CAN202B Workshop

CAN Protocol

使用 **CAN bus** 來實現
高速而可靠的通信

WHAT IS *CAN*?

C A N

C C A N

C C A N

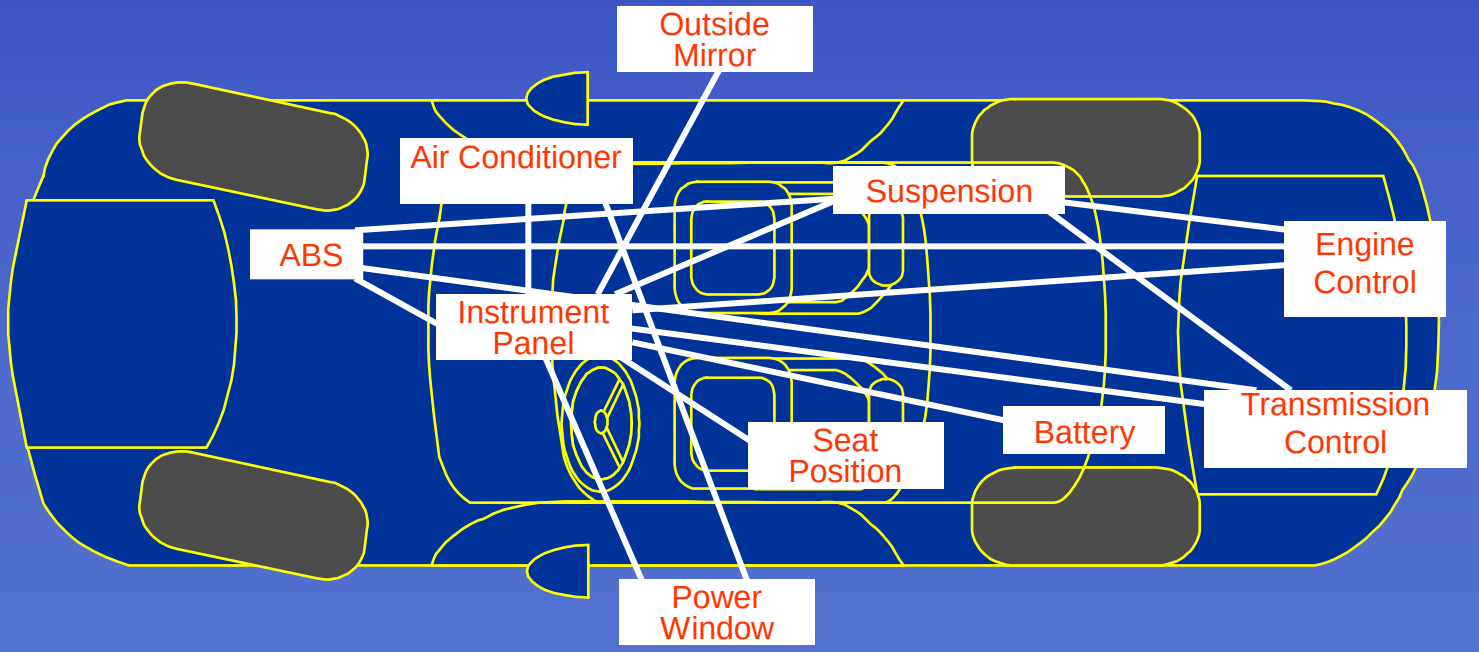
***C*ontroller *A*rea *N*etwork**

- 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 Bus !

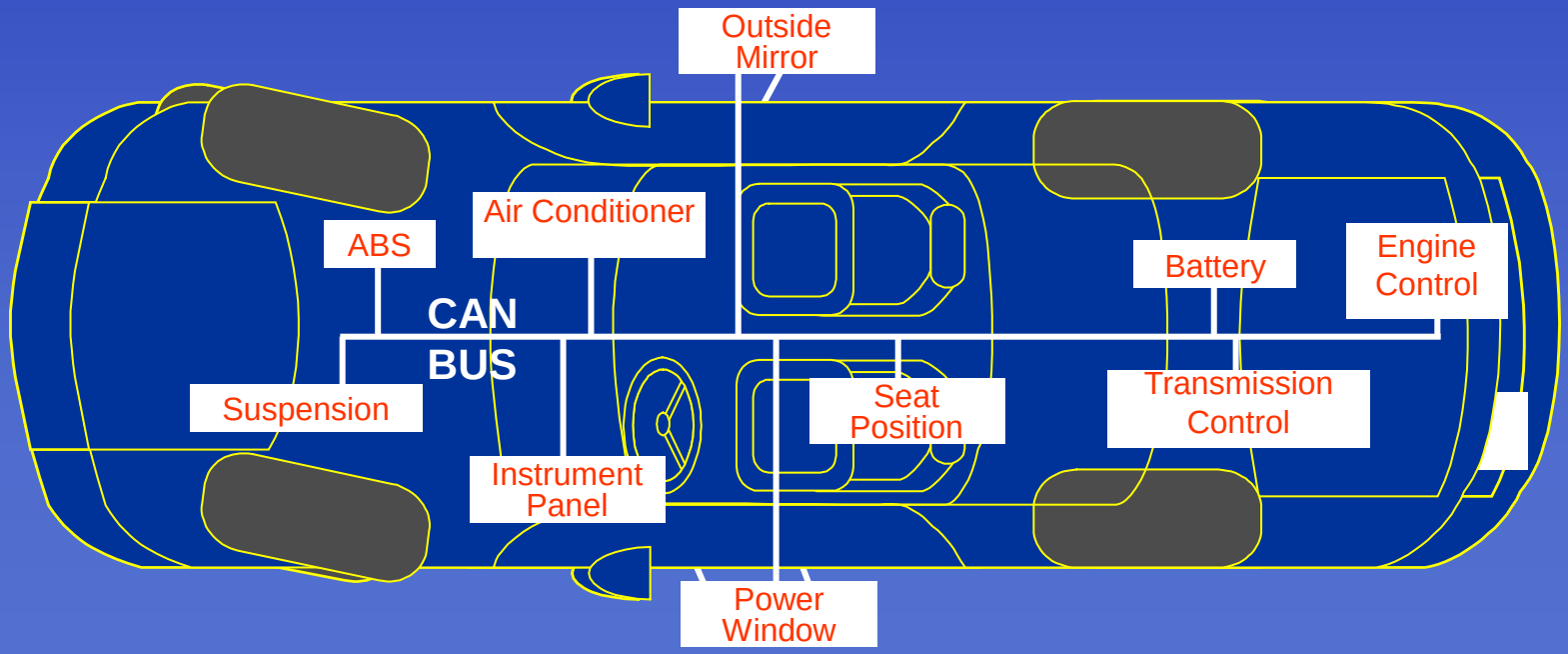
CAN (Controller Area Network) 通信協定於 80 年代由 Bosch 首先發展, 為的是因應使用於新型汽車上不斷增加的電子裝置. 這些裝置使汽車增加許多功能與附加價值, 也增加控制系統的複雜度

