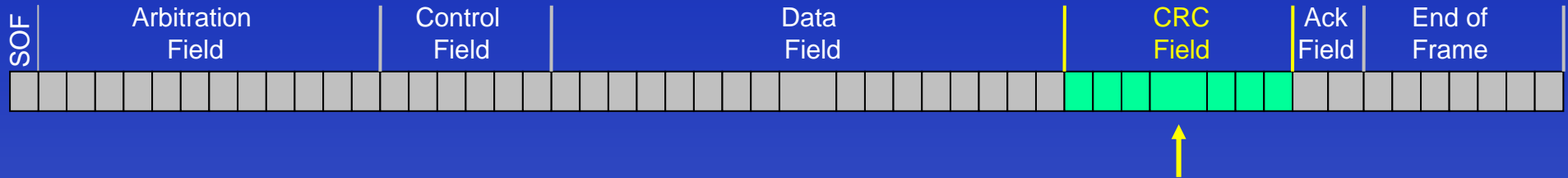
每個 **Timing Segment** 可被規劃為由指定數量的 **TQ** 所組成

CAN 對 Error 的處理

- 爲了正確的傳遞資訊, CAN 的通信協定非常嚴格地定義了許多錯誤的狀況, 當在不同的偵測點偵測到錯誤時皆會做出反應(根據 CAN 控制器當時的 Error State)
 - ◆ 確保 messages 的完整性
- 必需對有缺陷的 node 做處置並制約其行爲(Fault Confinement)
 - ◆ 依據錯誤程度的不同, CAN 的各 node 可能由正常的工作模式轉而進入完全離線的狀態
 - ◆ 對有錯誤或缺陷的 node 做出限制可防止有缺陷的 node 因爲不斷做出無效的傳送而使整個網路動彈不得

CAN 可偵測的各種錯誤

CAN Message

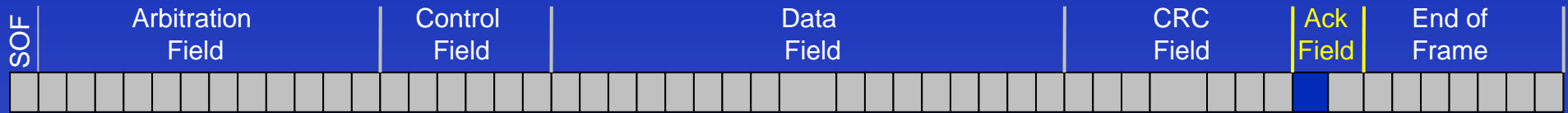


CRC Error

- 15-bit 的 CRC 會自動的被加入被傳送中的 Message 之 CRC 欄位中
- 所有 node 皆接收 Message, 並計算 CRC 後與接收到的 CRC 資料相比對
- 若兩者 CRC 不相等則視為發生 CRC 錯誤 並且產生一個 Error Frame
- 傳送此 message 的 CAN node 偵測到此錯誤後將重新傳送原來的 Message

CAN 可偵測的各種錯誤

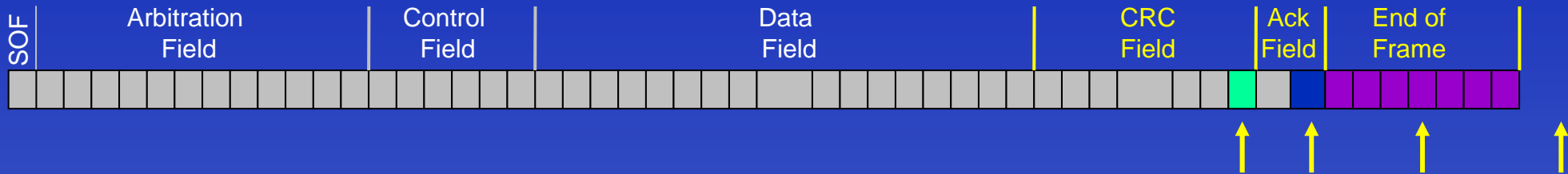
CAN Message



- Acknowledge Error (當無“告知收到”位元發生)
 - ◆ 傳送中的 node 在 Ack Slot 時檢查 Ack 位元, 此時它送出一個 recessive 位元並檢查是否有收到 dominant 位元
 - ◆ 如果偵測到 dominant 位元發生, 表示至少有一個 node 已正確地收到 Message
 - ◆ 否則, 將視為有 Ack 錯誤發生, 將產生一個“示誤封包(Error Message)”並重新傳送此次的資料

CAN 可偵測的各種錯誤

CAN Message

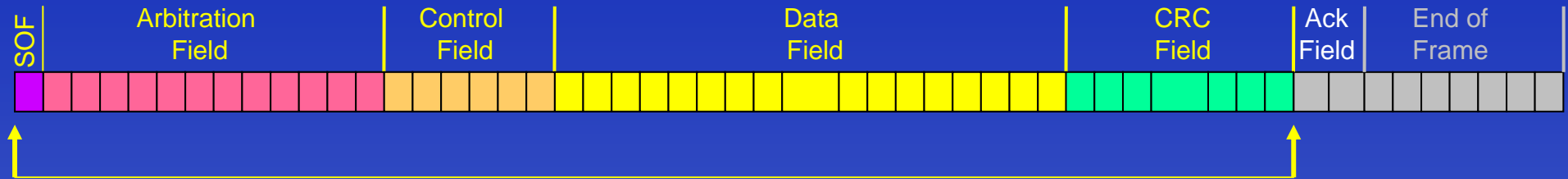


Form Error

- ▶ 若任一個 node 在 CRC Delimiter, Ack Delimiter, End of Frame (EOF) field 或 Interframe 的間隔偵測到有 dominate 位元則產生一個 Error Frame 來指明發生了 Form Error
- ▶ 原先的 message 將在 Error Frame 結束後重送

CAN 可偵測的各種錯誤

CAN Message

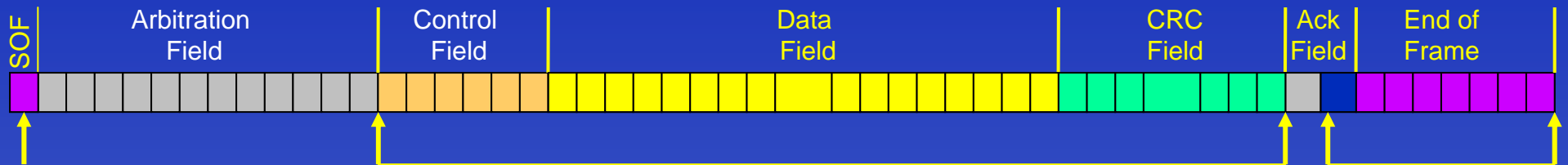


● Stuff Error

- ◆ CAN 的協定明確的定義，不可以有超過 5 個狀態相同的 bit 連續發生，若有要於 5 個相同的 bit 後補一個反相的 bit
- ◆ 如果有 6 相同的 bit 在 SOF 以及 CRC Delimiter 間連續發生，則被視為違反了位元填充 (bit Stuffing) 的原則，將產生 Error Frame 來回應偵測到的錯誤
- ◆ 原先的 message 將在 Error Frame 結束後重送

CAN 可偵測的各種錯誤

CAN Message



Bit Error

當傳送端發現它送出的信號與實際出現在 CAN Bus 上的不同, 則判斷有 Bit Error 發生

例外狀況：

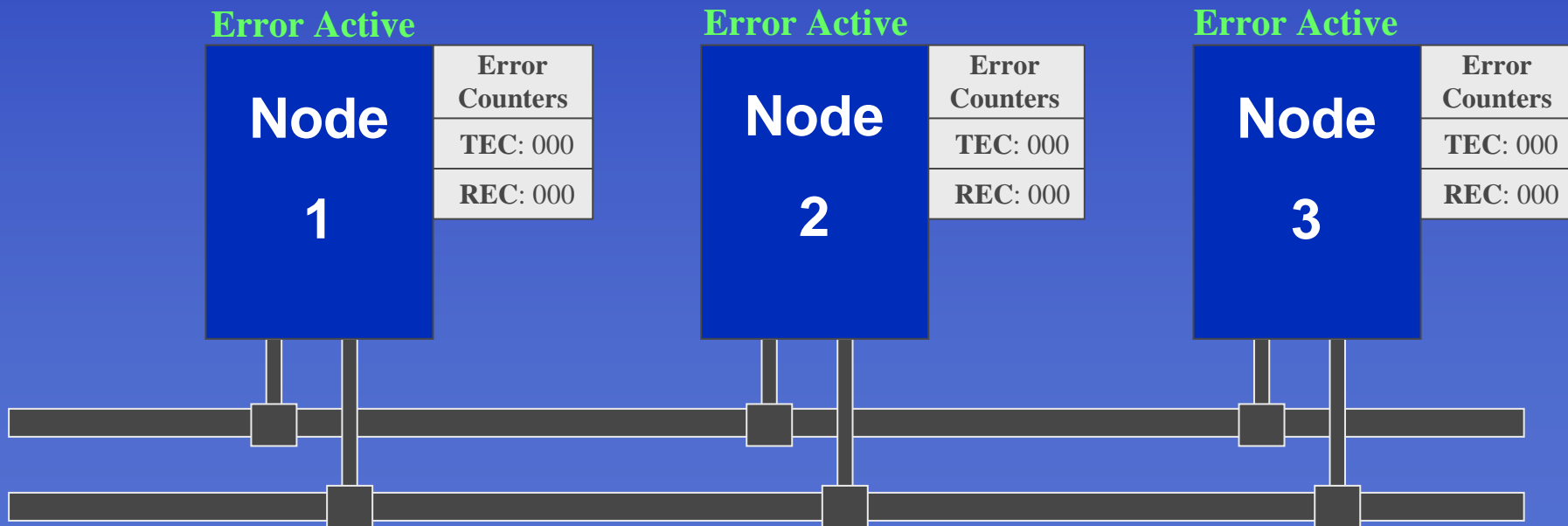
- 仲裁 (arbitration) 的區間不會發出 Error Frame (標準的仲裁程序)

- 在 Ack bit 的區間 (因為傳送端會送出 recessive 而 ACK 端要送出 dominate - 標準的 acknowledge 程序)

CAN 對錯誤的採取的限制

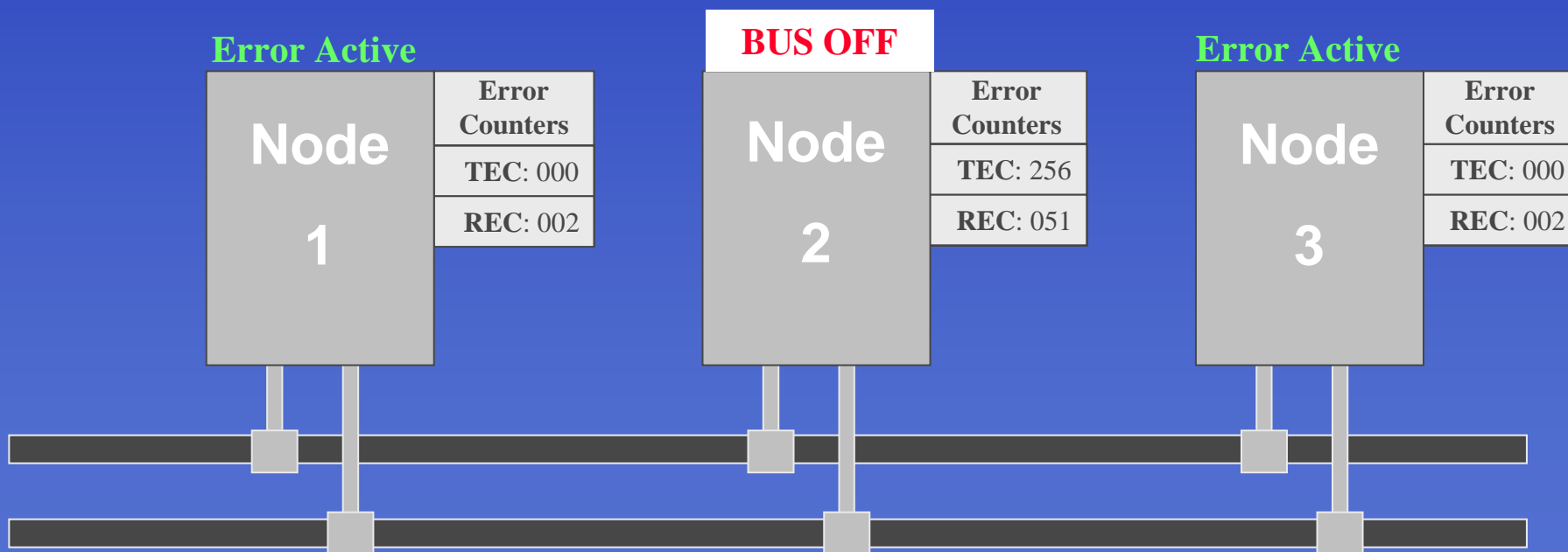
- 每一個 CAN node 都有各自獨立的 傳送以及接收錯誤計數器 (transmit 以及 receive error counters) TEC 與 REC

CAN 協定根據 Error Counter 的內容將錯誤狀態分爲三種：
: Error Active, Error Passive, and Bus-Off



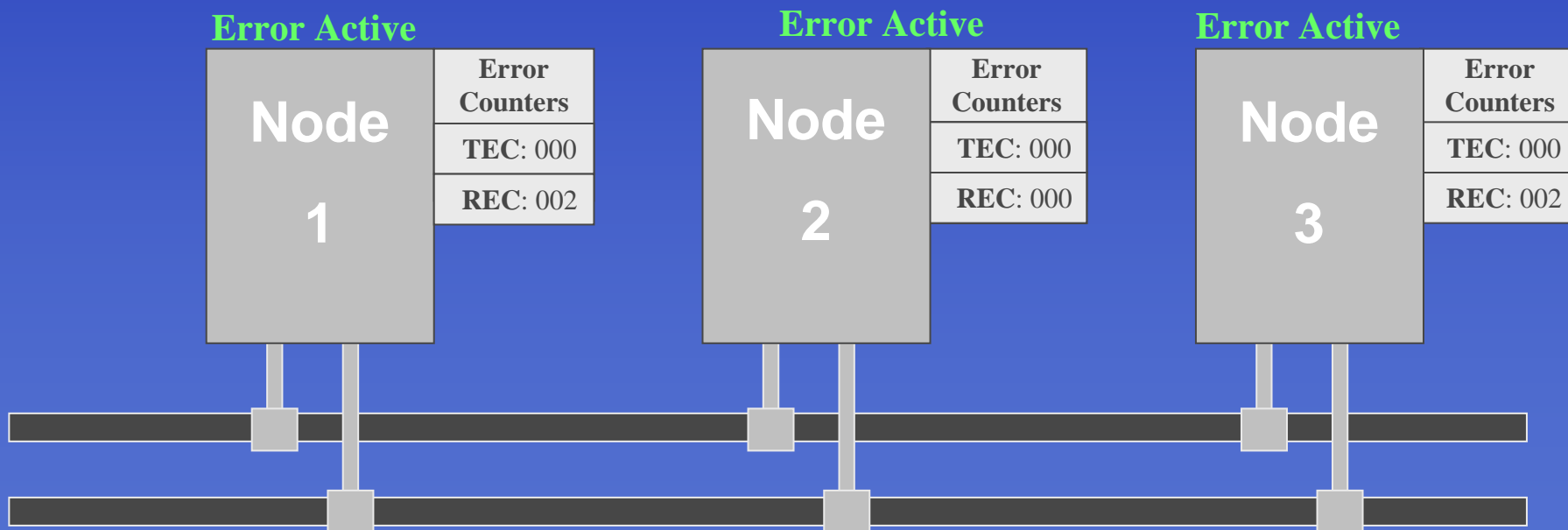
CAN 對錯誤的採取的限制

- 根據 node 所處的狀態不同, 其控制 CAN bus 的能力也跟著調整 (Active > Passive > Bus-Off)
- 如果到達了 Bus-Off 的狀態條件 (TEC > 255, CAN node 可將自身設定為完全離線狀態