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 的優點

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



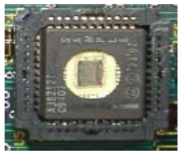
CAN 規範的發展沿革

84 **Bosch** start development on CAN

85

86 CAN **patent** filed 

87 CAN published at **SAE congress Detroit**

88 First **CAN chips** from Intel and Philips 

89

90 CAN introduced first in **weaving machines**

91 First Mercedes-Benz **S-class** with CAN 

92 Foundation of CAN in **Automation**

93 Standardization of CAN in **ISO 11898**

94

95 **CANopen** protocol published by CiA

96

97

99 

00 Introduction of **TTCAN**

01

02

03 **Specification of several ISO 11898-x:**

04 data link layer

05 high-speed physical layer

06 fault-tolerant physical layer

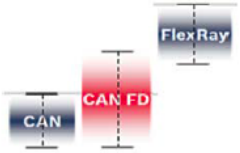
07 TTCAN

08 low-power mode

09 selective wake-up

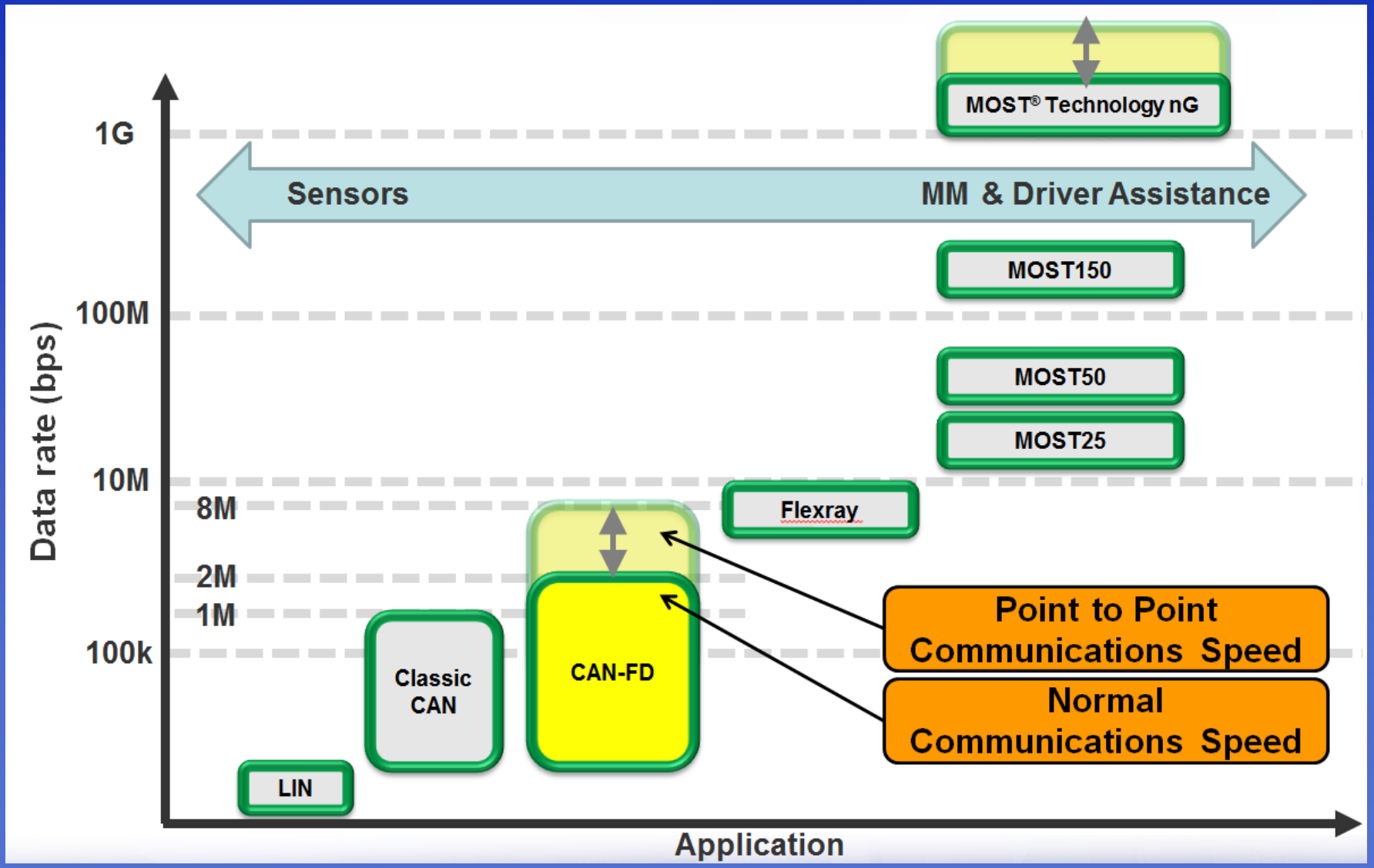
10

11 Invention of **CAN FD** (ISO 11898-7)

12 

Source: CiA

幾種常見的車用 Bus 之效能比較



CAN Network Solutions
Bosch Specifications

- ***Bosch Spec 2.0B (Active)***
  - ✓ CAN 通信協定的最新版本
  - ✓ 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

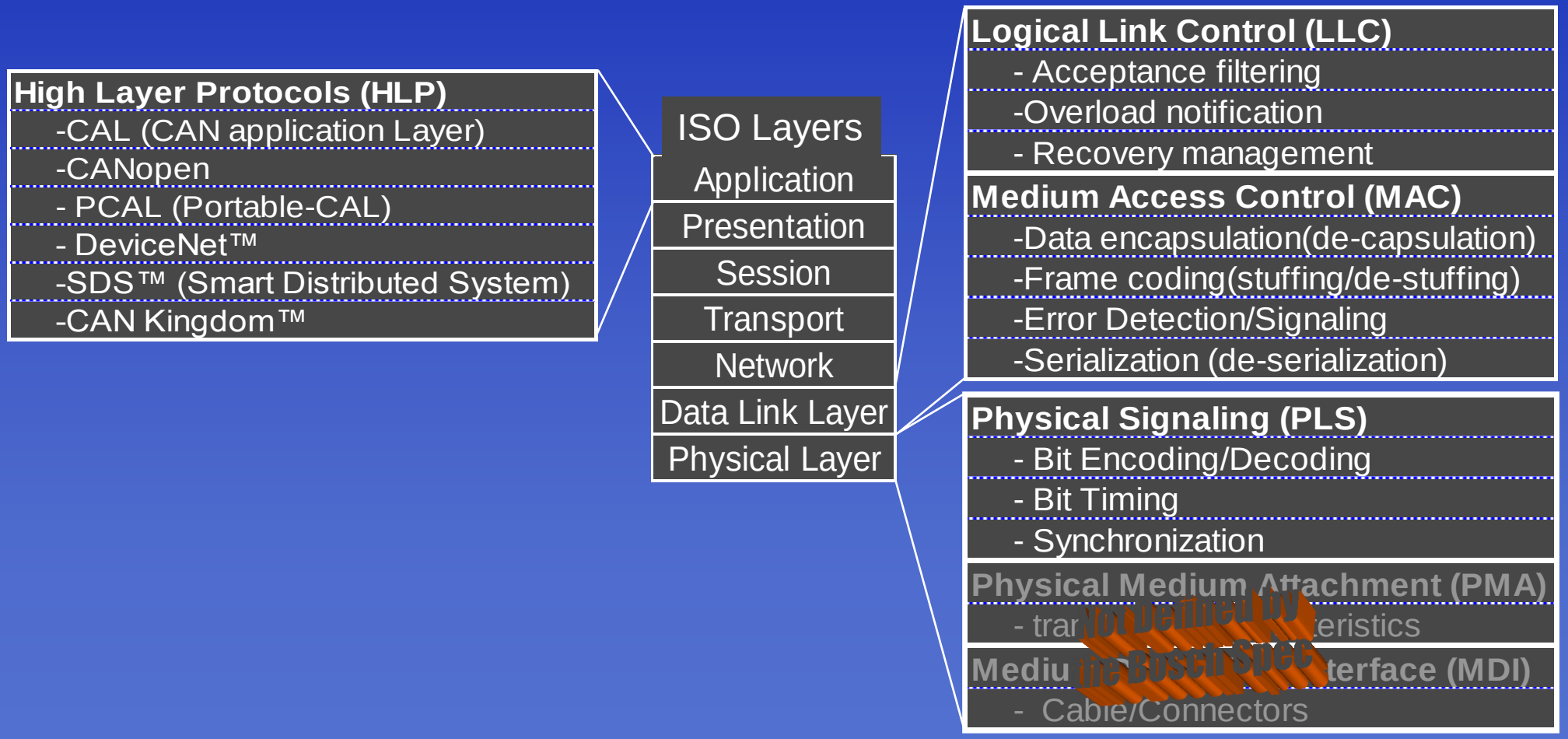
- ***Bosch Rev 2.0A***
  - Bosch 2.0A 的規格中共有 2 種不同的版本
    - ***Rev2.0A*** – 此版本的通信協定只能傳送與接收 **“standard frame message”** 的資料 **(11-bit Identifier) .** 假設接收到一個 **extended frame message (29-bit Identifier)** 則將產生 **“error flag “**
    - ***Rev2.0B Passive*** – 此版本的通信協定依然只能傳送及接收屬於 2.0A 規範的 message (Standard 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 Bus 概要總覽

- 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制定標準
  - ✓ International Standards Organization (ISO)
  - ✓ Society of Automotive Engineers (SAE)
- 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
- 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

BASIC NETWORK OVERVIEW

ISO/OSI Network Layering Reference Model for CAN Bus



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

CAN Bus 的相關規範

- ISO11898-1：定義了 CAN Bus 的 Data Link & Physical Layers
  - ✓ 但不包含 PMA & MDI
- 為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 ISO 11898-1 未定義的標準
  - ✓ ISO11898-2 是個適用於高速通信的標準規範 (up to 1Mbps)
  - ✓ ISO11898-3 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

ISO 11898-1 為 CAN Bus 所作之定義

Data Link Layer

Logical Link Control (LLC)

- Acceptance Filtering
- Overload Notification
- Recover Management

Medium Access Control (MAC)

- Data Encapsulation/Decapsulation
- Frame Coding (stuffing/destuffing)
- Error Detection/Signaling

Physical Layer

Physical Signaling (PLS)

- Bit Encoding/Decoding
- Bit Timing/Synchronization

ISO 11898-1 對實體層未定義的部分

Physical Layer

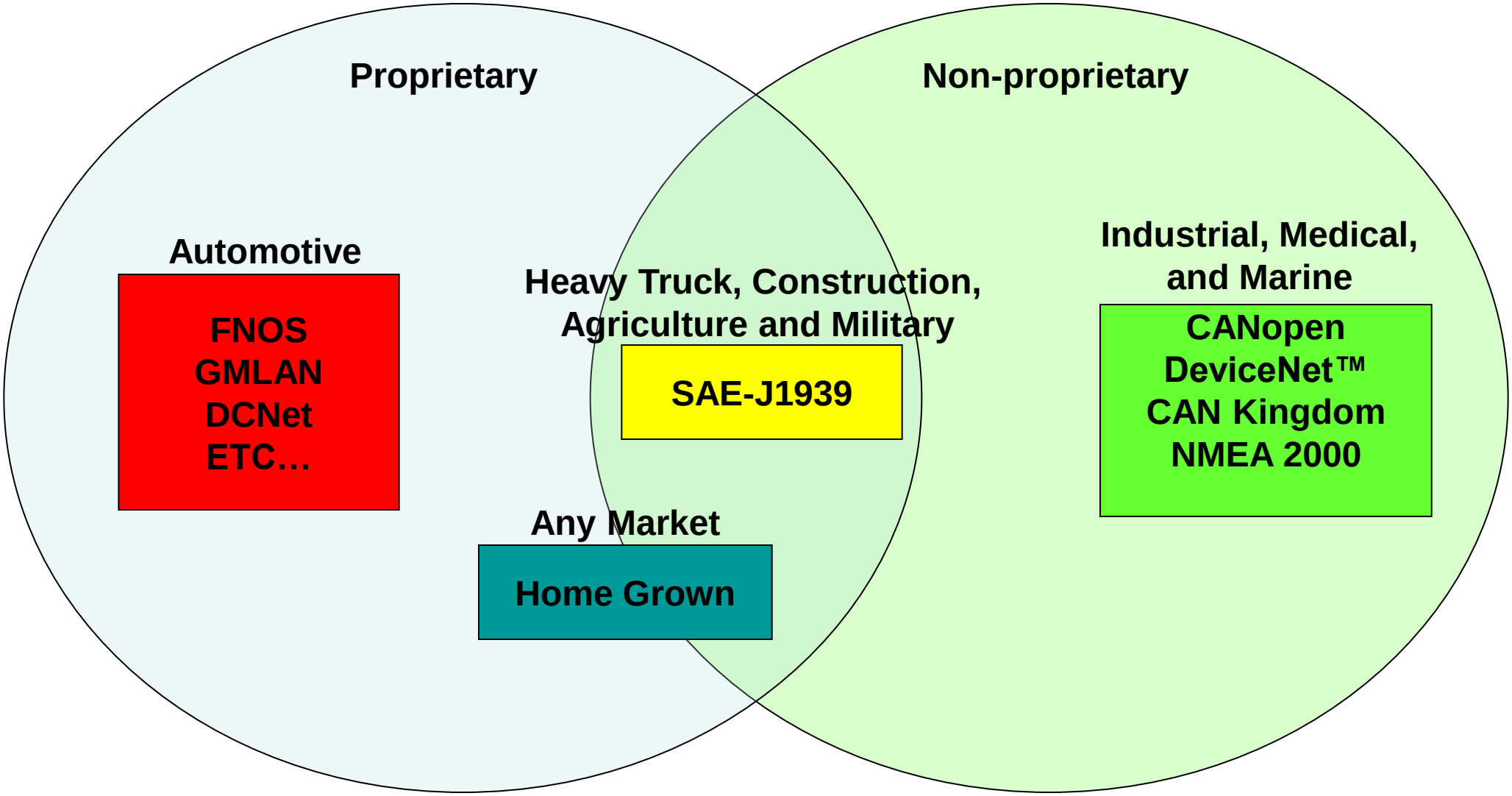
Physical Medium Attachment (PMA)

- Driver/Receiver Characteristics

Medium Dependent Interface (MDI)

- Connectors

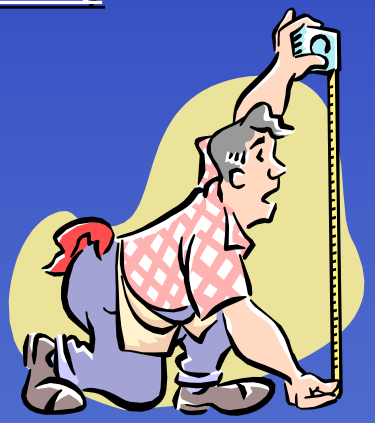
Controller Area Network HLPs



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

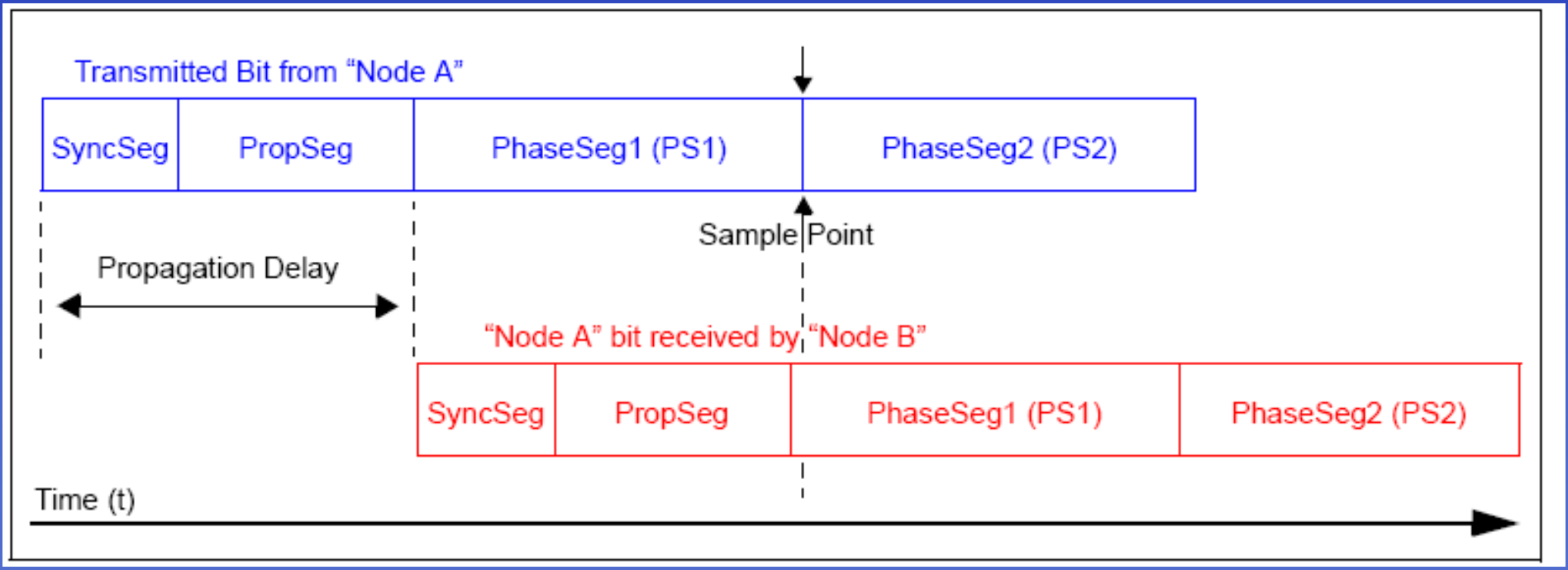
Length (meters)	(Bit per Second)
40m	1Mbps
500m	125Kbps
1,000m	50Kbps



- **CAN Network 的規模**
  - 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

Element : Propagation Delay



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 2/3 of total bit time
    - Result is a reduction in bus length with an increase in bit rate
  - Remaining 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 * 30cm/ns = 17cm/ns ($t_{prop\text{-}line}$)
    - Average propagation delay in transceiver = 25ns ($t_{prop\text{-}tcvr}$)
  - Equation - First step approximation
    - $2/3\ t_{bit} >= 4 * t_{prop\text{-}tcvr} + 2 * t_{prop\text{-}line}$
  - 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. MCP2551

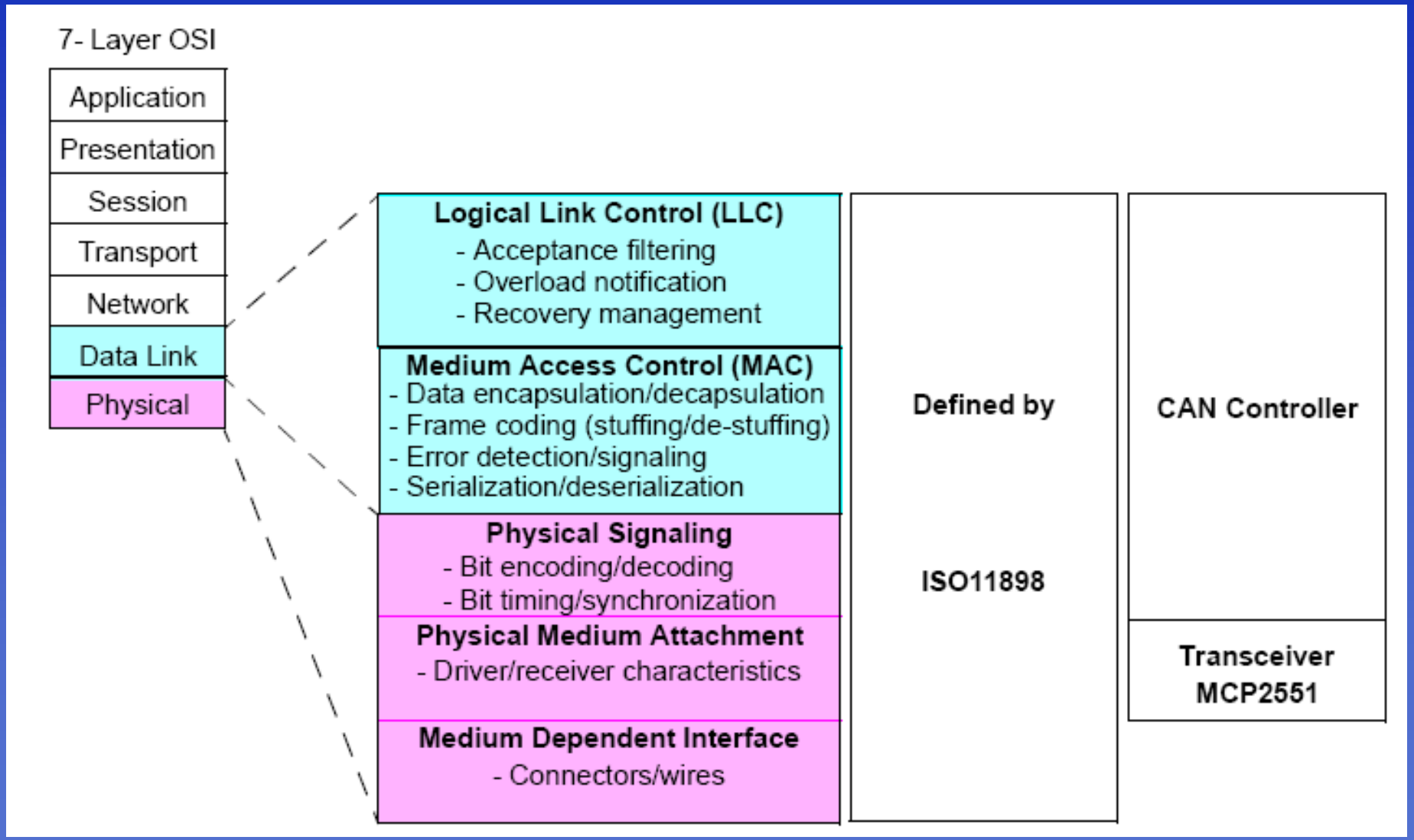
Bit Rate (Kbit/s)	Bus Length (m)
1000	30
500	100
250	250
125	500
62.5	1000