CAN 對錯誤的採取的限制

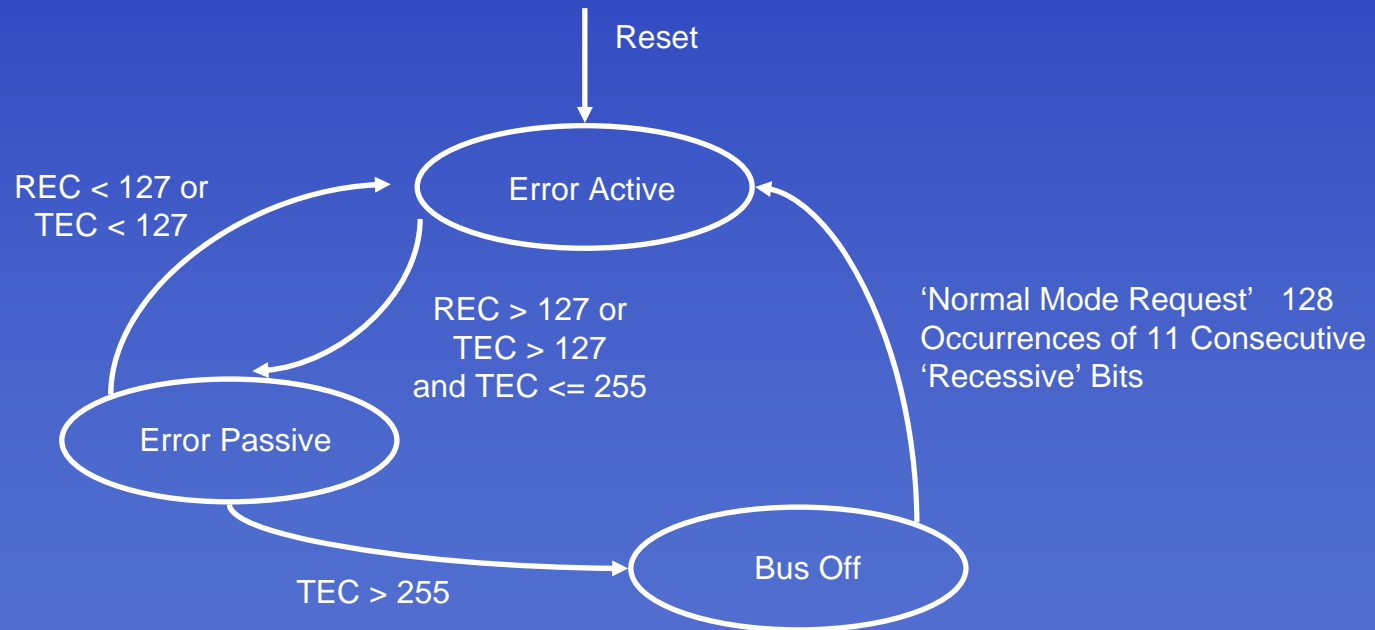
- 對已經進入 Bus-Off 狀態的 node ; CAN 定義了復原 (recovery) 的程序以便使其重回到活動(Active) 狀態
- 如此的做法可以防止嚴重錯誤的 node 持續的佔用有限的網路頻寬



CAN BUS ERROR HANDLING

- Error States

- ◆ Error Modes - State Transition Diagram



Error-Active (較積極的錯誤回應)

- CAN node 可以主動地傳送或接收訊息(messages) 和 Error-Frames (一般正常的工作模式)
- 當偵測到錯誤時, CAN node 將以送出正常 Error Frame (使用 6 個 dominate 位元的 Active Error Flag)的方式將正傳送中的 message 中斷掉
- 這樣的做法違反了位元填充的原則, 所以其他的 CAN node 也將產生它們自己的 Error Frames (稱為錯誤回應旗標 -> Error Echo Flags)
- 每當偵測到有錯誤發生, 適當的錯誤計數器 (TEC 或 REC) 將被加一
- 一旦Error Frame 完成後, bus 的活動又回到正常狀態
 - ◆ Message 將會被重新傳送 !!

Error-Passive (消極的錯誤回應)

- 當 TEC 或 REC 計數器超過了127, CAN node 變進入 **Error-Passive** 的狀態 (採用較消極的錯誤回應)
- 當一個處於 Error-Passive 狀態的 node 偵測到錯誤發生時將送出一個用 Passive Error Flag (6 個 recessive 位元)組成的 Error Flag
- 使用 Passive Error Flags 的 Error Frames 將不會影響傳送中的 Message
- 適當的錯誤計數器 (TEC 或 REC) 將被加一
- 當 Error Frame 完成後, 處於 Error-Passive 的 node 會等到以下的條件符合後才會被致能傳送
 - ◆ 在 dominate 位元之後有 3 個 intermission 的位元加上 8 個位元的時間長度

Bus-Off 狀態

- 當 TEC 計數器值超過 255
 - ◆ 控制器將進入 Bus-Off 狀態
- 此時的 CAN node 將無法傳送或接收任何的 message 或錯誤狀態
- 經過以下的程序, CAN node 可再度回到 Error Active 的狀態
 - ◆ 接收到正確的 “bus recovery” 程序
 - ✦ 11 個連續的 recessive 位元發生128 次 !!
 - ◆ 經由硬體或軟體的重置動作
 - ◆ 將控制器設定成 “Configuration mode”

錯誤偵測的重點整理

- CAN nodes 可依錯誤的嚴重程度來自動地減少它們對 bus 的控制能力
- 透過以下三種狀態的轉換可以達成這樣的要求
 - ◆ Error-Active (正常的操作模式)
 - ◆ Error-Passive (控制 bus 的能力被降低)
 - ◆ Bus-Off (無法控制 CAN bus)
- 不同 Error-States 間的差異在於錯誤的 node 該如何被限制行爲
- CAN 對錯誤時的能力限制 (Fault Confinement) 可防止有缺陷的 node 使得整個網路當掉

Microchip CAN Solutions



Microchip CAN Solutions Stand Alone Family and Transceiver

Performance/Memory

MCP2515
18/20 Pin
CAN Engine

MCP25050
Mixed Sig
I/O Exp

MCP25055
Mixed Sig
I/O Exp
1 wire

MCP2510
18/20 Pin
CAN Eng

MCP25020
Digital
I/O Exp

MCP25025
Digital
I/O Exp
1 wire

MCP2551
8 Pin

Transceiver family

CAN Stand-Alone Product family

In Production

Samples Available

In Development

Complexity/Pin count

PIC18F系列內含 CAN Controller 的產品現況

28 to 80 Pins

64Kb

PIC18F2680
nW/EIS/ECAN

PIC18F4680
nW/EIS/ECAN

PIC18F6680
ECAN

PIC18F8680
ECAN

48Kb

PIC18F2525
nW/EIS/ECAN

PIC18F4585
nW/EIS/ECAN

PIC18F6585
ECAN

PIC18F8585
ECAN

32Kb

PIC18F258
CAN

PIC18F458
CAN

16Kb

PIC18F248
CAN

PIC18F448
CAN

28/40/44Pins

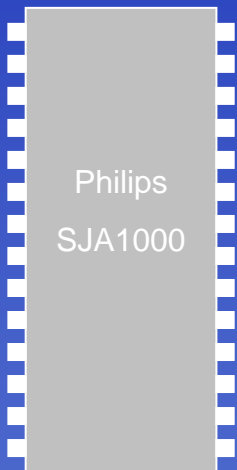
64/80 Pins

MCP2515

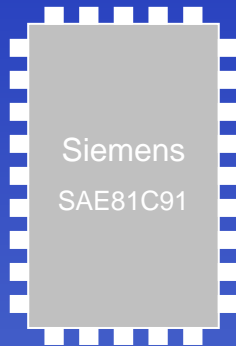
CAN Controller

MCP2515 CAN Controller

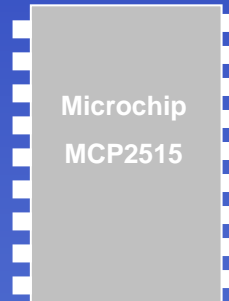
- 接腳數目最少的獨立式 CAN 協定控制器
- MCP2510 的改良版本 (Device Net Support added)



28-lead



28-lead



18-lead



44-lead



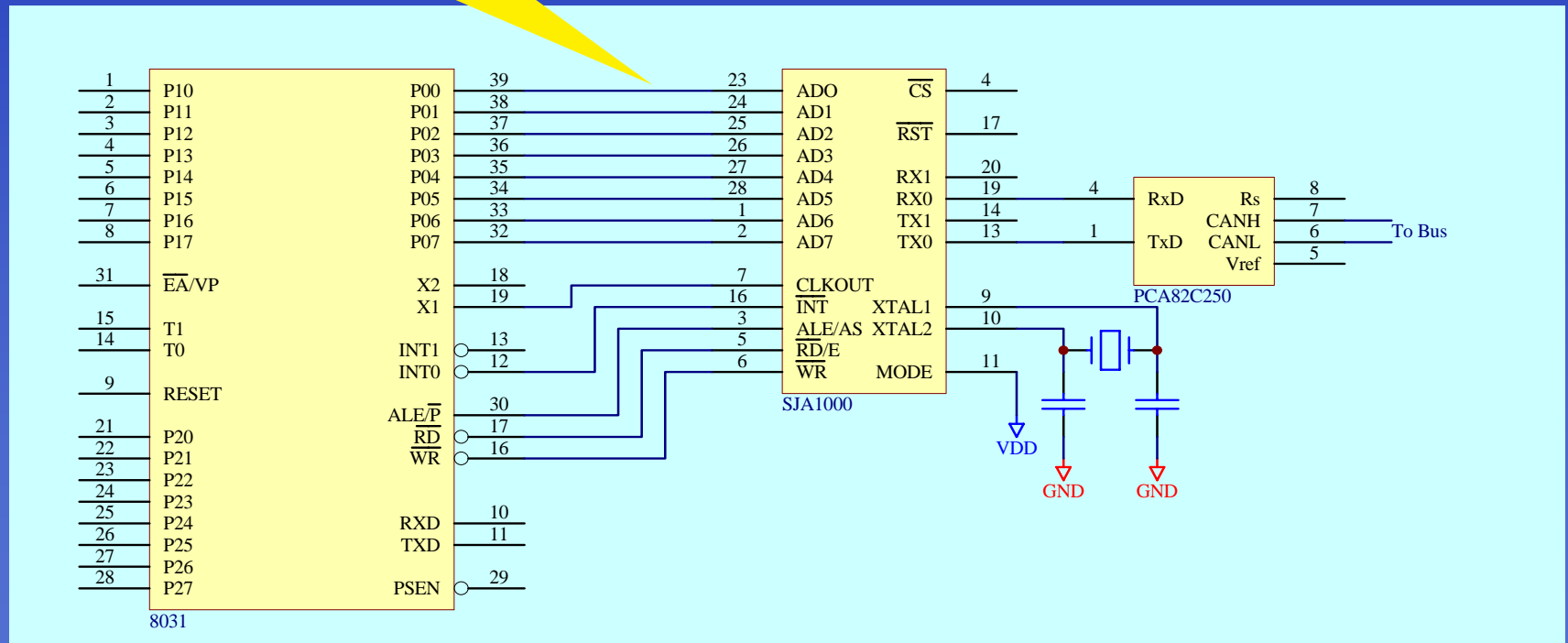
44-lead



44-lead

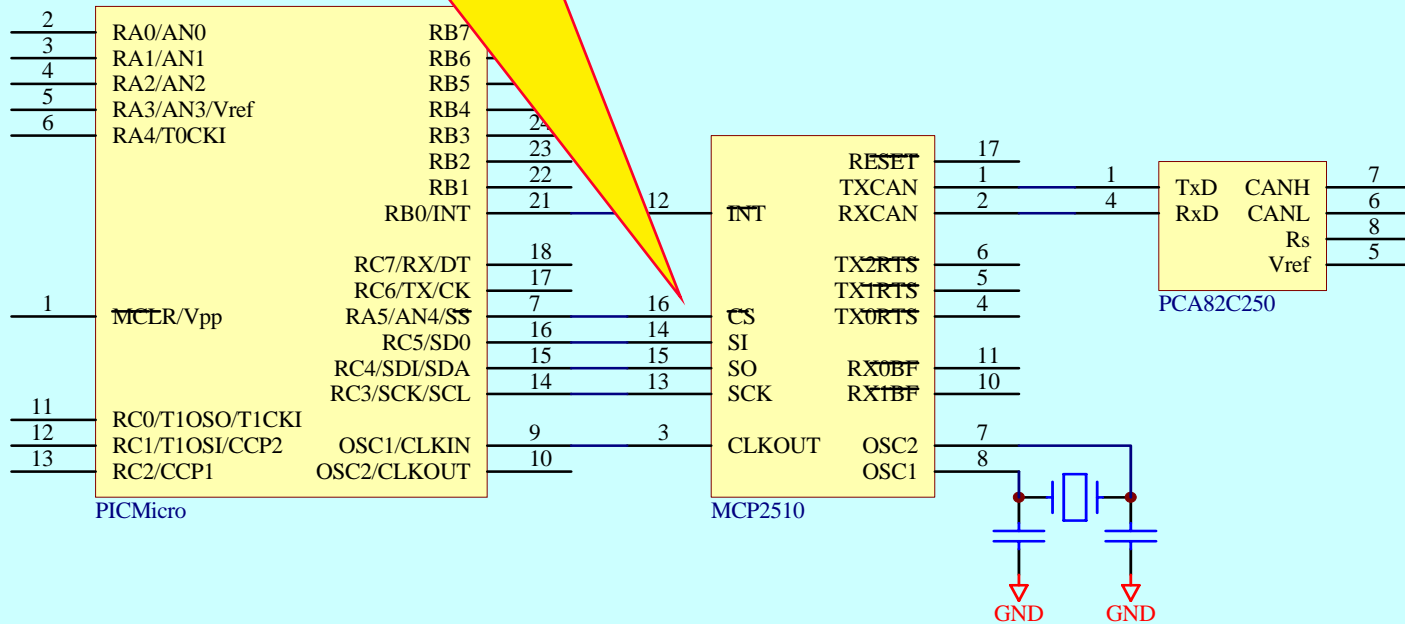
使用其他 Stand Alone CAN 協定控制器的例子

需要許多的 I/O 線來連接



使用 MCP2515 做為 CAN 協定控制器的範例

使用 **SPI** 界面
減少信號連接的困擾



MCP2515 CAN 協定控制器

- CAN 2.0B Active
 - ◆ 可傳送及接收 standard 和 extended 的 frames , 最高的通信速率為 1Mbps
- 支援 SPI™ Modes 0,0 and 1,1
- 多工式的 I/O pins 能將原本屬於驅動傳送或指示資料接收用的腳 pin 規劃成 I/O 的功用
- 內建振盪線路並有頻率輸出(CLKOUT) 的腳位以做為搭配處理器的工作頻率
- 外部的 Reset 腳位以及中斷的輸出接腳
- 由 3.0V 到 5.5V 的電壓都可以工作

MCP2515 的操作模式

- 根據不同的需求, MCP2515 共有五種不同的操作模式

CONFIGURATION

NORMAL

LISTEN ONLY

: 不送出任何資料但接收
正常

SLEEP

: 省電模式

LOOPBACK

: 用於 Debug 之用

MCP2515 的功能概要

- 三個可設定優先權且有終止傳送能力的傳送緩衝器 (Transmit Buffers)
- 兩個有 Message Storage 優先設定的接收緩衝器(Receive Buffers)
- 六個完整的 Acceptance Filters 以及兩個 Acceptance Filter 的 Masks 以便過濾掉不想要的 messages
- 支援 CAN V2.0B 的 Protocol Engine 以及 Message Assembly Buffer (MAB)
- SPI™ 介面, 最高的速率為 5 Mhz
- 有 Clock 的輸出腳位並有可軟體規劃的預除器, 並且可選擇是否要輸出
- 兩個Receive buffer 中斷的狀態接腳也可規劃為一般的可規劃輸出
- 三個Transmit buffer 的傳送要求輸入也可當做一般的輸入腳位
- 暫存器的安排與將完成的 PIC+CAN 的控制器相同；易於將來升級
- 低功耗的 Sleep Mode ($< 5 \mu\text{A}$)

內建於 PIC18F 系列的 CAN Solution

- PIC18F 系列內建的 CAN Solution 分爲兩大類
 - ◆ Standard CAN (Legacy Mode)
 - ✧ PIC18F248/258 , PIC18F448/458
 - ✧ 支援與 MCP2510/2515 相同的 Buffer & Mask 結構
 - 3 Transmitter Buffers
 - 2 Receiver Buffers & 1 MAB (Message Assembly Buffer)
 - 6 Acceptance Filter & 2 Mask

內建於 PIC18F 系列的 CAN Solution

◆ Enhanced CAN - ECAN

- ✦ PIC18F6680/6585 , PIC18F8680/8585
- ✦ PIC18F2680/2585 , PIC18F4680/4585 (nano Watt)
- ✦ 支援 3 種 Buffer Mode , Mode 0 與 Standard CAN 相容
 - “Mode 0” Legacy Mode
 - “Mode 1” Enhanced Mode
 - “Mode 2” FIFO Mode

PIC18FXX8 CAN Module

- First PIC18 micro with integrated CAN Module (CAN 2.0B Active Spec) and Flash Memory
- Programmable bit rate up to 1 Mbps
- Three Transmit, 2 Receive Message Buffers
- Six Acceptance Filters, 2 Acceptance Masks
- Wake-up on CAN message functionality
- Programmable time-stamp operation
- Multiple operation modes
- Supports to receive/transmit Remote Frames

PIC18F258/458

Key Features

CORE

- Operating Conditions:
 - ◆ Freq DC - 40MHz (10MIPs)
 - ◆ Volt 2.5 - 5.5v
 - ◆ Temp Ind. (-40 to 85C)
Ext. (-40 to 125C)
- Clocks Options
 - ◆ 32KHz options on TMR1
 - ◆ 4x PLL on Osc1
- 16Bit Instruction Bus 8Bit Data Bus
- 8x8 Hardware Multiply

MEMORY (Linearly Addressable)

- Flash: 32KBytes (1K e/w)
- RAM: 1.5K Bytes
- EEPROM: 256 Bytes (100K e/w)

PERIPHERAL SET

- CAN Bus rev2.0B Active
- 10-bit ADC
- 9-bit Addressable USART
- MSSP (SPI/I2C)
- Two Capture, Compare PWM's
- 3x16-bit, 1x8-bit, WDT
- Programmable Brown out Rest
- Low Voltage Detect
- Self Programmable Flash Mem.

PACKAGE

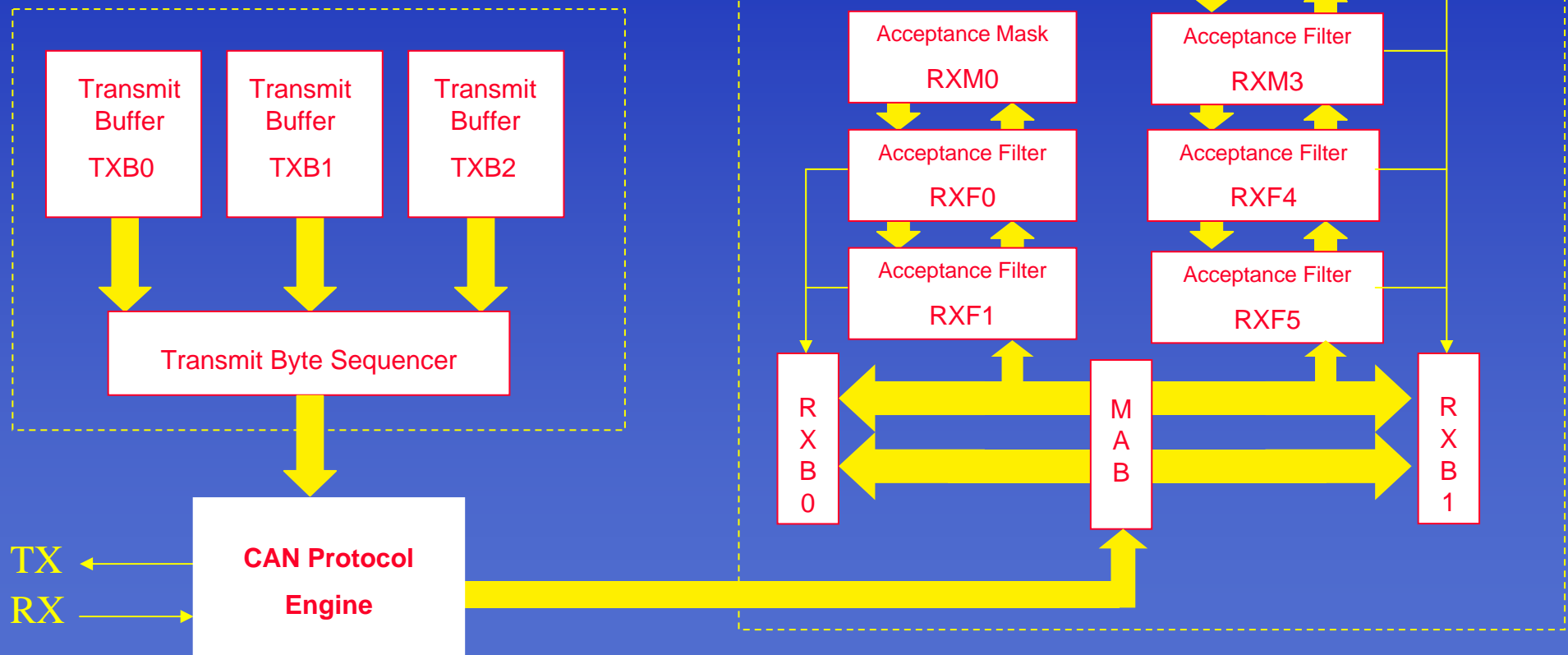
- 28L SDIP, SIOC
- 40L PDIP, 44L PLCC TQFP

RTOS and OSEK Support

DaimlerChrysler drivers developed by Vector

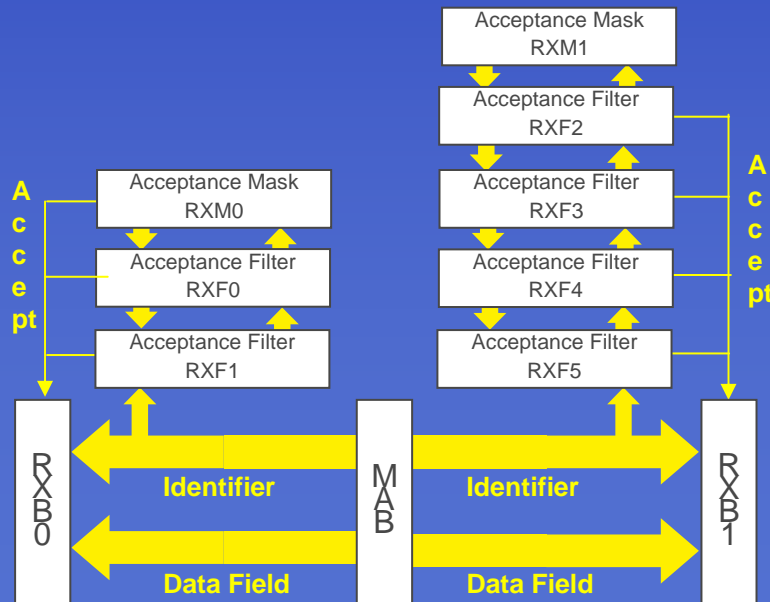
- Supports 18Cxxx and 16Cxxx microcontrollers
- Supports MCP2510 CAN Interface IC
- OSEK/VDX-OS V2.1 Revision 1
- OSEK/VDX-COM V2.1 Revision 1
- OSEK/VDX-NM V2.5
- Conformance classes BCC1, ECC1, BCC2, ECC2
- OSEK-COM Inter Task Communication
- OIL-configurator for Windows 95/98/NT
- Generation tool
- Sample programs

PIC18FXX8 支援的 CAN Buffers



Masks 和 Filters 的概念

- 用來決定什麼 Message 將被接受
- **Masks** 決定那些 filter 位元將被用來過濾 ID (Identifier)
 - ◆ '1' = 使用 filter 來過濾此位元, '0' = 無條件接受
- 在 **Identifier** 中的位元必需與 **Filter** 中對應的位元相等才被接受
- **Filter** 可以被想成是觸發 (trigger) 事件
- **Mask** 決定那些 Filter 位元是 “don't care”



FILTER/MASK TRUTH TABLE			
Mask Bit n	Filter Bit n	Message Identifier Bit n	Accept or Reject bit n
0	X	X	Accept
1	0	0	Accept
1	0	1	Reject
1	1	0	Reject
1	1	1	Accept

CAN Module 與 Programmer 的介面

- Total of 115 registers (addresses F00h - F76h) in Databank 15
- Only F60h-F76h in “Access Bank High”
- Special mechanism to speed up access to non-access bank registers (to be reviewed in later slides)
- Eight CAN related interrupt sources - all with programmable priority
- Compatible with future PIC18F2680/4680 and PIC18F6680/8680

PIC18FXX8 CAN Module 可用的操作模式

● Configuration Mode

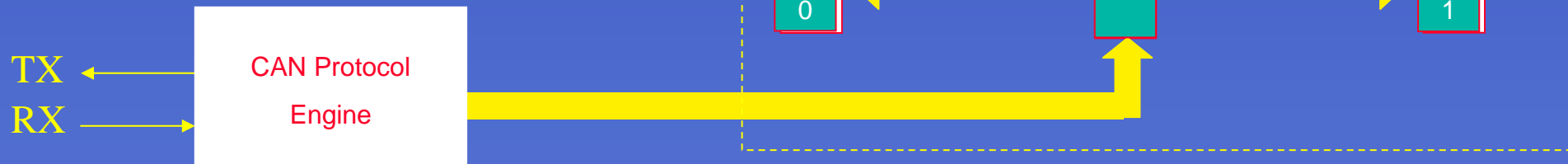
- ◆ Power-up 後, CAN Module 會被設定於此模式
- ◆ CAN Module 必須在此模式做適當的設定才可正確地工作
- ◆ 與 CAN Module Configuration 相關的暫存器是受保護的
 - ✦ 3 Bit Timing Registers (BRGCONn, $1 \leq n \leq 3$)
 - ✦ 24 Identifier Acceptance Filter Registers (RXFnSIDH, RXFnSIDL, RXFnEIDH, RXFnEIDL ($0 \leq n \leq 5$))
 - ✦ 8 Identifier Acceptance Mask Registers (RXMnSIDH, RXMnSIDL, RXMnEIDH, RXMnEIDL ($0 \leq n \leq 1$))
- ◆ 以上所列的暫存器在其他 Modes 時無法寫入而且在讀取時得到的結果都是“0”
- ◆ 當處於 Configuration mode, device 的狀態是“off-line”

PIC18FXX8 CAN Module 可用的操作模式

- Listen Only Mode
 - ◆ Useful for bus monitor, auto-baud detection application
- Loopback Mode
 - ◆ Useful during debug/development phase, self-tests
 - ◆ No CAN transceiver or node required on bus
- Disable Mode
 - ◆ Must, if wake-up on CAN bus is desired
 - ◆ Allows use of CAN pins for I/O function
- Normal Mode
 - ◆ To participate in CAN bus communication

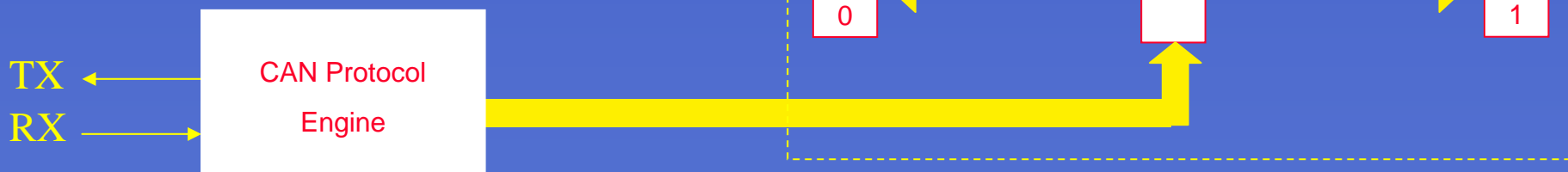
PICFXX8 的 Receive Buffers

兩個接收 Buffers 並
有事先定義的優先權
以決定資訊的儲存方式
(RXB0 有較高的優先
權)



PIC18FXX8 的 Receive Buffers

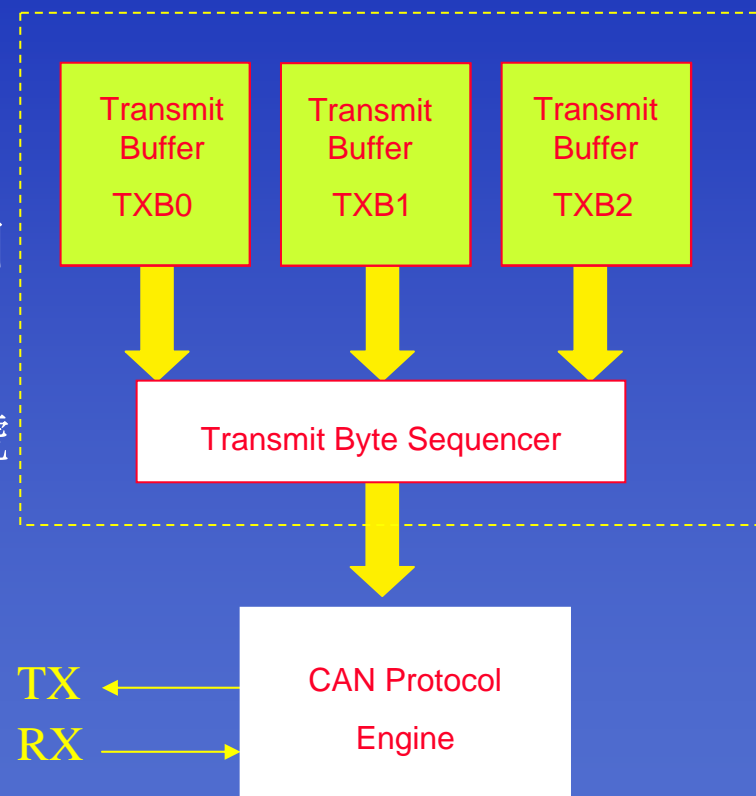
- RXB1 有四個 Filters 以及一個 Mask
 - RXF2, RXF3, RXF4, RXF5, 以及 **RXM1**
- 當 RXB0 無法即時被處理而又有後續的資料封包時, 可以被 'Rollover' 至 RXB1
 - 要設定 RXB0CON.RXB0DBEN
- RXB0 有兩個 Filters 以及一個 Mask
 - RXF0, RXF1 and **RXM0**
 - 對於訊息的接收有較高的優先
- PIE3.RXBnIE 使得 CAN Module 於接收到資料後能以 PIR3.RXBnIF 來中斷 CPU
- RXBxCON.RXFUL 會於接收完成時被設為 1
 - 必需被清除後下一個 Message 才能被接收
 - 如此可確保已收到的 Message 不會因失誤而被覆蓋掉



PIC18FXX8 的 Transmit Buffers

三個具備優先順序的傳送緩衝器(Transmit Buffers), 並有終止傳送的能力

- 由 TXBnCON.TXPRI 設定傳送的優先順序
 - 與識別欄中對 Message 所定的優先權並不相同, 莫混為一談
 - 較高的編號具有較高的優先權
 - 若各傳送 Buffer 的 TXPRI 設定相同則以編號來決定其最後的優先順序
- 傳送控制暫存器 TXBnCON 的 TXREQ 位元被用來要求傳送
 - 寫入 TX buffers 前需先確定此位元已被清除
 - 若用軟體清除則會被視為 “ABORT” 的請求
 - 傳送成功後 PIR3.TXnIF 旗號將被設定為 “1”



Microchip 的 AN738

- AN738 提供了完整的 CAN Library，使程式的開發更容易
 - ◆ Configuration / Initialization Functions
 - ◆ Operation Functions
 - ◆ Status Check Functions
- 使用列舉 (enum) 資料型別來定義及傳遞欲設定的操作形式
- 支援使用 Standard 及 Extended Data Frame
- 提供 MPLAB C18 及 HI-TECH C18 的通用宣告

Microchip 的 AN738

● AN738 提供的 Functions

Function	Category	Page Number
CANInitialize	Configuration	4
CANSetOperationMode	Configuration	6
CANSetOperationModeNoWait	Configuration	7
CANSetBaudRate	Configuration	8
CANSetMask	Configuration	10
CANSetFilter	Configuration	11
CANSendMessage	Operation	12
CANReceiveMessage	Operation	14
CANAbortAll	Operation	16
CANGetTxErrorCount	Status Check	17
CANGetRxErrorCount	Status Check	18
CANIsBusOff	Status Check	19
CANIsTxPassive	Status Check	20
CANIsRxPassive	Status Check	21
CANIsRxReady	Status Check	22
CANIsTxReady	Status Check	23

Microchip 的 AN738

- 使用 CAN Library , PIC18F458 的 CAN Module 設定更為簡單

```
CANInitialize(2,8,3,3,1, CAN_CONFIG_LINE_FILTER_OFF &  
              CAN_CONFIG_SAMPLE_ONCE &  
              CAN_CONFIG_STD_MSG &  
              CAN_CONFIG_VALID_STD_MSG &  
              CAN_CONFIG_PHSEG2_PRG_ON) ;
```

```
CANSetOperationMode(CAN_OP_MODE_CONFIG) ;
```

```
CANSetMask(CAN_MASK_B1, 0x000000FF, CAN_CONFIG_STD_MSG) ;  
CANSetMask(CAN_MASK_B2, 0x000000FF, CAN_CONFIG_STD_MSG) ;
```

```
CANSetFilter(CAN_FILTER_B1_F1, 0x00000000 , CAN_CONFIG_STD_MSG) ;  
CANSetFilter(CAN_FILTER_B1_F2, 0x00000000 , CAN_CONFIG_STD_MSG) ;  
CANSetFilter(CAN_FILTER_B2_F1, 0x0000002E , CAN_CONFIG_STD_MSG) ;  
CANSetFilter(CAN_FILTER_B2_F2, 0x00000000 , CAN_CONFIG_STD_MSG) ;  
CANSetFilter(CAN_FILTER_B2_F3, 0x00000000 , CAN_CONFIG_STD_MSG) ;  
CANSetFilter(CAN_FILTER_B2_F4, 0x00000000 , CAN_CONFIG_STD_MSG) ;
```

```
CANSetOperationMode(CAN_OP_MODE_NORMAL) ;
```

Microchip 的 AN738

- 使用 Functions Library 所需付出的代價
 - ◆ 除了對 Controller 的規格需有所瞭解外, 並且要參考到 Function Library 已經事先定義好的常數
 - ✧ CAN18XX8.h
 - ◆ 以 CANSetOperationMode (enum CAN_OP_MODE mode) 為例, 使用此 function 需參考以下的列舉資料型態

```
enum CAN_OP_MODE
```

```
{
```

```
    CAN_OP_MODE_BITS    = 0b11100000, // Use this to access opmode bits
```

```
    CAN_OP_MODE_NORMAL  = 0b00000000,
```

```
    CAN_OP_MODE_SLEEP   = 0b00100000,
```

```
    CAN_OP_MODE_LOOP    = 0b01000000,
```

```
    CAN_OP_MODE_LISTEN  = 0b01100000,
```

```
    CAN_OP_MODE_CONFIG  = 0b10000000
```

```
};
```

Ex1 : Configure PIC18F458 at 'Loopback Mode'

- 更進一步地瞭解 CAN Module

- ◆ Loopback Mode

- ✦ 這是一個有用的偵錯模式, CAN 的 device 可在不需 "On Bus" 的情況下直接由 TXCAN 將資料送至 RXCAN

- 所有的 Filter 及 Mask 都能正確地工作
- 可用來驗證裝置的操作原理與正確性
- 在不干擾 CAN Bus 的狀況下練習傳送與接收

- ◆ 送出一個簡單的 Message !!