* source - CAN System Engineering - Lawrenz

CAN BUS LENGTH -v- BIT RATES

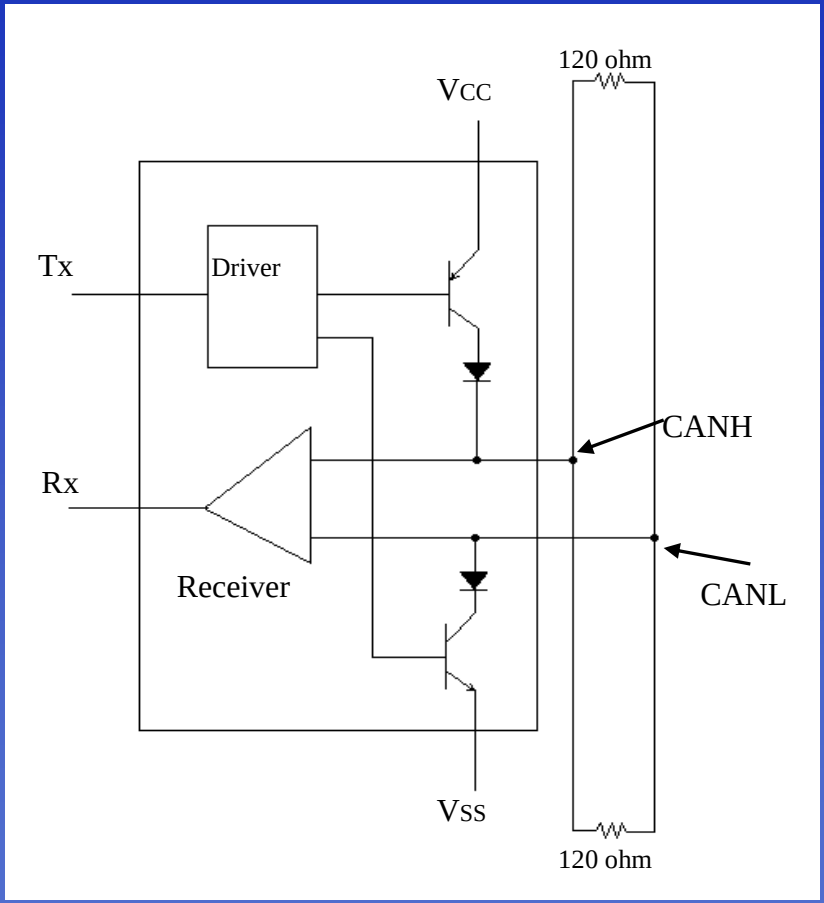
- Number of Nodes in a System
  - Factors
    - Electrical Characteristics of transceiver
    - Application specific drivers
  - Real Systems
    - Standard Differential Line Drivers - 32 Nodes
    - High Speed Diff. Line Drivers eg. MCP2561/2 =112 Nodes

What ISO 11898 Defined vs. solution

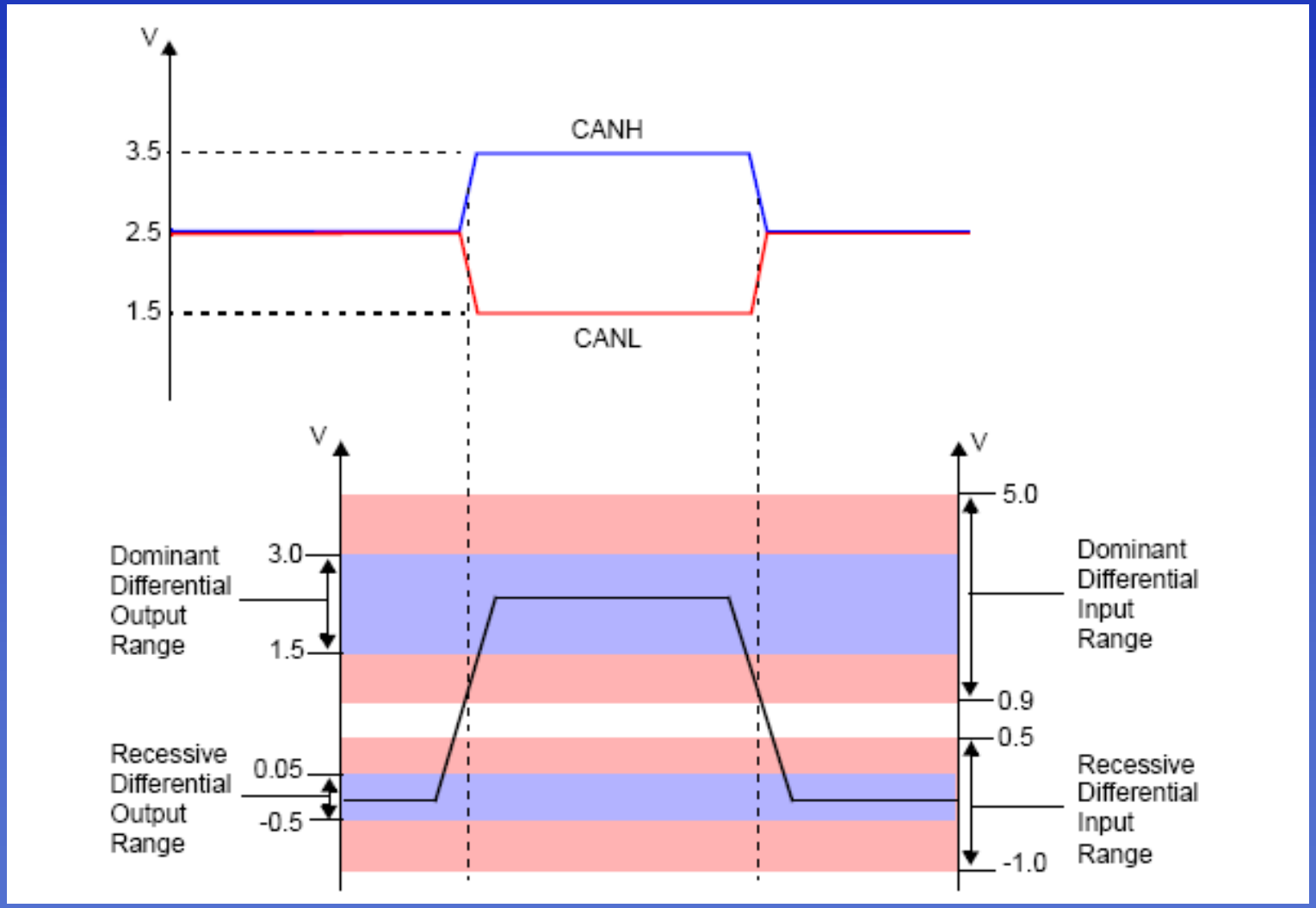


CAN 5V Differential Physical Layer

- 通常被稱為 CAN transceiver (例如: MCP2551/MCP2561/MCP2562)
- Vcc 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
  - ✓ ΔV > 1.0V 稱為 dominant
  - ✓ ΔV < 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



ISO11898 Nominal Bit Level



Comparing MCP2551 to ISO11898

Parameter	ISO-11898-4		MCP2551		Unit	Comments
	min	max	min	max		
DC Voltage on CANH and CANL	-3	+32	-40	+40	V	Exceeds ISO-11898
Transient voltage on CANH and CANL	-150	+100	-250	+250	V	Exceeds ISO-11898
Common Mode Bus Voltage	-2.0	+7.0	-12	+12	V	Exceeds ISO-11898
Recessive Output Bus Voltage	+2.0	+3.0	+2.0	+3.0	V	Meets ISO-11898
Recessive Differential Output Voltage	-500	+50	-500	+50	mV	Meets ISO-11898
Differential Internal Resistance	10	100	20	100	kΩ	Meets ISO-11898
Common Mode Input Resistance	5.0	50	5.0	50	kΩ	Meets ISO-11898
Differential Dominant Output Voltage	+1.5	+3.0	+1.5	+3.0	V	Meets ISO-11898
Dominant Output Voltage (CANH)	+2.75	+4.50	+2.75	+4.50	V	Meets ISO-11898
Dominant Output Voltage (CANL)	+0.50	+2.25	+0.50	+2.25	V	Meets ISO-11898
Permanent Dominant Detection (Driver)	Not Required		1.25	—	ms	
Power-On Reset and Brown-Out Detection	Not Required		Yes		--	