- ✦ Loopback mode 允許信息(Message)在 Device 的內部傳送
- ✦ CAN Module 不將資料由 TXCAN 送至外部, 而是經由內部的路徑轉至 MAB
- ✦ 將 PIC18F458 作正確的設定, 且傳送出一個簡單的 Message
 - 傳送的格式為 STANDARD Data FRAME
 - 將 5 個 Bytes 的資料由 TXB0 送出
 - 01,02,03,04,ADRESH (Channel 0)

Ex1 : Configure PIC18F458 at 'Loopback Mode'

- 利用 AN738 的副程式來達成目標
 - ◆ CANInitialize()
 - ✦ 用來規劃 PIC18F458 的工作條件
 - ◆ CANSetOperationMode()
 - ✦ 用來設定 PIC18F458 的操作模式
 - ◆ CANSetMask()
 - ✦ 用來設定 RXM0 及 RXM1
 - ◆ CANSetFilter()
 - ✦ 用來設定各 Filter 的內容
 - ◆ CANSendMessage()
 - ✦ 用來將資料傳出

Ex1 : Configure PIC18F458 at 'Loopback Mode'

- 利用 AN738 的副程式來達成目標 (Cont.)
 - ◆ CANIsRxReady()
 - ✦ 用來判斷有無符合過濾條件的資料被接收進來
 - ◆ CANReceiveMessage()
 - ✦ 用來接收資料

Ex1 : Configure PIC18F458 at 'Loopback Mode'

- 其他有關 LCD 的副程式
 - ◆ **OpenLCD(void)**
 - ✦ 設定 LCD 於要求的工作模式
 - ◆ **LCD_Set_Cursor(unsigned char , unsigned char)**
 - ✦ 設定 LCD 的游標位置
 - ◆ **putsLCD(char *)**
 - ✦ 將置於 Data Memory 中的字串輸出至 LCD
 - ◆ **putrsLCD(const rom char *)**
 - ✦ 將至於 Program Memory 中的字串輸出至 LCD

Ex1 : Configure PIC18F458 at 'Loopback Mode'

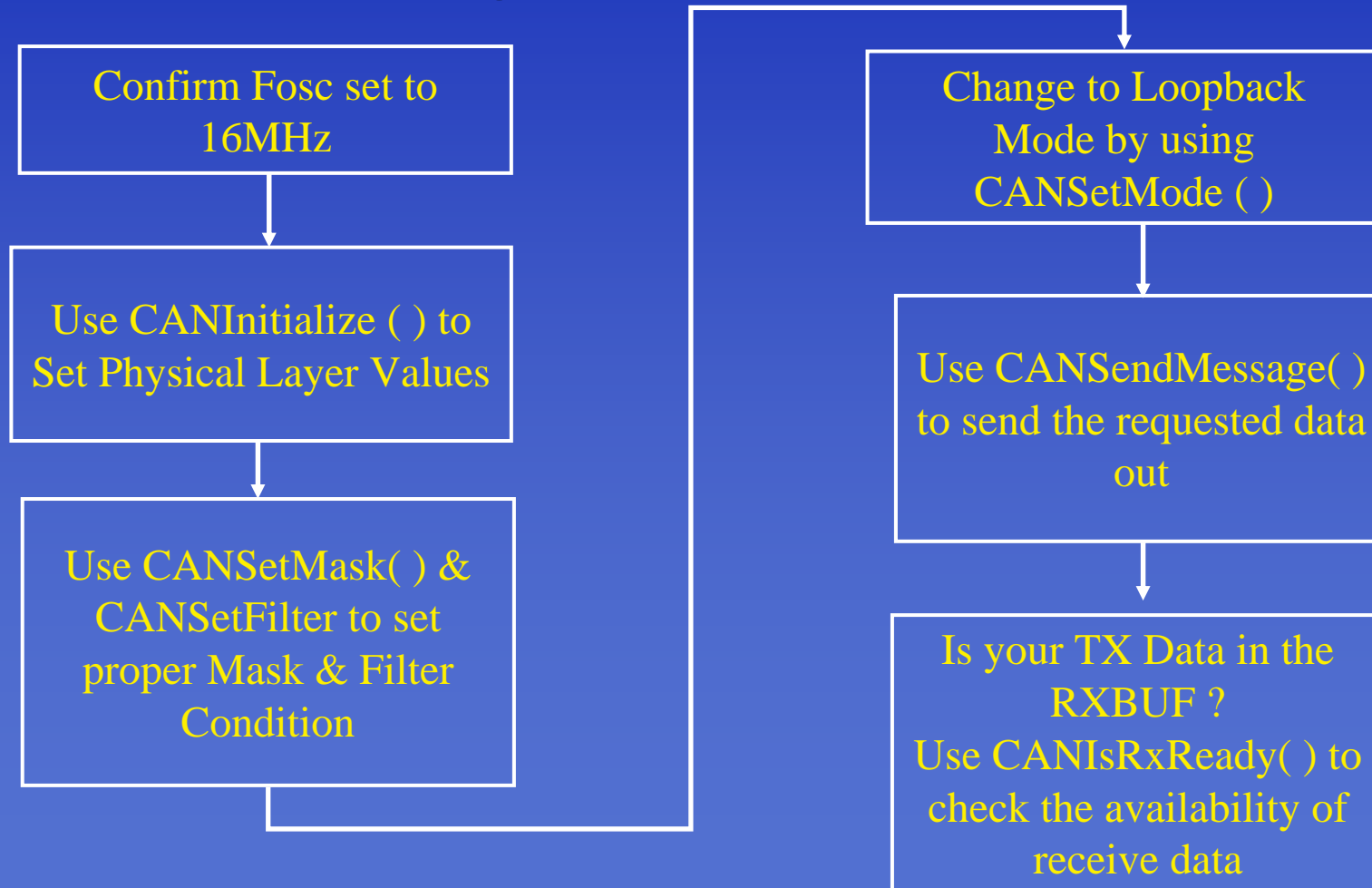
- 其他有關 LCD 的副程式 (Cont.)
 - ◆ putcLCD(unsigned char)
 - ✦ 將單一字元輸出至 LCD
 - ◆ puthexLCD(unsigned char)
 - ✦ 將一個位元組以 16 進位的格式顯示於 LCD
- LCD 各副程式的原形(Prototype)宣告於
 - ◆ WAP_LCD.h
- LCD 副程式的原始程式置於
 - ◆ WAP_LCD.C

APP001 實驗板



Ex1 : Configure PIC18F458 at 'Loopback Mode'

● Simple Message in Loopback Mode



Ex1 : Configure PIC18F458 at 'Loopback Mode'

- Simple Message in Loopback Mode
 - ◆ 您的 RXB0 會收到自己送出的資料嗎？
 - ✦ 正常的情形下不會！ Why？
 - ◆ Masks 及 Filters 間的關係為：
 - ✦ 只有那些符合資料過濾原則的 Message 才會到達 RXBx
 - ✦ 解決方式：
 - 將 MASK 及 Filter 的值作正確的設定
 - 關掉 Masks 和 Filters 的檢查
 - Set RXBxCON.RXM<1:0> bits to "11"
- 將 MESSAGE_ID1 和 RX_Filter0 設成一樣的值, 再重複一次
 - ✦ CANIsRxReady () 將會在收到自己 loopback 的資料後傳回非 "0" 的值來指出至少有一筆資料被正確接收到！

Ex1 : Configure PIC18F458 at 'Loopback Mode'

- 資料進入 RXB0 後的處置 < if CANIsRxReady() != 0 >
 - ◆ 使用 CANReceiveMessage() 將 RXB0 的資料讀入
 - ◆ 將 RX_ID1 顯示於 LCD Display 的 (4,0)
 - ◆ 將 RX_Data_Buf[3] & [4] 顯示於 LCD Display 的 (12,0)
- 若一切正常工作, 在 LCD Display 將有以下的資料顯示
 - ◆ (12,1) 顯示 10 進位的 A/D 轉換值 [0..1023]
 - ◆ (12,0) 顯示 16 進位的接收資料 [0..03FF]
 - ◆ (4,0) 顯示接收到的 Message ID

深入探討 PIC18F458 CAN Module & Microchip AN738 for PIC18XXXX

PIC18F458 CAN Module Initialization

CANCON - CAN Control Register

R/W-1	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0
REQOP2	REQOP1	REQOP0	ABAT	WIN2	WIN1	WIN0	
bit 7							bit 0

- CAN Module 的操作模式決定於 CANCON 的 Bit5 to Bit7
 - ◆ REQOP<2:0> requests different operational modes
 - ✦ Configuration, Listen Only, Loopback, Disable, Normal
- ABAT 將取消所有 Transmit Buffer 中處於 pending 的 messages 之傳送要求
- WIN<2:0> 會將 Bank 15 中的 CAN registers F20h - F2Eh, F30h - F3Eh, F40h - F4Eh, or F50h - F5Eh 對應到 access bank 中較高的位址 F60h - F76h
 - ◆ 讓這些暫存器的讀寫更有效率 > 直接透過 Access Bank !!

CAN Module Initialization

CANSTAT- CAN Status Register

R-1	R-0	R-0	U-0	R-0	R-0	R-0	U-0
OPMODE2	OPMODE1	OPMODE0		ICODE2	ICODE1	ICODE0	
bit 7							bit 0

- OPMODE<2:0> 反應 CAN Module 現在所處的操作模式
 - ◆ Configuration, Listen Only, Loopback, Disable, Normal
- ICODE<2:0> returns current interrupt code bits
 - ◆ 可以將此 3 位元直接 copy 至 CANCON.WIN<2:0> bits 來加速對 buffer 存取的效率

ICODE<2:0>	Boolean Expression	Description
000	!ERR•!WAK•!TX0•!TX1•!TX2•!RX0•!RX1	No interrupt
001	ERR	Error interrupt
010	!ERR•!TX0•!TX1•TX2	Tx Buffer 2 Interrupt
011	!ERR•!TX0•TX1	Tx Buffer 1 Interrupt
100	!ERR•TX0	Tx Buffer 0 Interrupt
101	!ERR•!TX0•!TX1•!TX2•!RX0•RX1	Rx Buffer 1 Interrupt
110	!ERR•!TX0•!TX1•!TX2•RX0	Rx Buffer 0 Interrupt
111	!ERR•!TX0•!TX1•!TX2•!RX0•!RX1•WAK	Wake-up on Interrupt

CAN Module Initialization

Bit Timing 的詳細資訊

- Programmable Digital Phase Lock Loop (DPLL)
- Max node-to-node oscillator tolerance up to 1.7%
- 一個位元所佔的時間事實上是由許多Time Quanta (TQ) 組合而成 (Max. = 25)
 - ◆ Baud Rate Prescaler (BRP) 用來設定 TQ.
- 由四個 segments 組成一個 bit , 而每個 Segment 佔用 1 或多個 TQ
- 確實地符合 CAN bit timing 的規格需求
 - ◆ Sync segment 永遠只佔用 1 個 TQ

Sync	PRSEG		SEG1PH			SEG2PH		
TQ	TQ	TQ	TQ	TQ	TQ	TQ	TQ	TQ
1TQ	1-8TQ		1-8TQ			1-8TQ		

CAN Module Initialization

BRGCON1 - Baud Rate Control Register 1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0
bit 7							bit 0

- SJW<1:0> 用來設定 (Re)Synchronization Jump Width
- BRP<5:0> 定義 : Baud Rate Prescale
 - ◆ 調整 BRP 可設定 TQ. $TQ = 2 * (BRP + 1) / F_{osc}$
 - ◆ 適切的規劃 BRP 的值讓每一位元時間所使用的 TQ 數量能 ≥ 8 TQ
 - ◆ Shorter TQ (i.e. More TQ/bit) provides robust bit rate

CAN Module Initialization

BRGCON2 - Baud Rate Control Register 2

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SEG2PHTS	SAM	SEG1PH3	SEG1PH1	SEG1PH0	PRSEG2	PRSEG1	PRSEG0
bit 7							bit 0

- SEG2PHTS 決定如何來規劃 SEG2PH (Phase Segment 2)
- SAM 定義每一位元的取樣次數 (1 or 3)
- SEG1PH<2:0> 定義 phase segment 1 的長度
 - ◆ Binary 的數值 0 - 7 對應至 1 x TQ - 8 x TQ
 - ◆ PRSEG + SEG1PH 必須 \geq SEG2PH

CAN Module Initialization

BRGCON2 - Continued

- PRSEG<1:0> defines Propagation segment width
 - ◆ Binary values of 0 - 7 corresponds to 1 x TQ - 8 x TQ
 - ◆ Compensates physical delay in network
 - ◆ $PRSEG + SEG1PH \geq SEG2PH$
 - ◆ $PRSEG + SEG1PH \geq T_{delay}$

CAN Module Initialization

BRGCON3 - Baud Rate Control Register 3

U-0	R/W-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
	WAKFIL				SEG2PH2	SEG2PH1	SEG2PH0
bit 7							bit 0

- WAKFIL enables/disables CAN bus line filter for wake-up
- SEG2PH<2:0> defines phase segment 2 width
 - ◆ $\text{SEG2PH} \geq \text{SJW}$
 - ◆ $\text{PRSEG} + \text{SEG1PH} \geq \text{SEG2PH}$
 - ◆ Overlaps with SJW and shrinks as SJW changes

CAN Module Initialization

Bit Rate Register Values

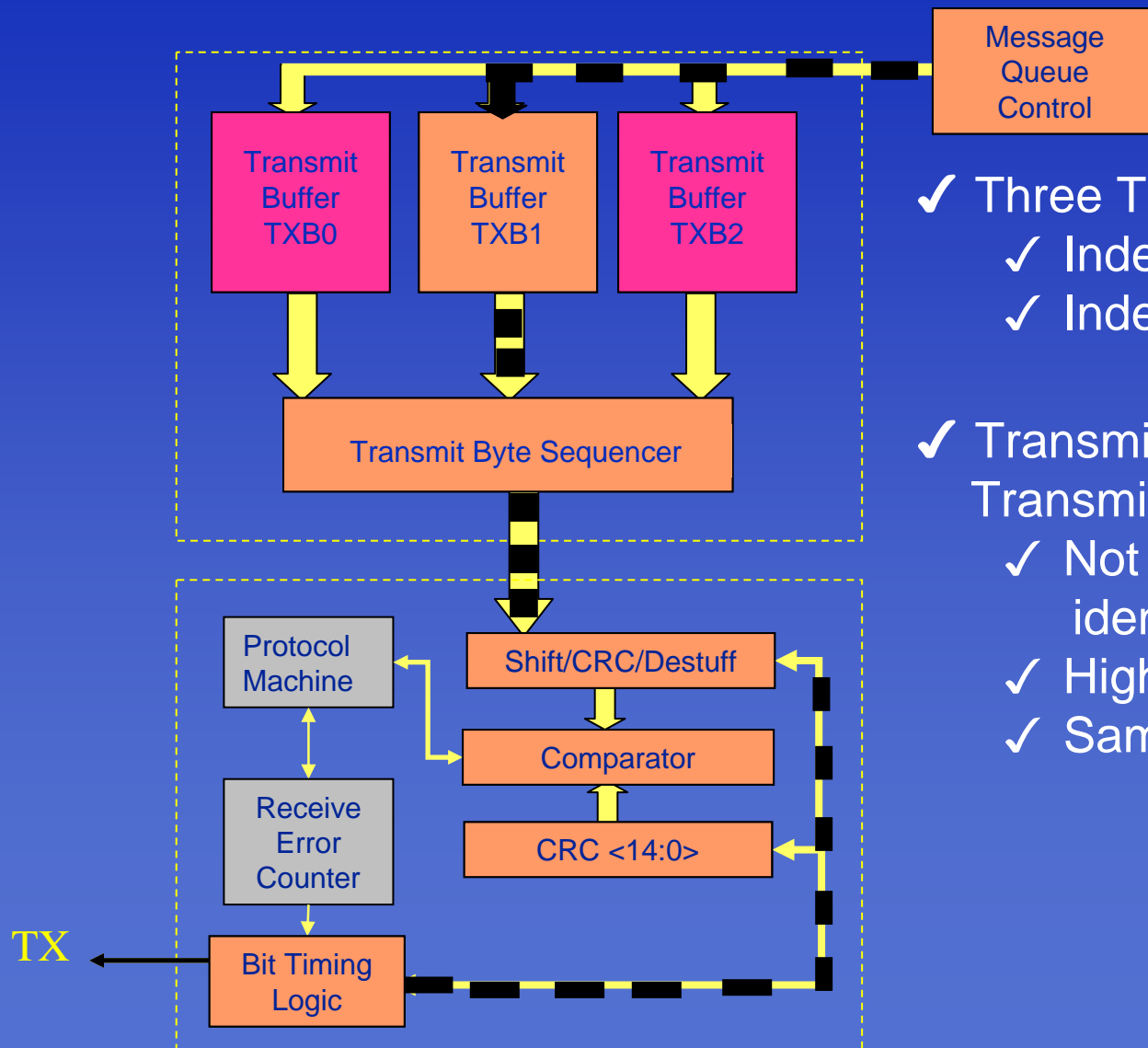
- For a baud rate of B at osc frequency of Fosc,
 - ◆ $T_{bit} = 1 / B$
 - ◆ $T_q = 2 * (BRP + 1) * T_{osc}$
 - ◆ $T_q = T_{bit} / N$, where N is Tq count per nominal bit
 - ◆ $BRP = ((F_{osc}/N) / (2 * B)) - 1$
 - ◆ Select a maximum N such that $(F_{osc}/N) / (2 * B)$ is near integer value, where $8 \leq N \leq 25$
 - ◆ Calculate BRP
 - ◆ Phase Segment 2 = SJW = $N * 0.30$ (Integer operation)
 - ◆ Prop Segment = 2 (May be increased as per length)
 - ◆ Phase Segment1 = $N - 1 - 2 - \text{Phase Segment 2}$

CAN Module Initialization

Bit Rate Register Values - Example

- Configure 125 kbps at 16 MHz
 - ◆ $T_{bit} = 1 / 125 \text{ kbps} = 8 \text{ } \mu\text{S}$
 - ◆ Selecting $N = 16$,
 - ✦ $(F_{osc}/N) / (2 * B) = (16\text{MHz}/16)/(2 * 125\text{kbps}) = 4$
 - ◆ $BRP = 4 - 1 = 3$
 - ◆ Phase Segment 2 = SJW = $16 * 0.30 = 5$
 - ◆ Prop Segment = 2
 - ◆ Phase Segment 1 = $16 - 1 - 2 - 5 = 8$
- Results: $BRP<5:0> = 3$, $SEG1PH = 7$, $SEG2PH = 4$, $SJW = 4$

Transmitter Buffers @ PIC18F458 CAN Module



- ✓ Three Transmit Buffers
 - ✓ Independent Selectable Priority
 - ✓ Independent Abort
- ✓ Transmit Control Register Sets Transmit Priority
 - ✓ Not to be confused with identifier message priority
 - ✓ Higher number = higher priority
 - ✓ Same priority TXPRI => higher buffer number has priority

CAN Module Buffer Access

TXBnCON - Transmit Buffer n Control Register

U-0	R-0	R-0	R-0	R/W-0	U-0	R/W-0	R/W-0
	TXABT	TXLARB	TXERR	TXREQ		TXPRI1	TXPRI0
bit 7						bit 0	

- Three transmit buffers ($0 \leq n \leq 2$)
- TXABT returns transmission abort status
- TXLARB returns arbitration status
- TXERR returns bus error status
- TXREQ requests transmission of current message
- TXPRI<1:0> defines priority of current message
 - ◆ Higher the number, higher the priority

CAN Module Buffer Access

TXBnSIDH - Transmit Buffer n Std. Identifier High

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3
bit 7							bit 0

- SID<10:3> defines Standard/Extended Identifier bits

{ SID<10:3> }

SOF	SID10...SID3	SID2...SID0	...
-----	--------------	-------------	-----

OR

{ SID<10:3> }

SOF	EID28...EID21	EID20...EID18	...
-----	---------------	---------------	-----

CAN Module Buffer Access

TXBnSIDL - Transmit Buffer n Std. Identifier Low

R/W-x	R/W-x	R/W-x	U-0	R/W-x	U-0	R/W-x	R/W-x
SID2	SID1	SID0		EXIDE		EID17	EID16
bit 7						bit 0	

- EXIDE declares current filter as Std./Extd.
- SID<2:0> defines Std./Extd. Identifier bits



OR



CAN Module Buffer Access

TXBnEIDH - Transmit Buffer n Extd. Identifier High

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8
bit 7							bit 0

- EID<15:8> defines Extended Identifier bits
 - Used for Extended Identifier value only



CAN Module Buffer Access

TXBnEIDL - Transmit Buffer n Extd. Identifier Low

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0
bit 7							bit 0

- EID<7:0> defines Extended Identifier bits
 - Used for Extended Identifier value only



CAN Module Buffer Access

TXBnDLC - Transmit Buffer n Data Length

U-0	R/W-x	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x
	TXRTR			DLC3	DLC2	DLC1	DLC0
bit 7				bit 0			

- TXRTR defines current message as RTR
 - RTR message has 0 data length
- DLC<3:0> defines length of current message
 - Valid values are 0 - 8

CAN Module Buffer Access

TXBnDm - Transmit Buffer n Data m

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
TXBnDm7	TXBnDm6	TXBnDm5	TXBnDm4	TXBnDm3	TXBnDm2	TXBnDm1	TXBnDm0
bit 7							bit 0

- PIC18F452 有 3 個 Transmitter
- Transmit Buffer 分別為
 - TXB0D0 to TXB0D7 for Transmitter Buffer 0
 - TXB1D0 to TXB1D7 for Transmitter Buffer 1
 - TXB2D0 to TXB2D7 for Transmitter Buffer 2

CAN Module Buffer Access

如何起始資料傳送

- 確定 TXBnCON.TXREQ = 0
- 將 Message ID 寫入 TxBnSIDL, TxBnSIDH, TXBnEIDL, TXBnEIDH 並決定是 Standard 或 Extended Message
 - ◆ TXBnSIDL.EXIDE 位元要正確地設定
- 將欲傳出的資料正確地寫入TXBnDm registers
 - ◆ 最多 8 個 Bytes
- 將 TXBnDLC 填入欲傳送的資料長度
- 依照資料的重要性設定 TXBnCON.TXPRI, 以便決定其優先順序
- 將 Transmitter Buffer 對應之 TXBnCON.TXREQ bit 設定為 “1”
 - ◆ Message will be transmitted on next bus idle time

AN738 CAN Subroutine

Part 1

AN738 CAN Subroutine – Part1

- 以下是要起始及進行 CAN 資料傳送時必須使用到的 CAN Subroutine
 - ◆ CANInitialize()
 - ◆ CANSetOperationMode()
 - ◆ CANSendMessage()
 - ◆ CANIsTxReady()

AN738 CAN Subroutine – CANInitialize()

- CANInitialize (BYTE SJW , BYTE BRP ,
BYTE PHSEG1 , BYTE PHSEG2 ,
BYTE PROPSEG ,
enum CAN_CONFIG_FLAGS config) ;
- 依據指定的參數對 CAN Module 進行初始化的工作
 - ◆ SJW : 1 through 4
 - ◆ BRP : 1 through 64
 - ◆ PHSEG1 : 1 through 8
 - ◆ PHSEG2 : 1 through 8
 - ◆ PROPSEG : 1 through 8
 - ◆ enum CAN_CONFIG_FLAG : See next Page

AN738 CAN Subroutine – CANInitialize()

- 可指定符合 CAN_CONFIG_FLAGS 型態的列舉值，傳入 CANInitialize()，可以將列表中的項目任意組合
- 各項設定值可用 “&” 來連結

Value	Meaning
CAN_CONFIG_DEFAULT	Specifies default flags
CAN_CONFIG_PHSEG2_PRG_ON	Specifies to use supplied PHSEG2 value
CAN_CONFIG_PHSEG2_PRG_OFF	Specifies to use maximum of PHSEG1 or Information Processing Time (IPT), whichever is greater
CAN_CONFIG_LINE_FILTER_ON	Specifies to use CAN bus line filter for wake-up
CAN_CONFIG_LINE_FILTER_OFF	Specifies to not use CAN bus line filter for wake-up
CAN_CONFIG_SAMPLE_ONCE	Specifies to sample bus once at the sample point
CAN_CONFIG_SAMPLE_THRICE	Specifies to sample bus three times prior to the sample point
CAN_CONFIG_ALL_MSG	Specifies to accept all messages including invalid ones
CAN_CONFIG_VALID_XTD_MSG	Specifies to accept only valid Extended Identifier messages
CAN_CONFIG_VALID_STD_MSG	Specifies to accept only valid Standard Identifier messages
CAN_CONFIG_ALL_VALID_MSG	Specifies to accept all valid messages
CAN_CONFIG_DBL_BUFFER_ON	Specifies to hardware double buffer Receive Buffer 1
CAN_CONFIG_DBL_BUFFER_OFF	Specifies to not hardware double buffer Receive Buffer 1

AN738 CAN Subroutine – CANSetOperationMode()

- CANSetOperationMode (enum CAN_OP_MDOE mode)
- 可指定符合 CAN_OP_MODE 型態的列舉值，傳入 CANSetOperationMode()，可以將列表中的任一系列值傳入
- 當執行 CANSetOperationMode () 時，正處於 Pending 的 Message 會被 Abort !!

Value	Meaning
CAN_OP_MODE_NORMAL	Specifies Normal mode of operation
CAN_OP_MODE_SLEEP	Specifies SLEEP mode of operation
CAN_OP_MODE_LOOP	Specifies Loopback mode of operation
CAN_OP_MODE_LISTEN	Specifies Listen Only mode of operation
CAN_OP_MODE_CONFIG	Specifies Configuration mode of operation

AN738 CAN Subroutine – CANSendMessage()

- **CANSendMessage** (unsigned long id , BYTE *Data
BYTE DataLen,
enum CAN_TX_MSG_FLAGS MsgFlags);
- 參數的定義如下
 - ◆ id : 要送出資料的 message ID
 - ◆ *Data : 資料指標，指向欲傳送資料的起始位址
 - ◆ DataLen : 要傳送的資料長度
 - ◆ MsgFlags : 符合 CAN_TX_MSG_FLAG 型態的列舉值集合
 - 有效的列舉值如下所示

AN738 CAN Subroutine – CANSendMessage()

- 符合 CAN_TX_MSG_FLAG 型態的列舉值如下
 - ◆ 各列舉值可用 “&” 來結合

Priority Value	Meaning
CAN_TX_PRIORITY_0	Specifies Transmit Priority 0
CAN_TX_PRIORITY_1	Specifies Transmit Priority 1
CAN_TX_PRIORITY_2	Specifies Transmit Priority 2
CAN_TX_PRIORITY_3	Specifies Transmit Priority 3
Note: See the PIC18CXX8 data sheet for further details on transmit priority.	

Identifier Value	Meaning
CAN_TX_STD_FRAME	Specifies Standard Identifier message
CAN_TX_XTD_FRAME	Specifies Extended Identifier message

Message Value	Meaning
CAN_TX_NO_RTR_FRAME	Specifies Regular message - not RTR
CAN_TX_RTR_FRAME	Specifies RTR message

AN738 CAN Subroutine – CANIsTxReady()

- **BOOL CANIsTxReady(void) ;**
 - ◆ 檢查是否有任何 TX Buffer 是可被寫入資料的
 - ◆ 傳回值：
 - ✦ TRUE : 至少有一個 CAN Transmit Buffer 是空的
 - ✦ FALSE : 沒有任何 CAN Transmit Buffer 是空的
- 傳送前最好先利用此函數來先判定是否有可用的 Transmit Buffer
 - If (CANIsTxReady())

MCP250XX Developer's Kit



MCP250XX Developer's Kit

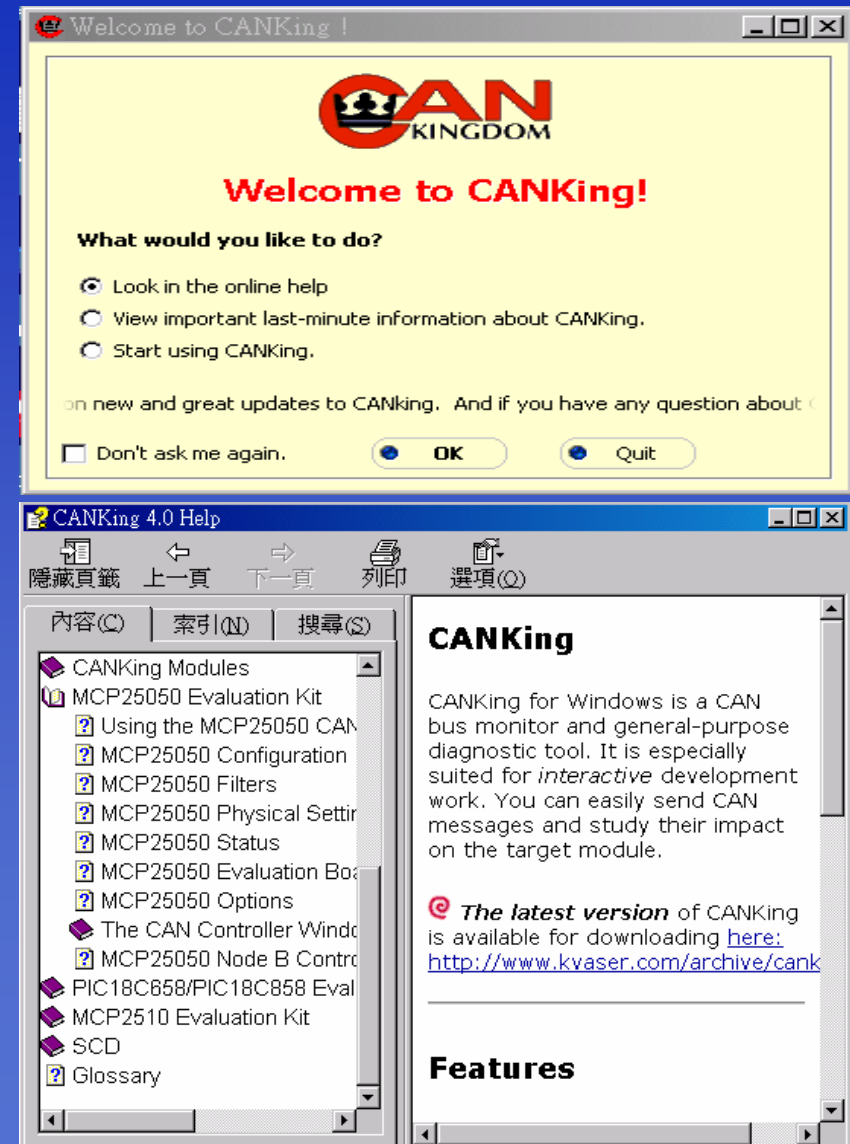
- 提供開發者快速且簡易的環境
 - ◆ 直接對 MCP250XX 作設定及燒錄
 - ◆ 可透過 CANKing 軟體作為簡易的 Bus Monitor
 - ◆ MCP250XX 可直接於 Kit 上工作以便驗證開發者的設定是否正確
 - ✧ A/D 輸入信號
 - ✧ PWM 輸出信號
 - ✧ Digital I/O
 - ◆ 提供連接其他外部 CAN Node 的接頭, 以便連接至其他 CAN Node 來進行對 CAN Network 的資料擷取
- 與 PC 的介面 – Parallel PORT