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

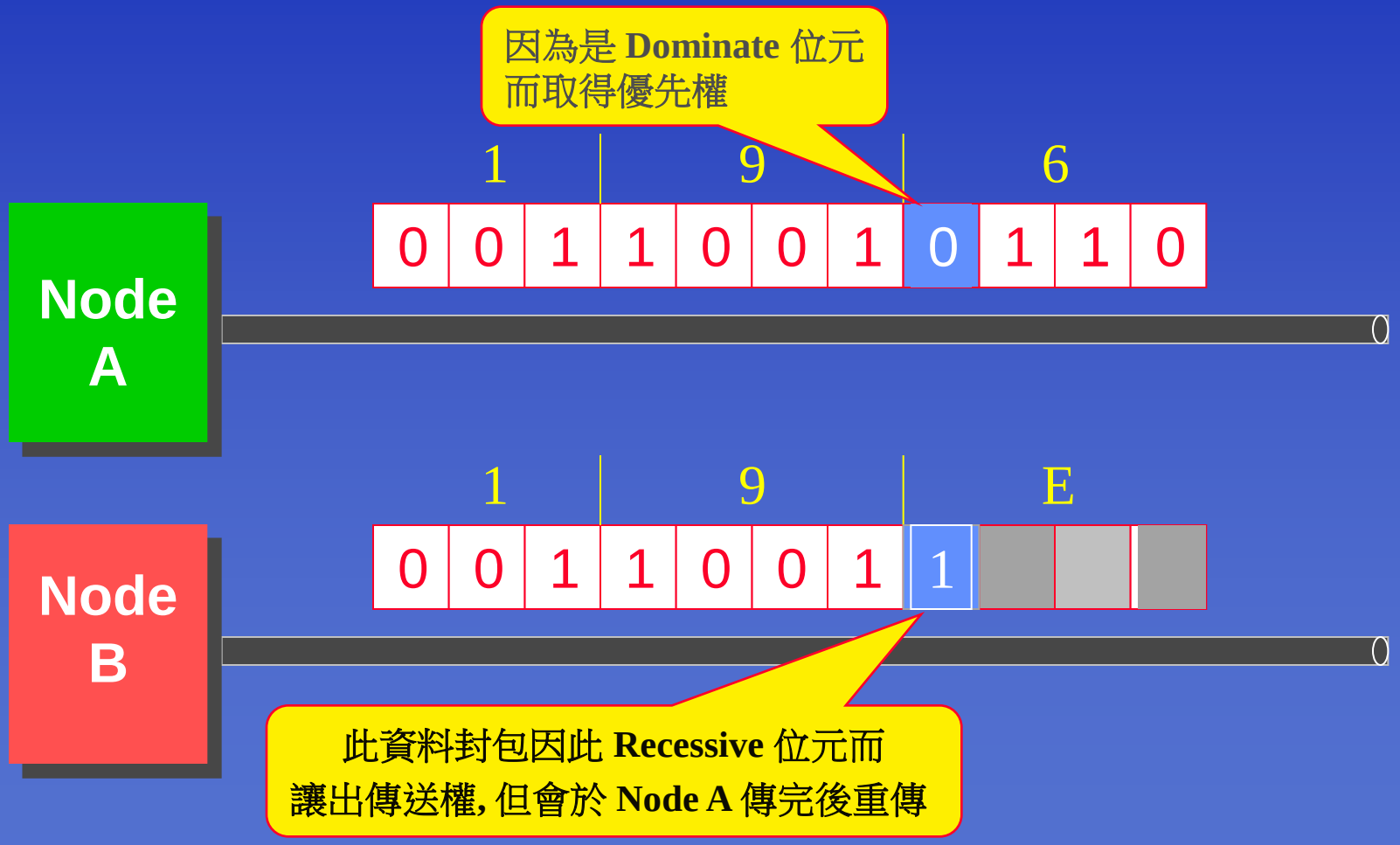
CSMA/CD-CR

CR Collision Resolution - 非破壞性的逐位元仲裁

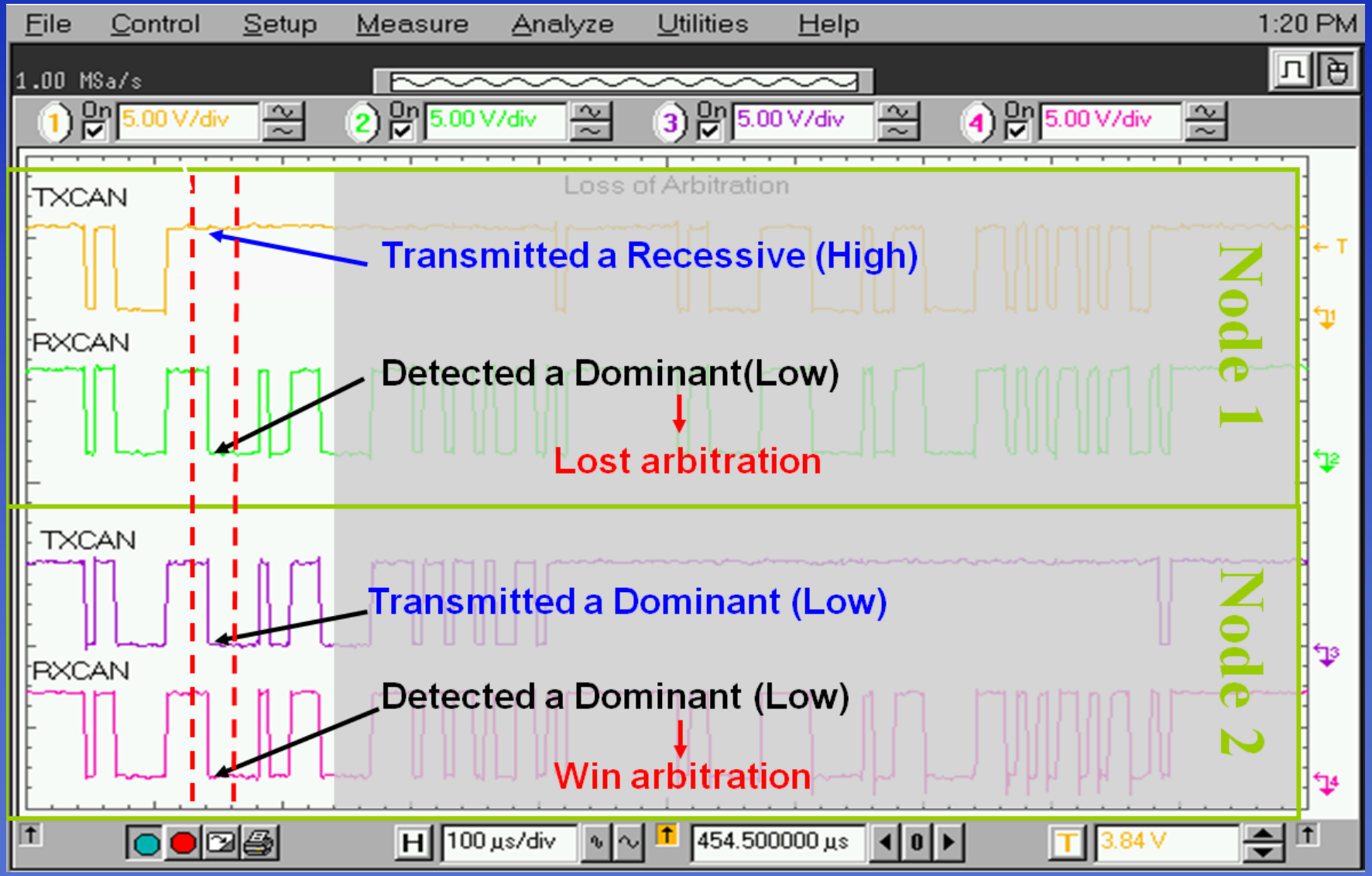
- 即使資料碰撞發生, 所要發送的 Messages 仍然保持其完整性
- 所有仲裁行為發生時都不會對具有最高優先權的資料造成任何延遲或者錯誤
- 任何因為優先權較低而無法在第一時間傳送的資料封包會自動於下一個可傳送的時間自動的被傳送
- 要達到上述功能的需求:
  - ✓ Dominant 以及 recessive 位元的狀態必需被事先定義
  - ✓ 每個 node 必須不斷的監視 bus 上的資料流, 比較自己送出的資料與真正出現在 bus 上的資料是否相符 !

CSMA/CD-CR

CR Collision Resolution - 非破壞性仲裁的例子



Arbitration(仲裁) 的信號範例



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)

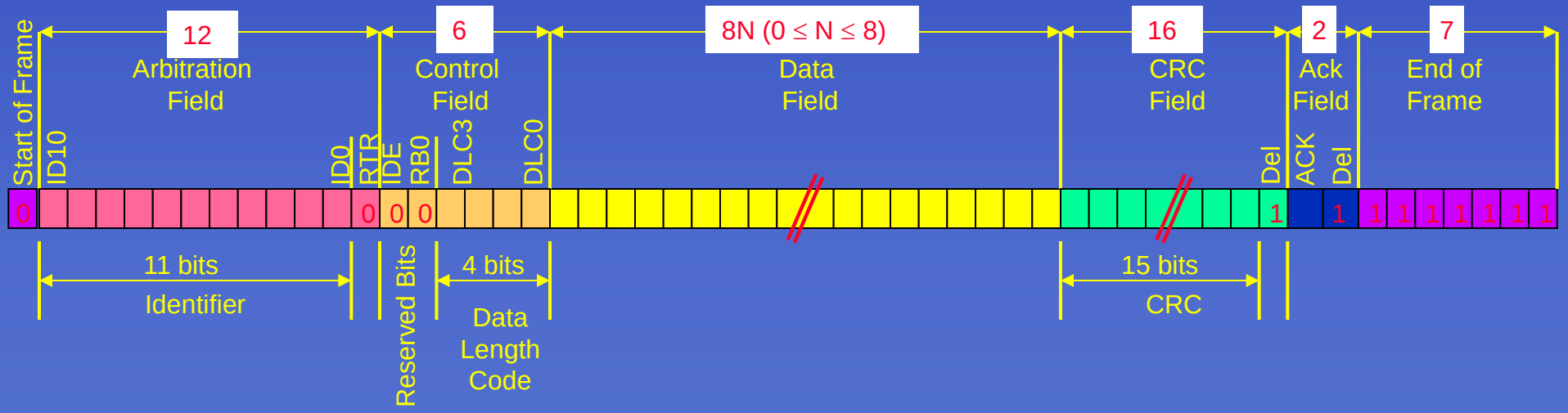
Message Frames

- **Data Frame**
  - 最為普遍的類型
  - 資料被從一個節點傳送到所有的節點
- **Remote Frame**
  - 與 Data Frame 極為相似 (只有 RTR bit 被設為 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

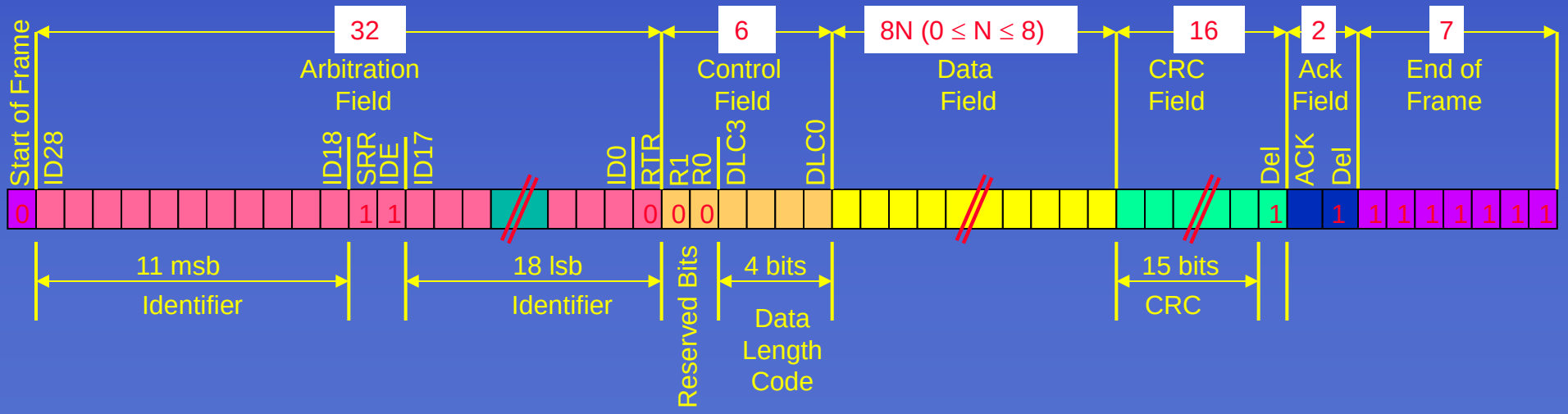
有兩種型式的 Data Frame, 依據不同版本的 CAN 協定規範而有差異, 但新版本可相容於舊的版本