What's CANKing for Window

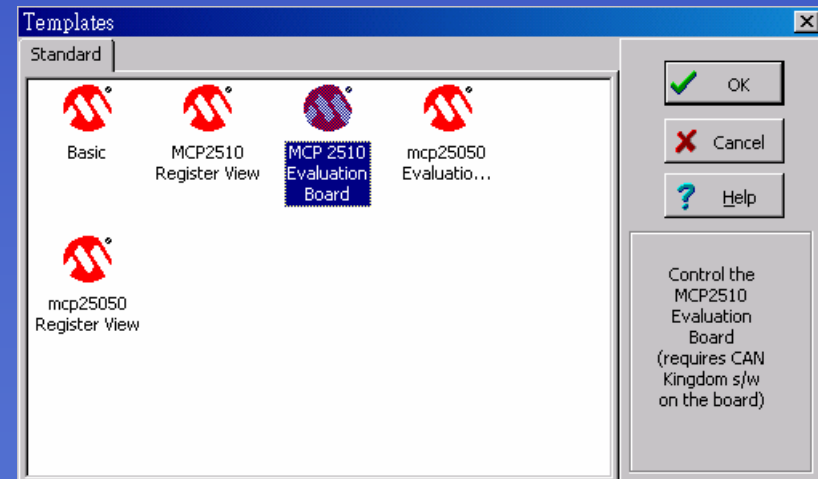
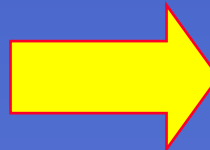
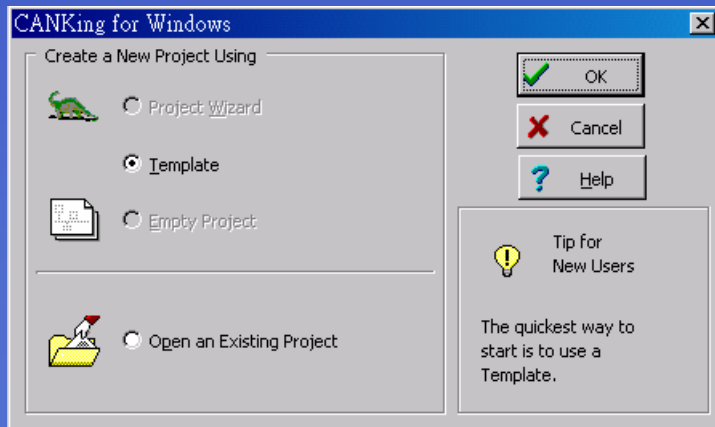
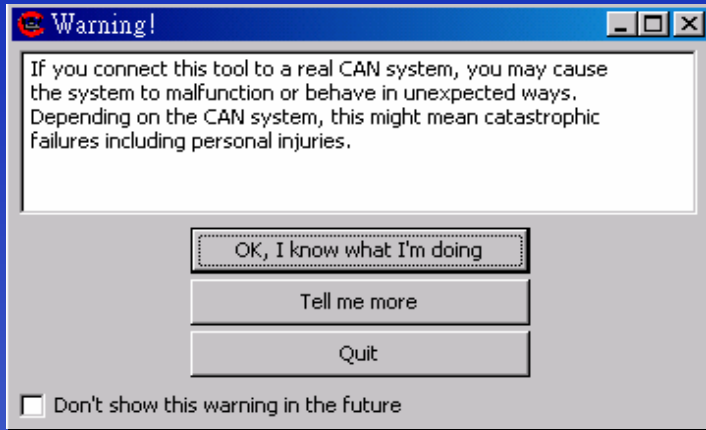
- 可搭配 MCP250XX , MCP2510 , PIC18C658/858 的 Evaluation Kit 來輔助 CAN Application 的開發
 - ◆ 可作為 CAN BUS Monitor
 - ◆ 可以是泛用型的除錯工具
 - ✧ 可設定欲使用的 Bus Speed
 - ✧ 可直接送出資料封包以便觀察 CAN Node 的動作與反應

How CANKing Work

- 執行 CANKing 後將出現如右方的選項
 - ◆ 選取第一項可以讀取各項功能的說明
 - ◆ 第二項能列出重要的版本及功能資訊
 - ◆ 使用第三項便直接進入 CANKing 程式



How CANKing Work

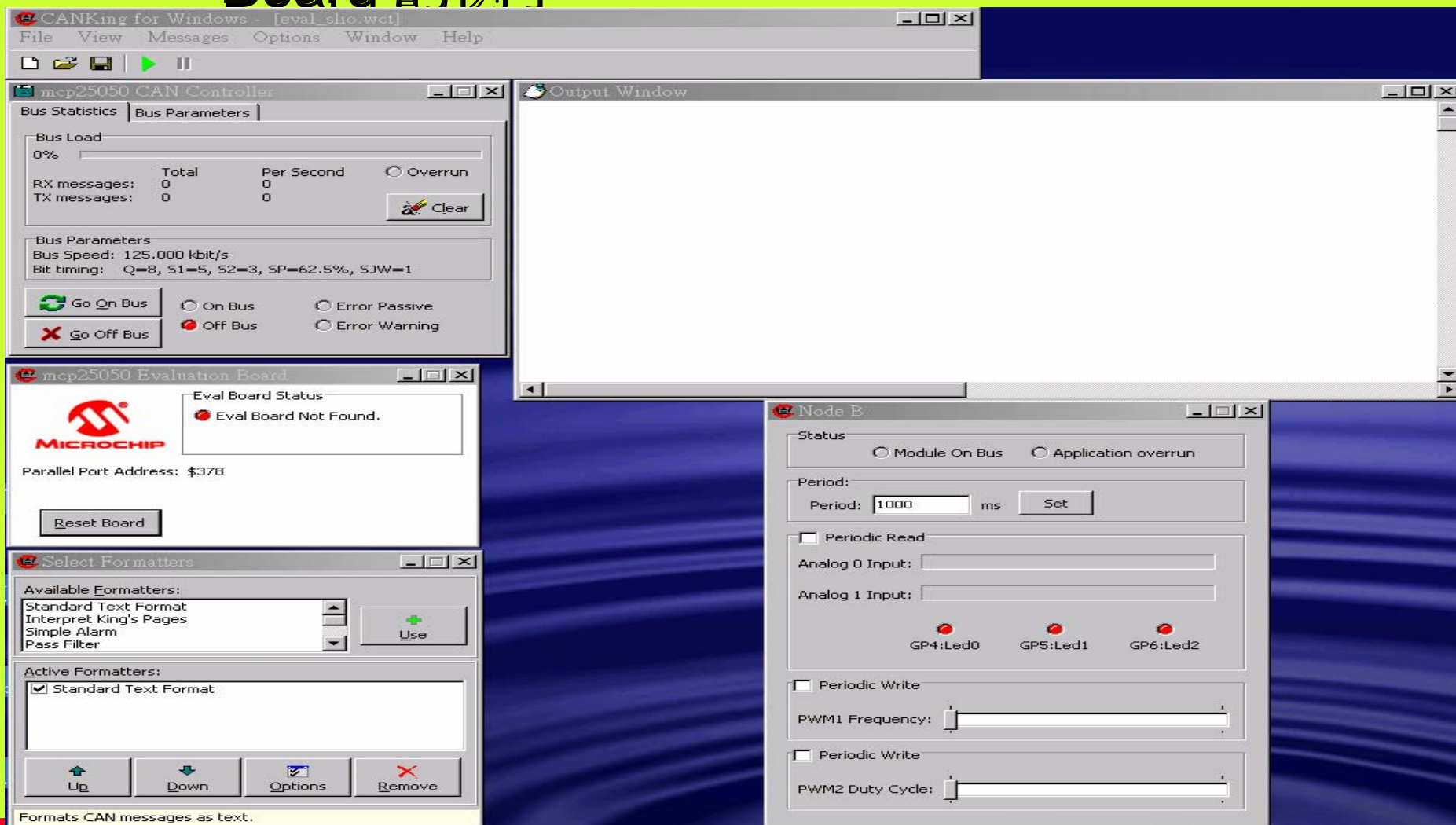


•CANKing 程式的執行順序

- Open an Existing Project 可開啓已經存在的專案檔
- 使用 Template 選項可開啓已經存在的幾個範本

How CANKing Work

使用 Template – MCP25050 Evaluation Board 的例子



Ex2 : GO On the Bus

- 將 EX1.C 中 CANInitialization() 中的內容作適當的修改
 - ◆ 將參數 CAN_OP_MODE_LOOP 改成
"CAN_OP_MODE_NORMAL"
 - ◆ 將 MESSAGE_ID1 依照指定作修改 (0x100 .. 0x150)
 - ◆ 把檔案存成 EX2.C

Ex2 : GO On the Bus

- 重新 Compiler 並執行後，PIC18F458 TXB0 的資料將以程式中的 MESSAGE_ID1 為 Message ID 被送出
- 因為在 NORMAL MODE 中，故 PIC18F458 將不再收到自己送出的資料
- 使用 CANKing，再 PC 的螢幕中將可以觀察到各個 CAN Node 傳來的資料
 - ◆ 0x100 , 0x110 , 0x120 , 0x130 , 0x140 , 0x150

Receive Buffers @ PIC18F458 CAN Module

PIC18F458 Receive Buffer 的相關設定

- 要正確地接收資料，必須正確地設定以下參數
 - ◆ Filters
 - ✧ RXFnSIDH, RXFnSIDL, RXFnEIDH, RXFnEIDL
 - ◆ Mask
 - ✧ RXMnSIDH, RXMnSIDL, RXMnEIDH , RXMnEIDL
 - ◆ Buffer Mode
 - ✧ RXBnCTRL.RXM <1:0>
 - ✧ 決定要接收何種 Message !
 - All Valid Standard , All Valid Extended , Both Valid Standard/Extended or All Message
 - ◆ 最重要的參數 : Bit Rate (通信速度)

CAN Module Buffer Access

RXB0CON - Receive Buffer 0 Control Register

R/C-0	R/W-0	R/W-0	U-0	R-0	R/W-0	R-0	R-0
RXFUL	RXM1	RXM0		RXRTRR0	RXB0DBEN	JTOFF	FILHIT0
bit 7							bit 0

- RXFUL returns/clears buffer full status
 - ◆ Set by CAN FSM and cleared by software
- RXM<1:0> defines buffer mode
 - ◆ To receive all valid, only Std., only Extd. Or all messages including those with errors
- RXRTRR0 returns RTR status
- RXB0DBEN defines hardware double-buffer option
 - ◆ Buffer 0 overflow will load Buffer 1

CAN Module Buffer Access

RXB0CON - Continued

- JTOFF (Jump Table OFFset) maps Buffer 0 filter hit information to that of Buffer 1
 - ◆ To be reviewed in next slide
- FILHIT0 returns filter number which caused receive
 - ◆ RXF0 or RXF1

CAN Module Buffer Access

Receive Buffer Modes

- Defined by RXBnCON.RXM<1:0> bits
 - ◆ “Receive All (Including Invalid)”, “Receive Valid Extd.”, “Receive Valid Std.”, “Receive All Valid”
- “Receive All (Including Invalid)”
 - ◆ Acceptance mask/filter values are overridden
 - ◆ Provides Basic CAN functionality
- “Receive Valid Extd./Std.”
 - ◆ RXFnSIDL.EXIDEN must match
- “Receive All Valid”
 - ◆ RXFnSIDL.EXIDEN is used

CAN Module Buffer Access

RXB1CON - Receive Buffer 1 Control Register

R/C-0	R/W-0	R/W-0	U-0	R-0	R-0	R-0	R-0
RXFUL	RXM1	RXM0		RXRTRR0	FILHIT2	FILHIT1	FILHIT0
bit 7							bit 0

- RXFUL returns/clears buffer full status
 - ◆ Set by CAN FSM and cleared by software
- RXM<1:0> defines buffer mode
 - ◆ To receive all valid, only Std., only Extd. Or all messages including those with errors
- RXRTRR0 returns RTR status
- FILHIT<2:0> returns filter number which caused receive
 - ◆ RXF0, RXF1, RXF2, RXF3, RXF4, RXF5

CAN Module Initialization

RxFnSIDH

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3
bit 7							bit 0

- SID<10:3> defines Standard/Extended Identifier Filter bits

{ SID<10:3> }

SOF	SID10...SID3	SID2...SID0	...
-----	--------------	-------------	-----

OR

{ SID<10:3> }

SOF	EID28...EID21	EID20...EID18	...
-----	---------------	---------------	-----

CAN Module Initialization

RxFnSIDL

R/W-x	R/W-x	R/W-x	U-0	R/W-x	U-0	R/W-x	R/W-x
SID2	SID1	SID0		EXIDEN		EID17	EID16
bit 7							bit 0

- EXIDEN declares current filter as Std./Extd.
- SID<2:0> defines Std./Extd. Identifier Filter bits

{ SID<2:0> } { EXIDEN }

SOF	SID10...3	SID2...0	RTR	IDE	...
-----	-----------	----------	-----	-----	-----

OR

{ SID<2:0> } { EXIDEN } EID<17,16> }

SOF	...	EID20...18	SRR	IDE	EID17, 16	...
-----	-----	------------	-----	-----	-----------	-----

CAN Module Initialization

RXFnEIDH

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8
bit 7							bit 0

- EID<15:8> defines Extended Identifier Filter bits
 - Used for Extended Identifier Filter value only



CAN Module Initialization

RxFnEIDL

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0
bit 7							bit 0

- EID<7:0> defines Extended Identifier Filter bits
 - Used for Extended Identifier Filter value only

{ EID<7:0> }

SOF	EID28...18	CTRL BITS	EID17, 8	EID7...0	...
-----	------------	-----------	----------	----------	-----

CAN Module Initialization

RXMnSIDH

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3
bit 7							bit 0

- SID<10:3> defines Standard/Extended Identifier Mask bits

{ SID<10:3> }

SOF	SID10...SID3	SID2...SID0	...
-----	--------------	-------------	-----

OR

{ SID<10:3> }

SOF	EID28...EID21	EID20...EID18	...
-----	---------------	---------------	-----

CAN Module Initialization

RXMnSIDL

R/W-x	R/W-x	R/W-x	U-0	U-0	U-0	R/W-x	R/W-x
SID2	SID1	SID0				EID17	EID16
bit 7						bit 0	

- No equivalent “EXIDEN”
- SID<2:0> defines Std./Extd. Identifier Masks,
EID<17,16> defines Extd. Identifier Masks only

{ SID<2:0> }

SOF	SID10...3	SID2...0	RTR	IDE	...
-----	-----------	----------	-----	-----	-----

OR

{ SID<2:0> }

{ EID<17,16> }

SOF	...	EID20...18	SRR	IDE	EID17, 16	...
-----	-----	------------	-----	-----	-----------	-----

CAN Module Initialization

RXMnEIDH

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8
bit 7							bit 0

- EID<15:8> defines Extended Identifier Mask bits
 - Used for Extended Identifier Mask value only

{ EID<15:8> }

SOF	EID28...18	CTRL BITS	EID17, 16	EID15...8	...
-----	------------	-----------	-----------	-----------	-----

CAN Module Initialization

RXMnEIDL

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0
bit 7							bit 0

- EID<7:0> defines Extended Identifier Mask bits
 - Used for Extended Identifier Mask value only

{ EID<7:0> }

SOF	EID28...18	CTRL BITS	EID17, 8	EID7...0	...
-----	------------	-----------	----------	----------	-----

CAN Module Buffer Access

RXBnSIDH - Receive Buffer n Std. Identifier High

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3
bit 7							bit 0

- SID<10:3> defines Standard/Extended Identifier bits

{ SID<10:3> }

SOF	SID10...SID3	SID2...SID0	...
-----	--------------	-------------	-----

OR

{ SID<10:3> }

SOF	EID28...EID21	EID20...EID18	...
-----	---------------	---------------	-----

CAN Module Buffer Access

RXBnSIDL - Receive Buffer n Std. Identifier Low

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	U-0	R/W-x	R/W-x
SID2	SID1	SID0	SRR	EXID		EID17	EID16
bit 7						bit 0	

- EXID declares current messages as Std./Extd.
- SRR indicates RTR when EXID = 0, else meaningless
- SID<2:0> defines Std./Extd. Identifier bits

$\left\{ \text{SID}<2:0> \right\} \quad \left\{ \text{EXID} \right\}$

SOF	SID10...3	SID2...0	RTR	IDE	...
-----	-----------	----------	-----	-----	-----

OR

$\left\{ \text{SID}<2:0> \right\} \quad \left\{ \text{EXID} \right\} \quad \left\{ \text{EID}<17,16> \right\}$

SOF	...	EID20...18	SRR	IDE	EID17, 16	...
-----	-----	------------	-----	-----	-----------	-----

CAN Module Buffer Access

RXBnEIDH - Receive Buffer n Extd. Identifier High

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8
bit 7						bit 0	

- EID<15:8> defines Extended Identifier bits
 - Used for Extended Identifier value only



CAN Module Buffer Access

RXBnEIDL - Receive Buffer n Extd. Identifier Low

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0
bit 7							bit 0

- EID<7:0> defines Extended Identifier bits
 - Used for Extended Identifier value only



CAN Module Buffer Access

RXBnDLC - Receive Buffer n Data Length

U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	RXRTR	RB1	RB0	DLC3	DLC2	DLC1	DLC0
bit 7							bit 0

- RXRTR defines current message as RTR
 - RTR message has 0 data length
- RB1, RB0 reserved by CAN spec, read '0'
- DLC<3:0> defines length of current message
 - Valid values are 0 - 8

CAN Module Buffer Access

RXBnDm - Receive Buffer n Data m

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
RXBnDm7	RXBnDm6	RXBnDm5	RXBnDm4	RXBnDm3	RXBnDm2	RXBnDm1	RXBnDm0
bit 7							bit 0

- 兩個 Receive Buffer , 所佔用位址分別為
 - Buffer 0 : RXB0D0 to RXB0D7 (0xF66 – 0xF6D)
 - Buffer 1 : RXB1D0 to RXB1D7 (0xF56 – 0xF5D)
- RXB0 相關的 ID Registers 及 Data Buffer 佔用的為址為 0xF60-0xF6E (It's **Access Bank**)
- RXB1 可以透過正確設定 CANCON.WIN< 2:0 > 來使原先 0xF50 – 0xF5E 的位址對應至 0xF60-0xF6E

CAN Module Buffer Access

如何判定接收

- 當 `RXBnCON.RXFUL` 或 `PIR3.RXBnIF = 1`
- 若 `PIE3.RXBnIE = 1` 時將會引發中斷
- 經由 `RXBnDm` 將傳至 Receive Buffer 的資料取出 (最多 8 Bytes)
- 讀出 Message ID 及其他相關資訊
- 清除 `RXBnCON.RXFUL` 位元
 - ◆ `RXBnCON.RXFUL` 被清除之後，CAN module 才可繼續由 **MAB** 接收新的資料

AN738 CAN Subroutine

Part 2

AN738 CAN Subroutine – Part2

- 以下是要起始及進行 CAN 資料傳送時必須使用到的 CAN Subroutine
 - ◆ CANSetMask()
 - ◆ CANSetFilter()
 - ◆ CANIsRxReady()
 - ◆ CANReceiveMessage()

AN738 CAN Subroutine – CANSetMask()

- CANSetMask (enum CAN_MASK code ,
 unsigned long Value ,
 enum CAN_CONFIG Type) ;
- 依據指定的參數對 CAN Module 進行MASK的設定工作
 - ◆ code : 定義於列舉型別 CAN_MASK 中的列舉值
 - ✦ 定義要設定哪一個 Buffer 的 MASK
 - ✦ CAN_MASK_B1 or CAN_MASK_B2
 - ◆ Value : long 資料型別的 MASK 值, 對應於 11 or 29 bits 的 Message ID 值
 - ◆ Type : 定義於列舉型別 CAN_CONFIG 中的列舉值
 - ✦ 定義要設定為 Standard 或 Extended ID 的 Mask
 - ✦ CAN_CONFIG_STD_MSG or CAN_CONFIG_XTD_MSG

AN738 CAN Subroutine – CANSetFilter()

- CANSetFilter (enum CAN_FILTER code ,
 unsigned long Value ,
 enum CAN_CONFIG Type) ;
- 依據指定的參數對 CAN Module 進行 Filter 的設定工作
 - ◆ code : 定義於列舉型別 CAN_FILTER 中的列舉值
 - ✦ 定義要設定哪一個 Buffer 的 Filter
 - ✦ CAN_FILTER 列舉型別的有效列舉值如下頁所示
 - ◆ Value : long 資料型別的 FILTER 值，對應於 11 or 29 bits 的 Message ID 值
 - ◆ Type : 定義於列舉型別 CAN_CONFIG 中的列舉值
 - ✦ 定義要設定為 Standard 或 Extended ID 的 Filter
 - ✦ CAN_CONFIG_STD_MSG or CAN_CONFIG_XTD_MSG

AN738 CAN Subroutine – CANSetFilter()

- CAN_FILTER 列舉型別的有效列舉值如下
 - ◆ 只能選擇其中一種做為 CANSetFilter () 的第一個參數

Value	Meaning
CAN_FILTER_B1_F1	Specifies Receive Buffer 1, Filter 1 value
CAN_FILTER_B1_F2	Specifies Receive Buffer 1, Filter 2 value
CAN_FILTER_B2_F1	Specifies Receive Buffer 2, Filter 1 value
CAN_FILTER_B2_F2	Specifies Receive Buffer 2, Filter 2 value
CAN_FILTER_B2_F3	Specifies Receive Buffer 2, Filter 3 value
CAN_FILTER_B2_F4	Specifies Receive Buffer 2, Filter 4 value

AN738 CAN Subroutine – CANIsRxReady()

- BOOL CANIsRxReady () ;
- 用來判斷是否有任何有效的資料封包被接收
 - ◆ 傳回值：
 - ✦ TRUE : 有至少一筆資料在 Receive Buffer
 - ✦ FALSE : 沒有任何資料在 Receive Buffer
 - ◆ 通常使用 CANIsRxReady() 判斷資料封包的有效性後再做正式的接收

```
If ( CANIsRxReady( ) )  
    {  
        CANReceiveMessage ( ..... ) ;  
    }  
Else  
    .....
```

AN738 CAN Subroutine – CANReceiveMessage()

- CANReceiveMessage (unsigned long *id ,
 BYTE *Data ,
 BYTE *DataLen ,
 enum CAN_RX_MSG_FLAGS *MsgFlags
) ;
- 將指定的參數位址傳入 CANReceiveMessage 以便由
CAN Module 接收資料
 - ◆ *id : 欲存放 Message ID 的變數之位址
 - ◆ *Data : 欲存放資料的 Buffer 之開頭位址
 - ◆ *DataLen : 欲存放資料長度的變數之位址
 - ◆ *MsgFlags : 存放接收狀態的列舉變數之位址

AN738 CAN Subroutine – CANReceiveMessage()

- 接收狀態的列舉變數所含訊息的列表
- 判斷方式
 - ◆ If (MsgFlags & CAN_RX_RTR_FRAME)

Buffer Value	Meaning
CAN_RX_FILTER_1	Specifies Receive Buffer Filter 1 caused this message to be accepted
CAN_RX_FILTER_2	Specifies Receive Buffer Filter 2 caused this message to be accepted
CAN_RX_FILTER_3	Specifies Receive Buffer Filter 3 caused this message to be accepted
CAN_RX_FILTER_4	Specifies Receive Buffer Filter 4 caused this message to be accepted
CAN_RX_FILTER_5	Specifies Receive Buffer Filter 5 caused this message to be accepted
CAN_RX_FILTER_6	Specifies Receive Buffer Filter 6 caused this message to be accepted

Condition Value	Meaning
CAN_RX_OVERFLOW	Specifies Receive Buffer overflow condition
CAN_RX_INVALID_MSG	Specifies invalid message
CAN_RX_XTD_FRAME	Specifies Extended Identifier message
CAN_RX_RTR_FRAME	Specifies RTR message
CAN_RX_DBL_BUFFERED	Specifies that this message was double buffered

Ex3 : GET Data From Bus

- 將 EX2.C 中 RX_Filter0 中的內容改為 0x88
 - ◆ 將修改過的 EX2.C 存成 EX3.C
- 因為主控台的 Message ID 為 0x88，故各站皆可收到主控台的資料而在 LCD Display 上有所顯示
- 若 EX3.C 的動作正確，可將 RX_Filter0 改成鄰站的 Message ID，然後嘗試是否可相互控制

Ex4 : 正確的 On Bus 程序

- 再送出 Message 之前, 最好先進入 LISTEN ONLY MODE
 - ◆ 確保不會因為錯誤的設定而影響 CAN network
 - ◆ 在收到至少一筆正確資料後再將 MCP2510 設定成 NORMAL MODE 並開始設定送出資料
- 將 EX3.C 修改並存成 EX4.C , 於 CANSetOperationMode() 中將 PIC18F458 設定為 LISTEN ONLY mode
 - ◆ 在CANInitialize() 中指定參數 **CAN_CONFIG_ALL_MSG** 給 CAN_CONFIG_FLAGS , 以便關掉 Filter & Mask 的檢查
 - ◆ 寫一個檢查 CANIsRxReady() 的程式迴圈, 一直到接收到 Message ID = 0x088 的資料才跳出
 - ◆ 跳出檢查迴圈後進入 CONFIGURATION mode
 - ✦ 重新執行 CANInitialize() 並指定參數 CAN_CONFIG_VALID_STD_MSG 給 CAN_CONFIG_FLAGS , 以便重設 Filter & Mask 的檢查為只收 Valid Standard Message
 - ◆ 重設 PIC18F458 為 NORMAL mode , 讓資料可正常地收送 !!

EX5 : 測試

REMOTE Transfer Request

- REMOTE Transfer Request 的正確程序為：
 - ◆ 將 TXBnDLC 的 bit 6 (RTR) 位元設定為 “1”
 - ◆ TXBnDLC 的 DLC<3:0> 設定為 “0”
 - ◆ Message ID (TXBnSIDH , TXBnSIDL , TXBnEIDL & TXBnEIDH) 寫入欲讀入的 Message ID
 - ◆ 傳送 RTR Frame 後將立刻接收到被要求的 Message ID 送來的資料
- 修改 EX4.C , 當 TMR1_OVERFLOW = 1 時先不送出資料而是對指定的 Message ID 送出 RTR Frame
- 使用 AN738 的 CANSendMessage() 來送出 RTR Frame
 - ◆ 指定 CAN_TX_RTR_FRAME 於 CAN_TX_MSG_FLAGS 的參數列中
 - ◆ 第三個參數 DataLen 設定為 “0” 即可

EX5 : 測試 REMOTE Transfer Request

- 修改後的程式為 EX5.C
 - ◆ RTR Frame 中的 ID 將分別為 0x81 .. 0x86，各個 CAN Node 依照指定的資訊做設定
 - ◆ 主站於收到 RTR Frame 後將 A/D 的資料以該 RTR Frame 所指定的 Message ID 傳出去
 - ◆ 各個 Node 若是動作正確，將在 LCD 上顯示主站的 A/D 值

EX6 : 深入探討 REMOTE Transfer Request

- 在完成了用 RTR Frame 來要求資料後, 試著完成能接收 RTR Frame 並正確反應的程式 EX6.C
 - ◆ 使用 RXB1 來接收外來的 RTR Frame
 - ✦ RX_Filter2 to RX_Filter5 中有一個要設定與自己的 MESSAGE_ID1 的值相同! Why ??
 - ✦ 注意 RXB1 的 MASK 值的設定是否正確!
 - ◆ 若 RXB1 有資料進來並符合以下條件則將自己的 A/D 值送出
 - ✦ RX_ID2 = MESSAGE_ID1
 - ✦ 傳回的列舉變數中 CAN_RX_RTR_FRAME = True
 - 若指定的列舉變數名稱爲 CANMessageFlag
 - 則判定方式爲:
If (CANMessageFlag & CAN_RX_RTR_FRAME)

EX6 : 深入探討 REMOTE Transfer Request

- 主站會選擇由 0x100 .. 0x150 等不同的 Message ID 來發送 RTR Frame，也就是說當主站未要求時，在 CANKing 的顯示幕上不該看到未被 RTR Frame 要求的 Node 送出資料 !!

補充教材

Microchip PIC18FXXX
with
ECAN



ECAN

What is the ECAN module and
what makes it special?

PLCmicro®'s ECAN Module

- The ECAN module is compatible with the Bosch specifications for CAN 2.0B
- Three Operational Modes in One CAN engine
 - ◆ “Mode 0” Legacy Mode
 - ◆ “Mode 1” Enhanced Mode
 - ◆ “Mode 2” FIFO Mode
- Modes 1 and 2 support DeviceNet Protocol





ECAN

64/80 Pin ECAN Solution

Device Definition for 80-pins

- **PIC18F8680**

- ◆ 64K bytes Enh. Flash
- ◆ 3Kbytes Data RAM
- ◆ 1024 bytes Data E²

- **PIC18F8585**

- ◆ 48Kbytes Enh. Flash
- ◆ 3K bytes Data RAM
- ◆ 1024 bytes Data E²

- **Packages**

- ◆ 80 TQFP (68 I/O)

- **Core**

- ◆ 16-bit instruction
- ◆ 8x8 hardware Multiply
- ◆ Multi-level Interrupts
- ◆ 10MIP at 10MHz

- **Peripherals**

- ◆ Analog
 - ✦ 10-Bit ADC
 - ✦ 2xComparators
- ◆ Communication
 - ✦ **AUSART (LIN)**
 - ✦ SPI / I²C
 - ✦ Parallel Slave Port
 - ✦ **ECAN**
- ◆ Timers
 - ✦ 3x16, 1x8 Bits
 - ✦ 1xCCP & 1 ECCP

Device Definition for 64-pins

- **PIC18F6680**

- ◆ 64K bytes Enh. Flash
- ◆ 3Kbytes Data RAM
- ◆ 1024 bytes Data E²

- **PIC18F6585**

- ◆ 48Kbytes Enh. Flash
- ◆ 3K bytes Data RAM
- ◆ 1024 bytes Data E²

- **Packages**

- ◆ 64L TQFP (52 I/O)
- ◆ 68L PLCC (52 I/O)

- **Core**

- ◆ 16-bit instruction
- ◆ 8x8 hardware Multiply
- ◆ Multi-level Interrupts
- ◆ 10MIP at 10MHz

- **Peripherals**

- ◆ Analog
 - ✦ 10-Bit ADC
 - ✦ 2xComparators
- ◆ Communication
 - ✦ **AUSART (LIN)**
 - ✦ SPI / I²C
 - ✦ Parallel Slave Port
 - ✦ **ECAN**
- ◆ Timers
 - ✦ 3x16, 1x8 Bits
 - ✦ 1xCCP & 1 xECCP

ECAN Module “Legacy Mode”

- Mode 0: "Legacy Mode"
 - ◆ Legacy Mode uses the same CAN interface as the MCP2510, PIC18C658 & PIC18F458
 - ◆ It provide for seamless migration from existing PICmicro[®] CAN devices to new PICmicro[®]'s with ECAN

Mode 0 Resources

3x TX Buffers (Dedicated)
2x RX Buffers (Dedicated)
1x Message Assembler Buffer
6x Full Acceptance Filters
2x Full Acceptance Masks



ECAN Module “Enhanced Mode”

- Mode 1: "Enhanced Mode"
 - ◆ The Enhanced Operating Mode provides additional resources and expanded functionality over and beyond what is offered in Mode 0 "Legacy Mode"

Mode 1 Functionality

DeviceNet protocol support in hardware

Mode 1 Resources:

- 3x TX Buffers (Dedicated)
- 6x TX/RX Buffers (Independently Configured)
- 2x RX Buffers (Dedicated)
- 1x Message Assembler Buffer
- 16x Full Acceptance Filters
- 2x Full Acceptance Masks



ECAN Module “FIFO Mode”

- Mode 2: “FIFO Mode”
 - ◆ RX buffer are NOT associated with a given Acceptance Filter. Messages are stored in the FIFO buffers based on the sequence in which they are received.