- **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

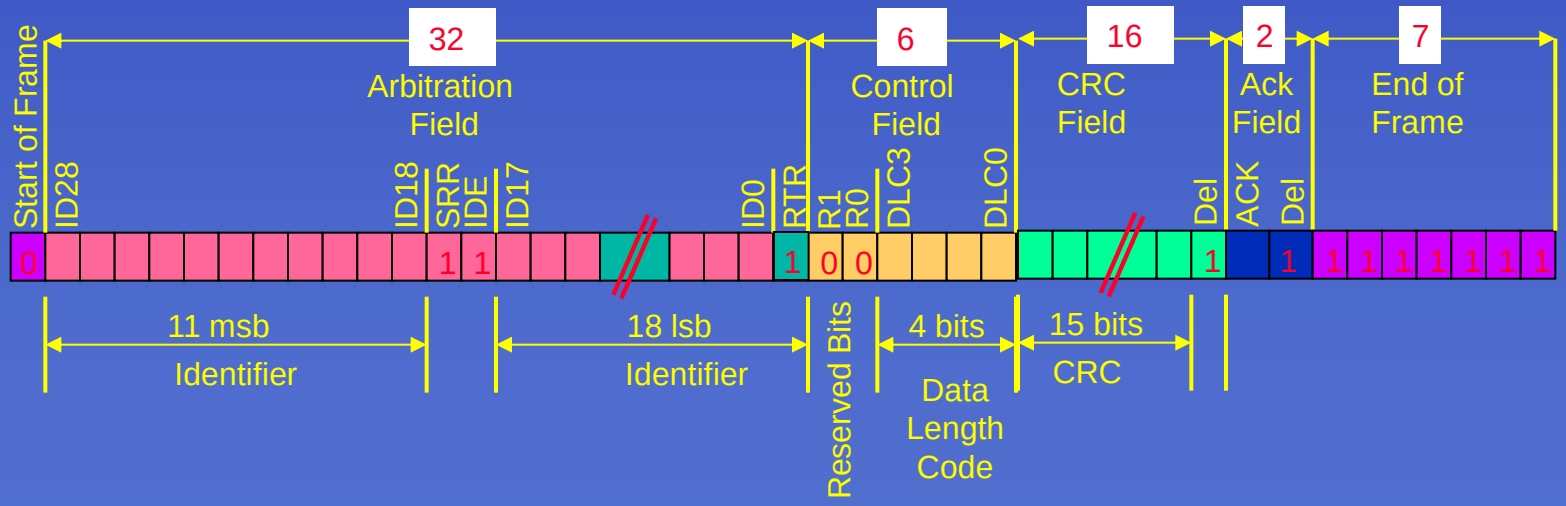
- Extended Data Frame
  - 依據最新的 CAN 協定規範所定義 (version 2.0B). 辨識欄位(identifier field) 被擴展為 29 位元; 增加了可用 Message 的數量



Remote Frame

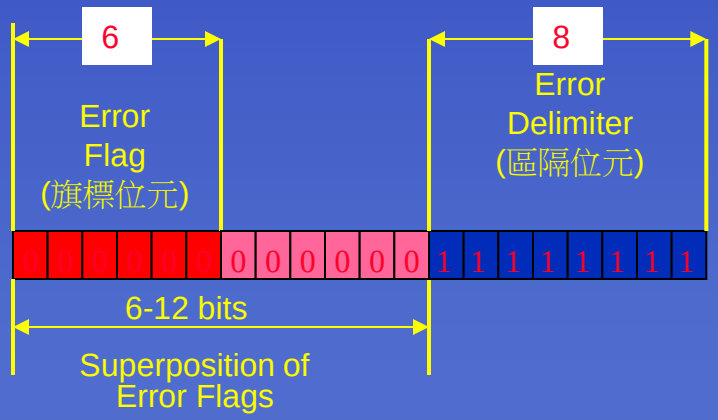
- 與 Data Frame 一樣, 有兩種型式的 Remote Frame, 依據不同版本的 CAN 協定規範而有差異, 但新版本可相容於舊的版本

- 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



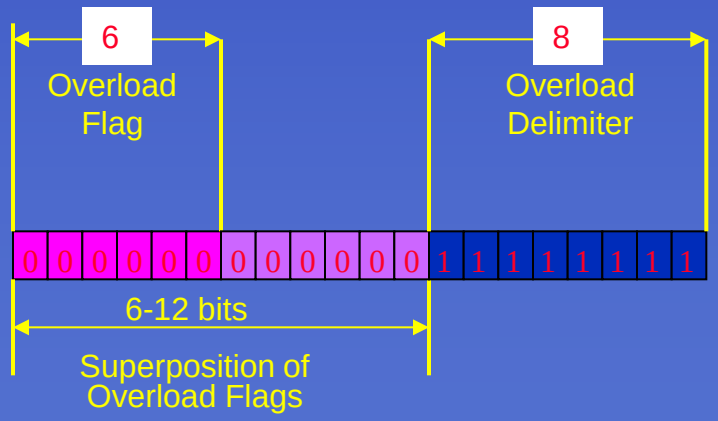
Error Frame

- 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

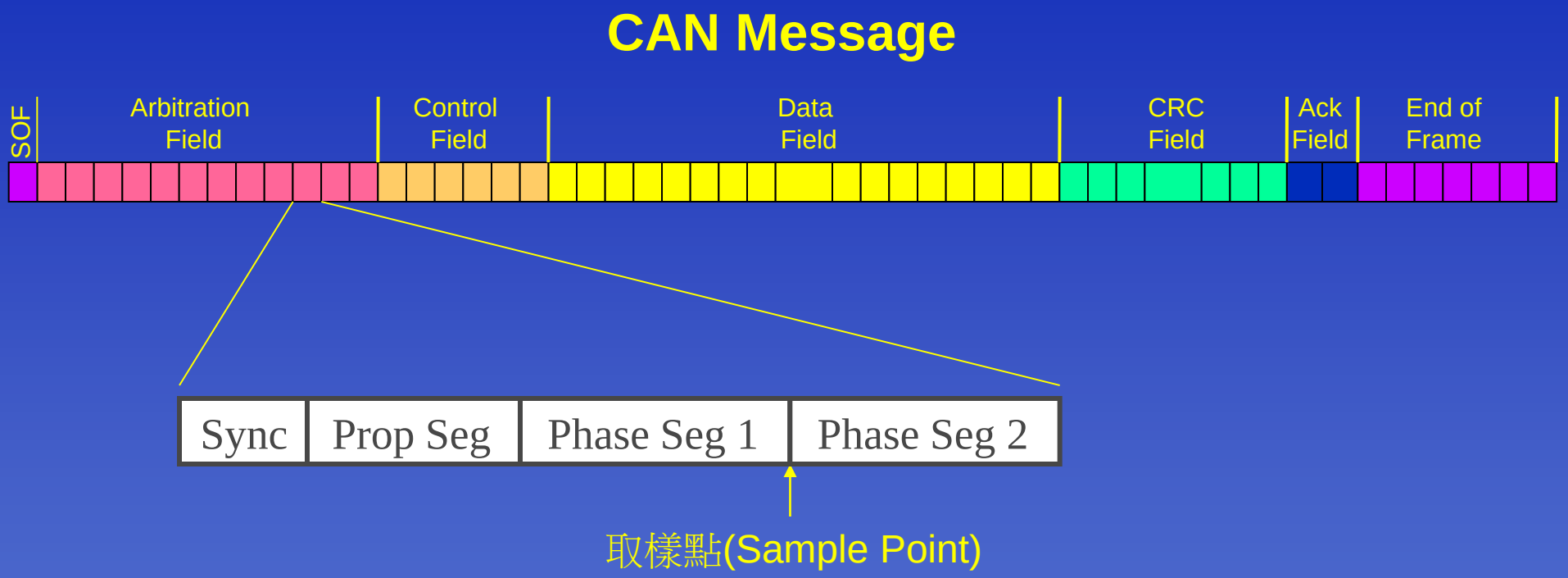


Overload Frame

- 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



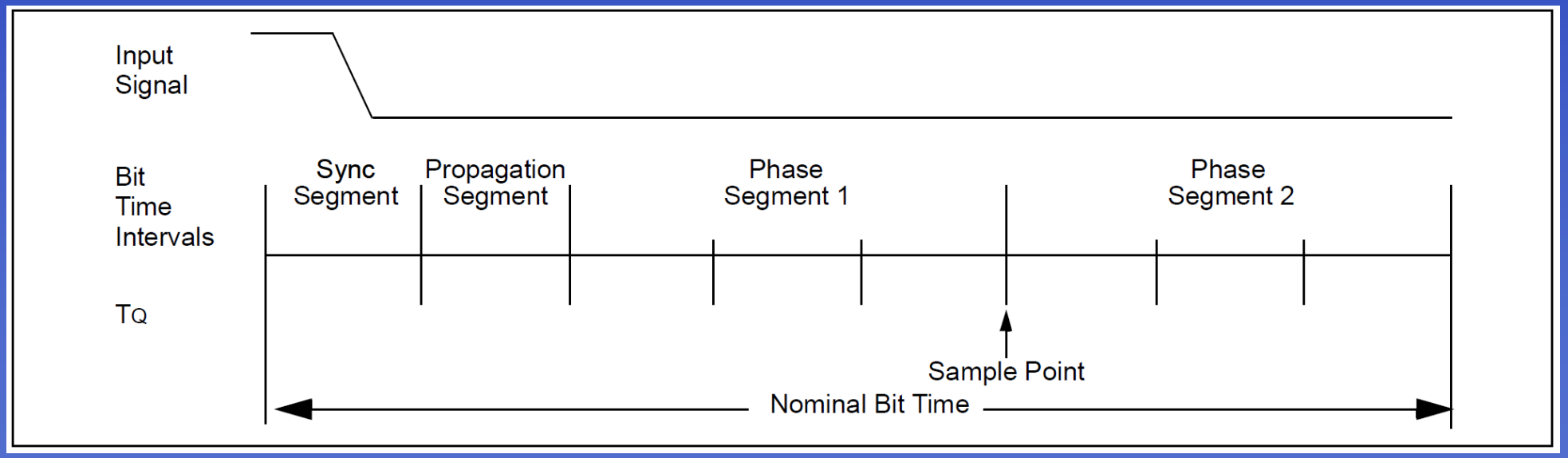
CAN Message 的位元結構



每個 CAN 的 message bit 是由 4 個 Segment 所組成

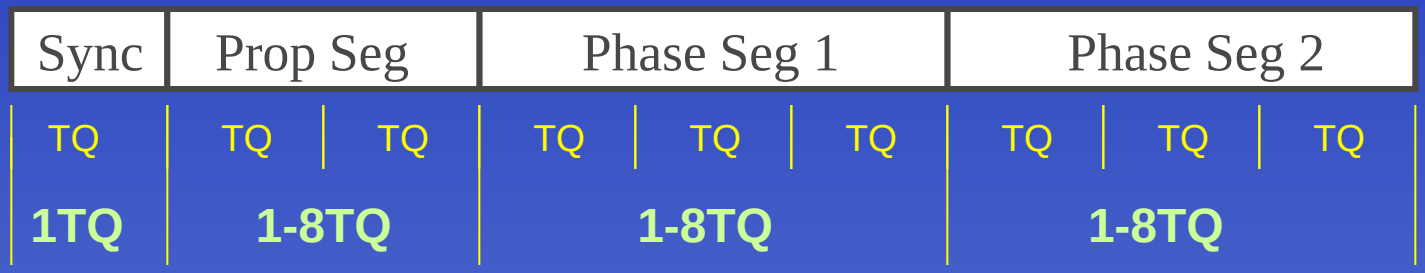
CAN Message 的位元結構

- 每一個 CAN 位元，由 4 個定義好的單元組成
- CAN 對 Bus 信號的取樣點被設定在 Phase Segment 1 與 Phase Segment 2 的交界