Mode 2 Functionality

DeviceNet protocol support in hardware

Mode 2 Resources:

3x TX Buffers (Dedicated)

8x FIFO Buffer

(6 buffers are Independently Configurable)

1x Message Assembler Buffer

16x Full Acceptance Filters

2x Full Acceptance Masks



ECAN



28/40 Pin ECAN Solution

With EIS

&

nanoWatt Technology

Enhanced Flash Devices in 40/44-pins

● PIC18F4680

- ◆ 64K bytes Enh. Flash
- ◆ 4Kbytes Data RAM
- ◆ 1024 bytes Data E²

● PIC18F4585

- ◆ 48Kbytes Enh. Flash
- ◆ 4K bytes Data RAM
- ◆ 1024 bytes Data E²

● Packages

- ◆ 40 DIP (36 I/O)
- ◆ 44 TQFP/QFN (36 I/O)

● Core

- ◆ Extended Instruction Set
- ◆ 8x8 hardware Multiply
- ◆ 10 MIPS at 10MHz
- ◆ nanoWatt

● Peripherals

- ◆ Analog
 - ◆ 10-Bit ADC (11 ch)
 - ◆ 2xComparators
- ◆ Communication
 - ◆ EUSART (LIN)
 - ◆ MSSP (SPI /MI²C)
 - ◆ ECAN
- ◆ Timers
 - ◆ 3x16, 2x8 Bits
 - ◆ 1xCCP & 1xEECP

Enhanced Flash Devices in 28-pins

- **PIC18F2680**
 - ◆ 64K bytes Enh. Flash
 - ◆ 4Kbytes Data RAM
 - ◆ 1024 bytes Data E²
- **PIC18F2585**
 - ◆ 48Kbytes Enh. Flash
 - ◆ 4K bytes Data RAM
 - ◆ 1024 bytes Data E²
- **Packages**
 - ◆ 28L SDIP/SOIC (25 I/O)
- **Core**
 - ◆ **Extended Instruction Set**
 - ◆ 8x8 hardware Multiply
 - ◆ 10 MIPS at 10MHz
 - ◆ **nanoWatt**
- **Peripherals**
 - ◆ **Analog**
 - ✦ **10-Bit ADC (8 ch)**
 - ✦ **2xComparators**
 - ◆ **Communication**
 - ✦ **EUSART (LIN)**
 - ✦ **MSSP (SPI /I²C)**
 - ✦ **ECAN**
 - ◆ **Timers**
 - ✦ **3x16, 2x8 Bits**
 - ✦ **2 x CCP**

What's New?

- ECAN Module on a 28/44 Pin Device
- Extended Instruction Set on a CAN Device
- Changes to the nanoWatt module
- Development Tools Support

PIC18F4680 系列 CAN MCU 對現有 PIC18FX58 使用者的助益

- 當現有 PIC18F458/258 客戶們需要以下任何一樣增加的效能或週邊時
 - ◆ More Memory
 - ◆ Better Power Management
 - ◆ More ADC Channels
 - ◆ Faster ADC
 - ◆ Analog Comparators
 - ◆ Optimized Core for RTOS
 - ◆ LIN capable UART

PIC18F458 PC18F4680

的主要差異之表列

	PIC18F458	PIC18F4680/4585
Optimized Core for RTOS	Traditional	E.I.S
More Program Memory	32Kbytes	48K or 64Kbytes
More RAM	1.5Kbytes	3Kbytes
More EEPROM	256 bytes	1024 bytes
Boot Block	512 bytes	2K bytes
Better Power Management	Sleep/TMR1 (2)	nanoWatt (6)
More ADC Channels	8 ADC Inputs	11 ADC Inputs
Faster ADC	30Ksps	100Ksps
LIN capable USART	Software	Yes
4xPLL Configuration	Fuse	Fuse / Soft PLL
WDT Time	20ms	Up to 2 minutes
CAN Engine	CAN	ECAN

Extended Instruction Set Core

Here is what has Changed

- PIC18F Architectural Enhancements:
 - ◆ One (1) new addressing mode (Literal Indexed)
 - ◆ Eight (8) new instructions
- The “Extended Instruction Set” and “Literal Indexing” are transparent to the customers using “C”

These extensions greatly simplify the code that is needed for software stack accesses.

New Indexed Addressing

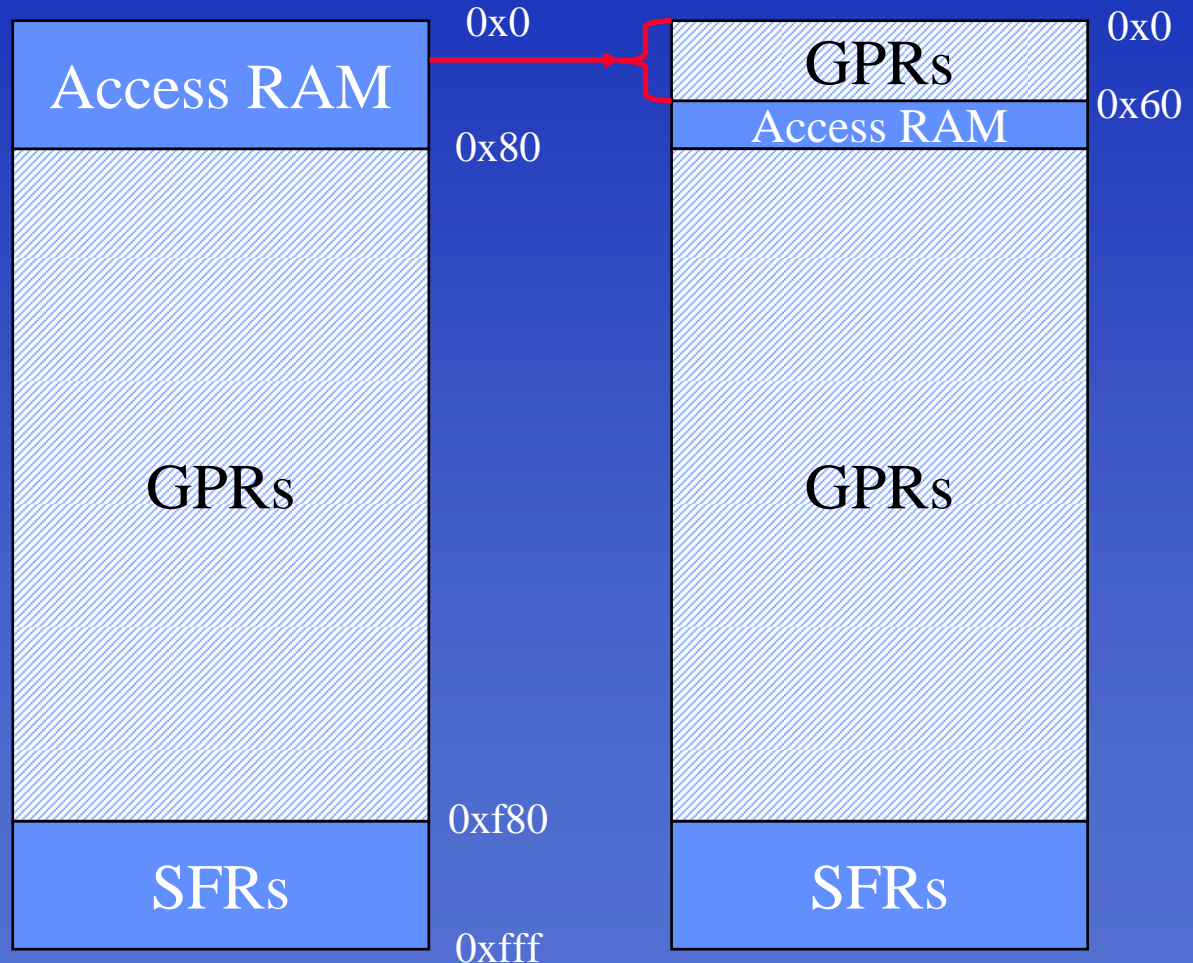
Indexed literal addressing remaps the access RAM window to be a stack offset window.



= *banked addressing*



= *unbanked addressing*



Traditional 18F

Extended 18F

Extended Instruction Set

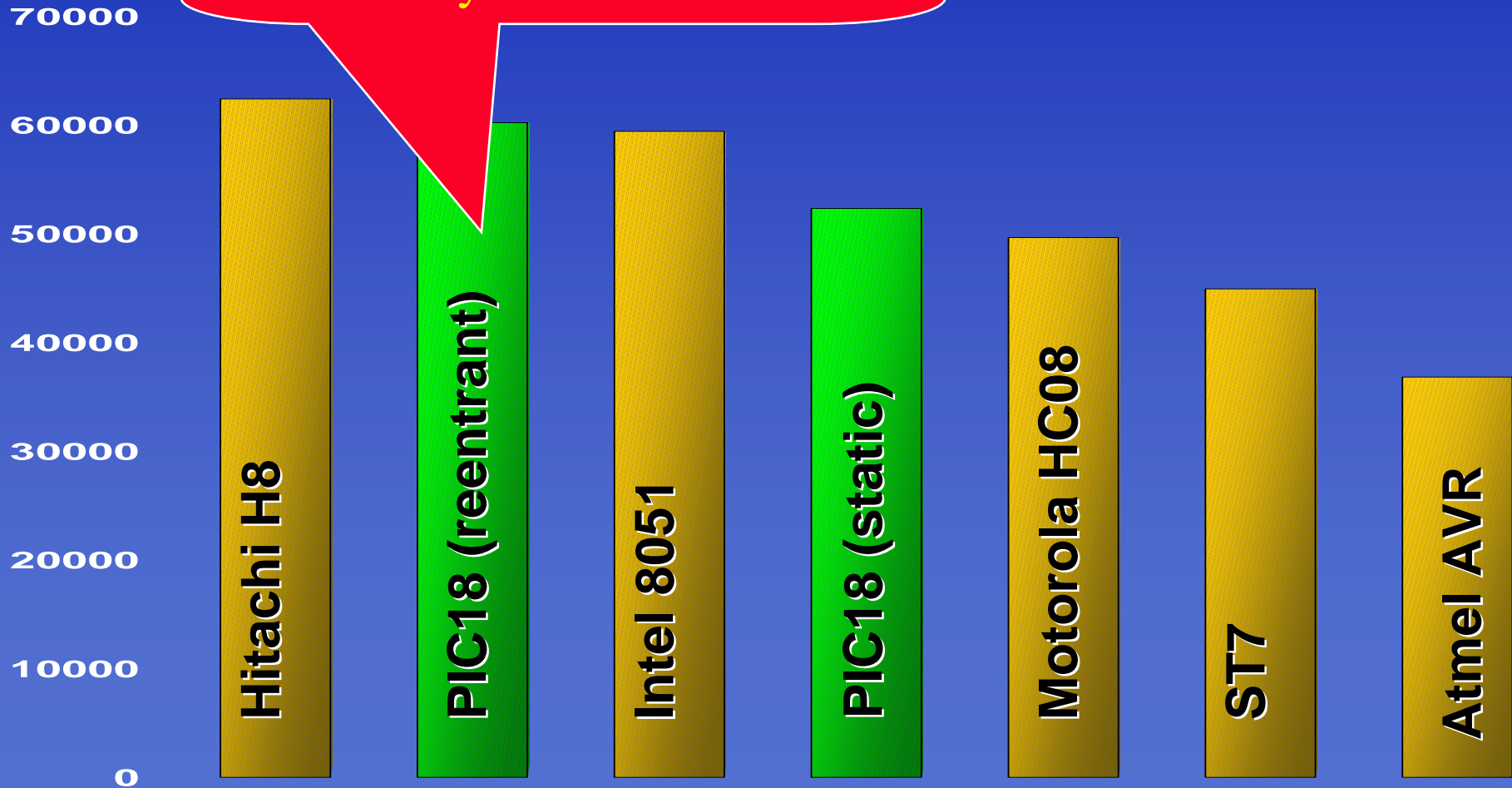
- EIS consists of 8 new instructions
 - ◆ ONLY FSR2 is effected by these new Instructions

EIS	Description	Code Size on EIS	Code size on Traditional PIC18F
ADDFSR	Add Literal to FSR	2bytes	4 to 10 byte
SUBFSR	Subtract Literal for FSR	2bytes	4 to 10 byte
ADDULNK	Add Literal To FSR2 and Return	4bytes	6 to 12 byte
SUBULNK	Subtract Literal To FSR2 and Return	4bytes	6 to 12 byte
CALLW	Call subroutine using Wreg	14bytes, 0 branches	18bytes, 2 branches
MOVSF	Move Indexed to file	4bytes	6bytes
MOVSS	Move Indexed to Indexed	4bytes	12bytes
PUSHL	Store Literal in Indexed Address	2bytes	4bytes

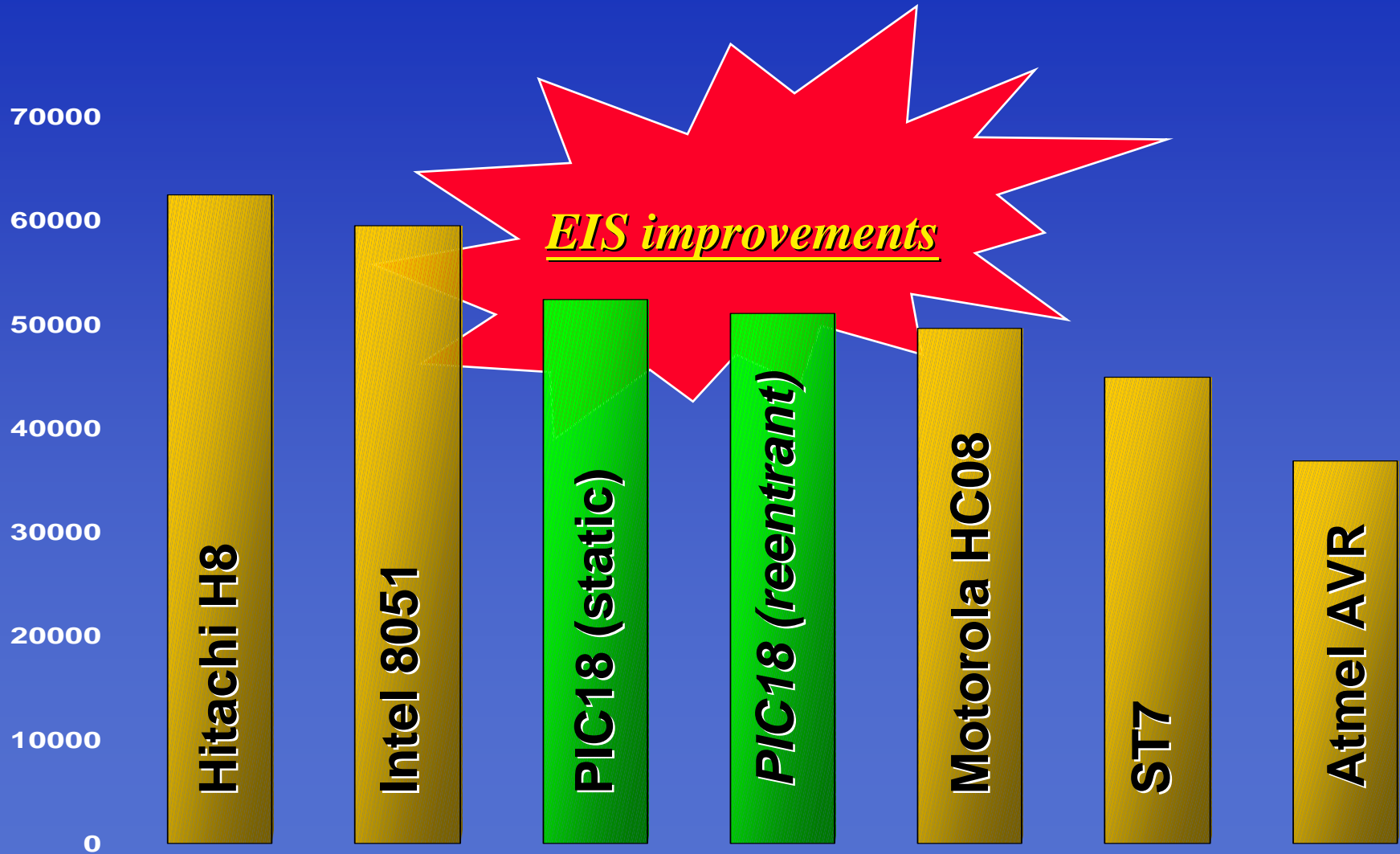
- Additional information is provided in Section 24.2 of the Data Sheet

C-Compiler Bench Marks

Improve Reentrant "C" Code
Efficiency in MPLAB® C18



Bench Marks with new EIS's



What Mode to Use

Use Traditional Mode When...

- The static model is O.K.
- Modifying an existing design
- Code is mostly written in ASM
- Code must also work on non-extended cores
- RAM space is not critical

Use Extended Mode When...

- Reentrancy is required
- Starting a new design
- Code is mostly written in C
- Portability to non-extended cores is not needed
- RAM space is critical

Enabling the extensions

A single configuration bit (XINST) will select between:

Traditional Mode

- New instruction opcodes are interpreted as NOPs
- Access RAM is available
- Default for Assembler
- Default for C-Compiler
- Default for MPLAB[®]-IDE
- Default for Silicon

Extended Mode

- New instructions are available
- Access Bank [0,0x5f] is converted to General Purpose for Literal Indexing

Microchip ECAN 的 C Subroutines

- Microchip 針對 ECAN 系列，提供 ECAN 的 C Subroutines
- AN878 包含 ECAN C Subroutine 的說明以及 C Source Code
- 若是使用 PIC18F458/258，其 CAN Module 屬於所謂 Legacy CAN. 只要用 AN738 即可