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)



每個 Timing Segment 可被規劃為由指定數量的 TQ 所組成

- TQ 被硬體設定為 2*(BRP + 1) (Tosc) -> (Tosc = 1/Fosc)
- 更改 Baud Rate 的預除器 (BRP) 將變更 TQ 的時間
  - Min = 1:1, Max = 1:64

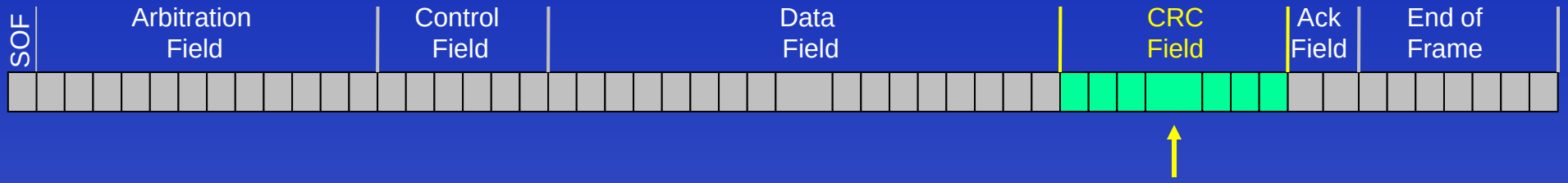
CAN 對 Error 的處理

- 為了正確的傳遞資訊, 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 做處置並制約其行為(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 因為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

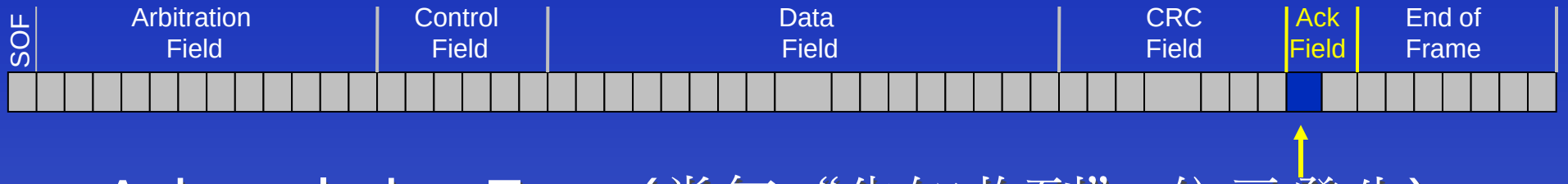
CAN Message



- 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

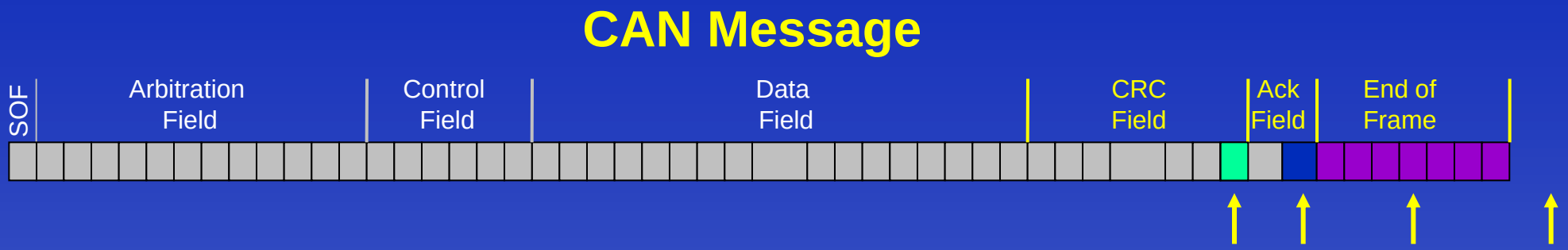
CAN 可偵測的各種錯誤

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

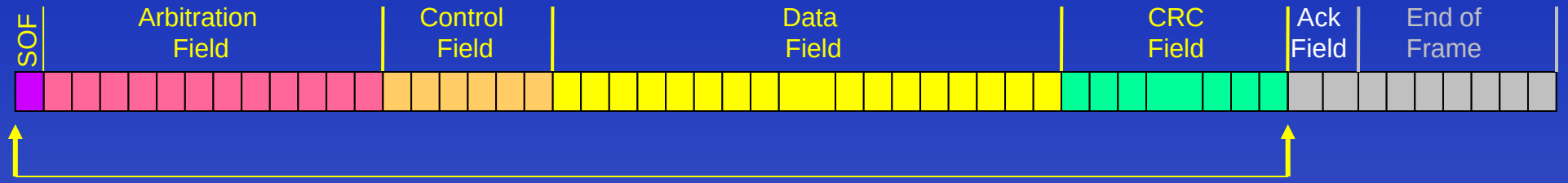
CAN 可偵測的各種錯誤



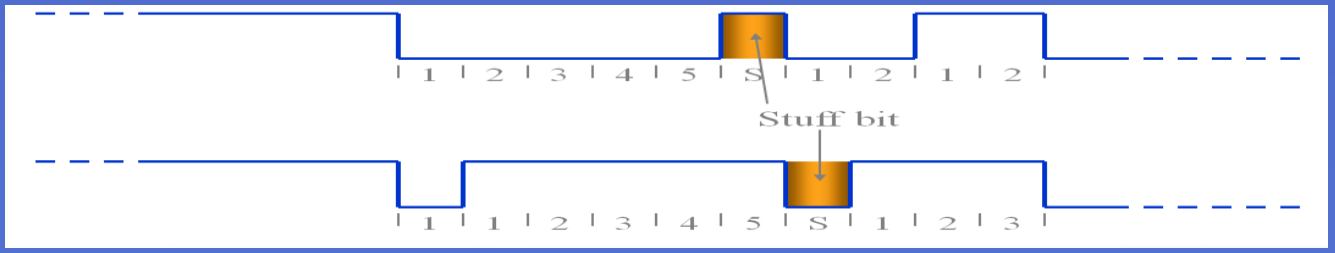
- Form Error
  - 若任一個 node 在 CRC Delimiter, Ack Delimiter, End of Frame (EOF) field 或 Interframe 的間隔偵測到有 dominate 位元則產生一個 Error Frame 來指明發生了 Form Error
  - 原先的 message 將在 Error Frame 結束後重送

CAN 可偵測的各種錯誤

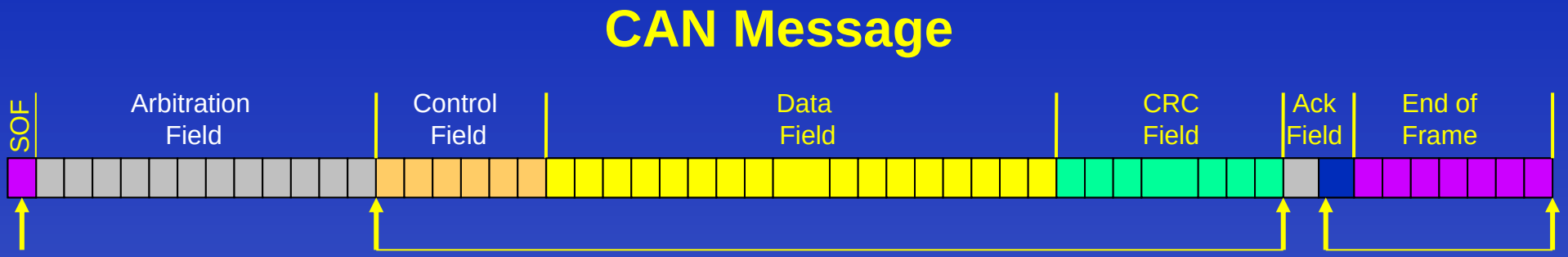
CAN Message



- 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



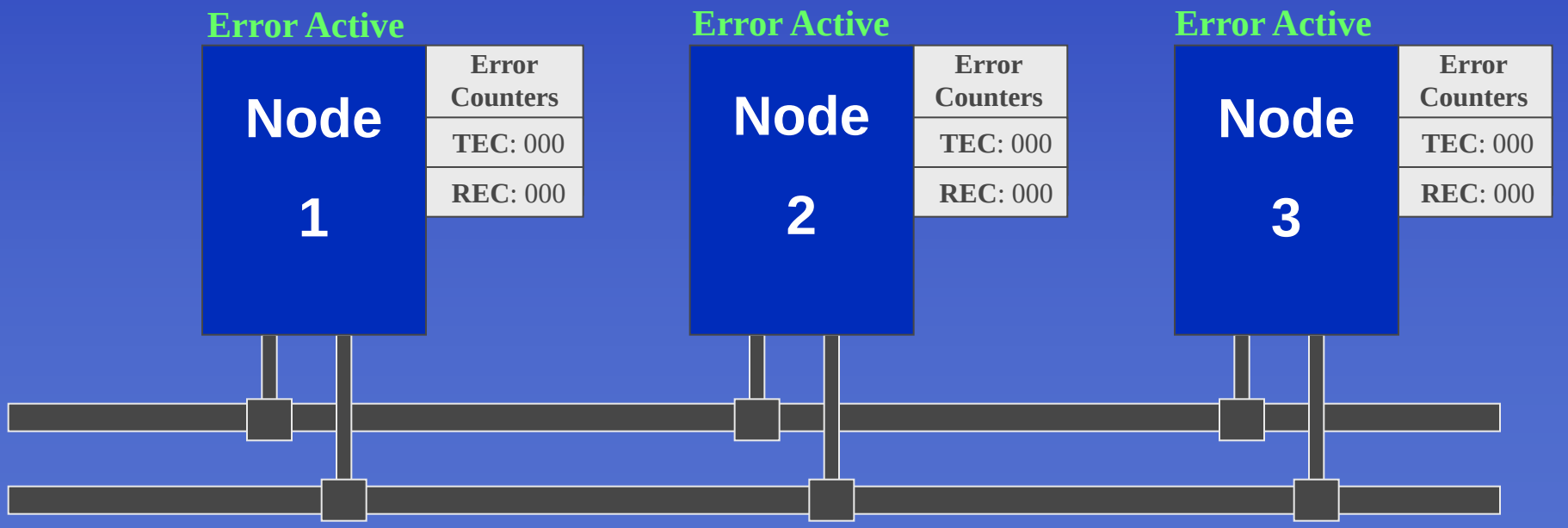
CAN 可偵測的各種錯誤



- 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)

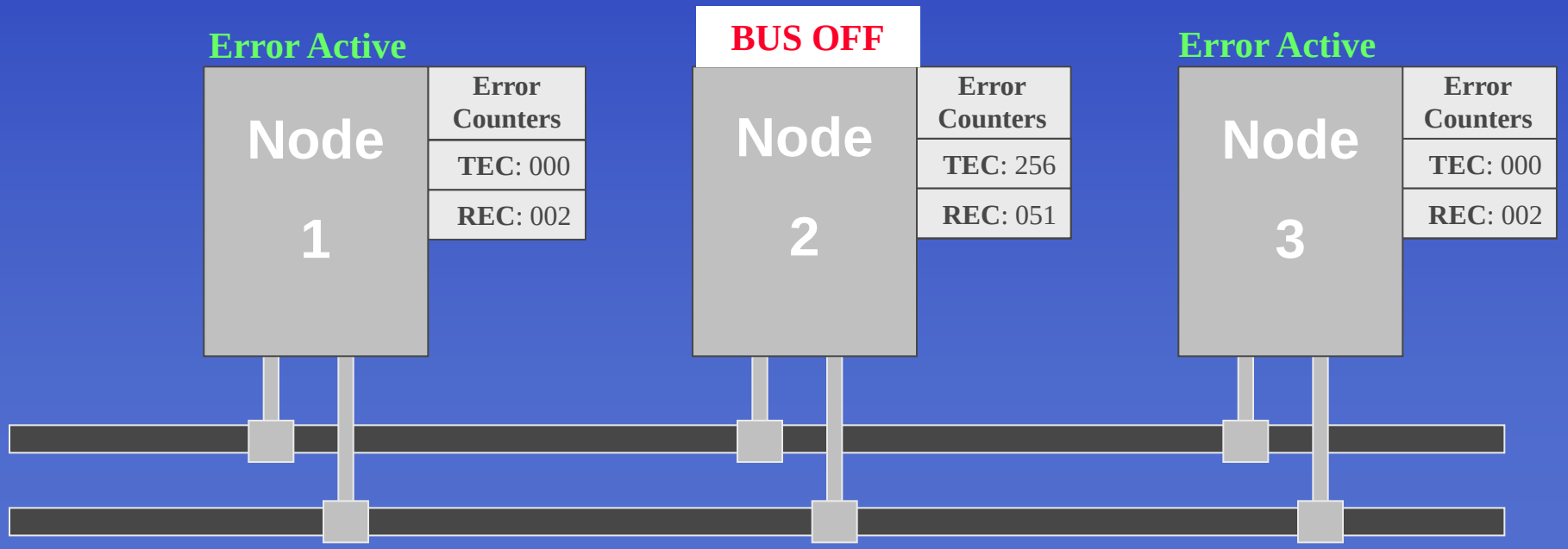
CAN 對錯誤的採取的限制

- 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為三種： Error Active 、Error Passive、 Bus-Off



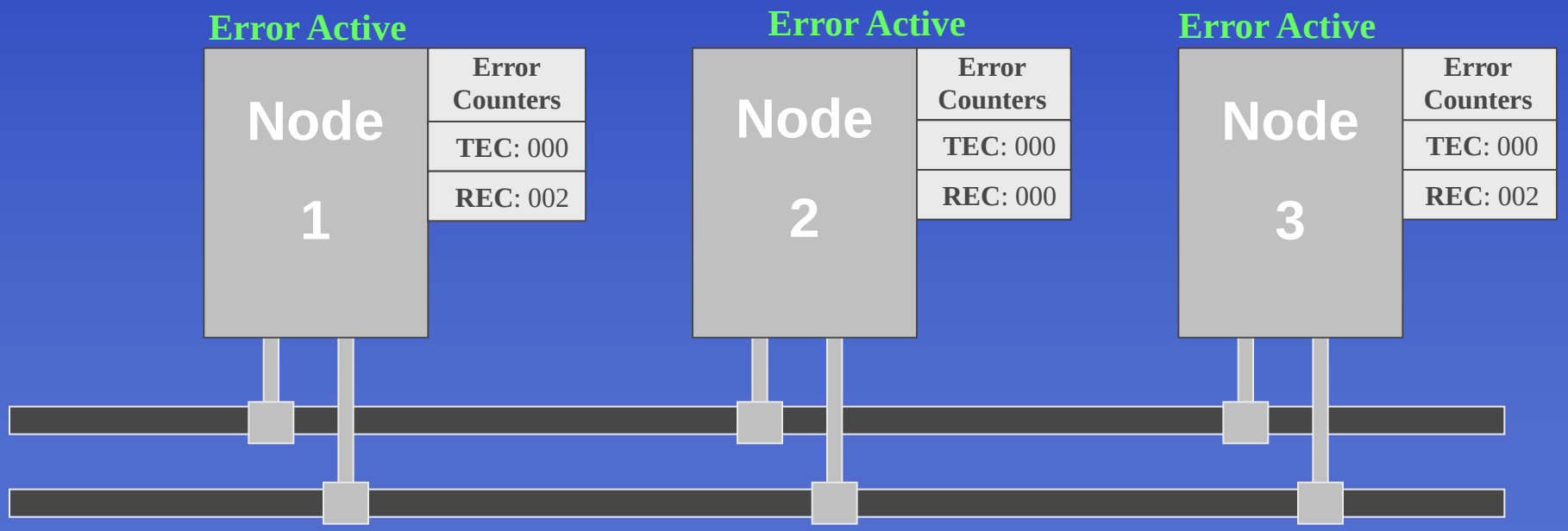
CAN 對錯誤的採取的限制

- 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



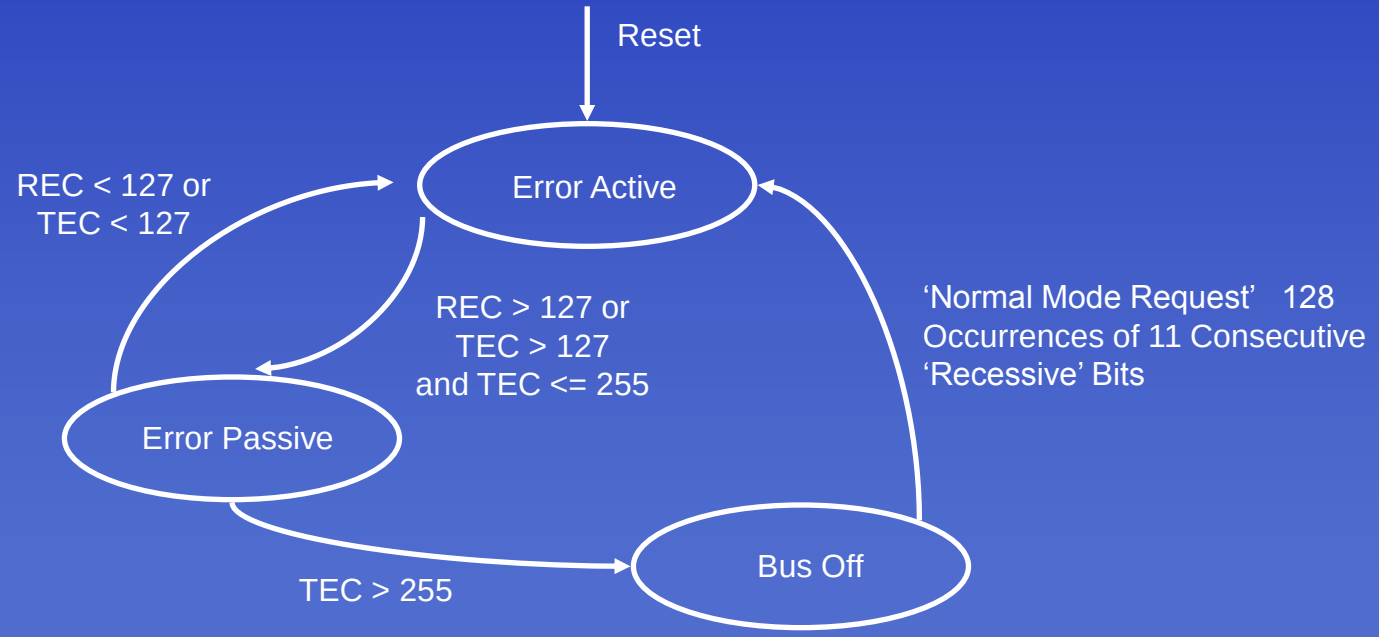
CAN 對錯誤的採取的限制

- 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) 將被依當時的狀態累加
- 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 Frame
- 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) 將被加累加
- 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

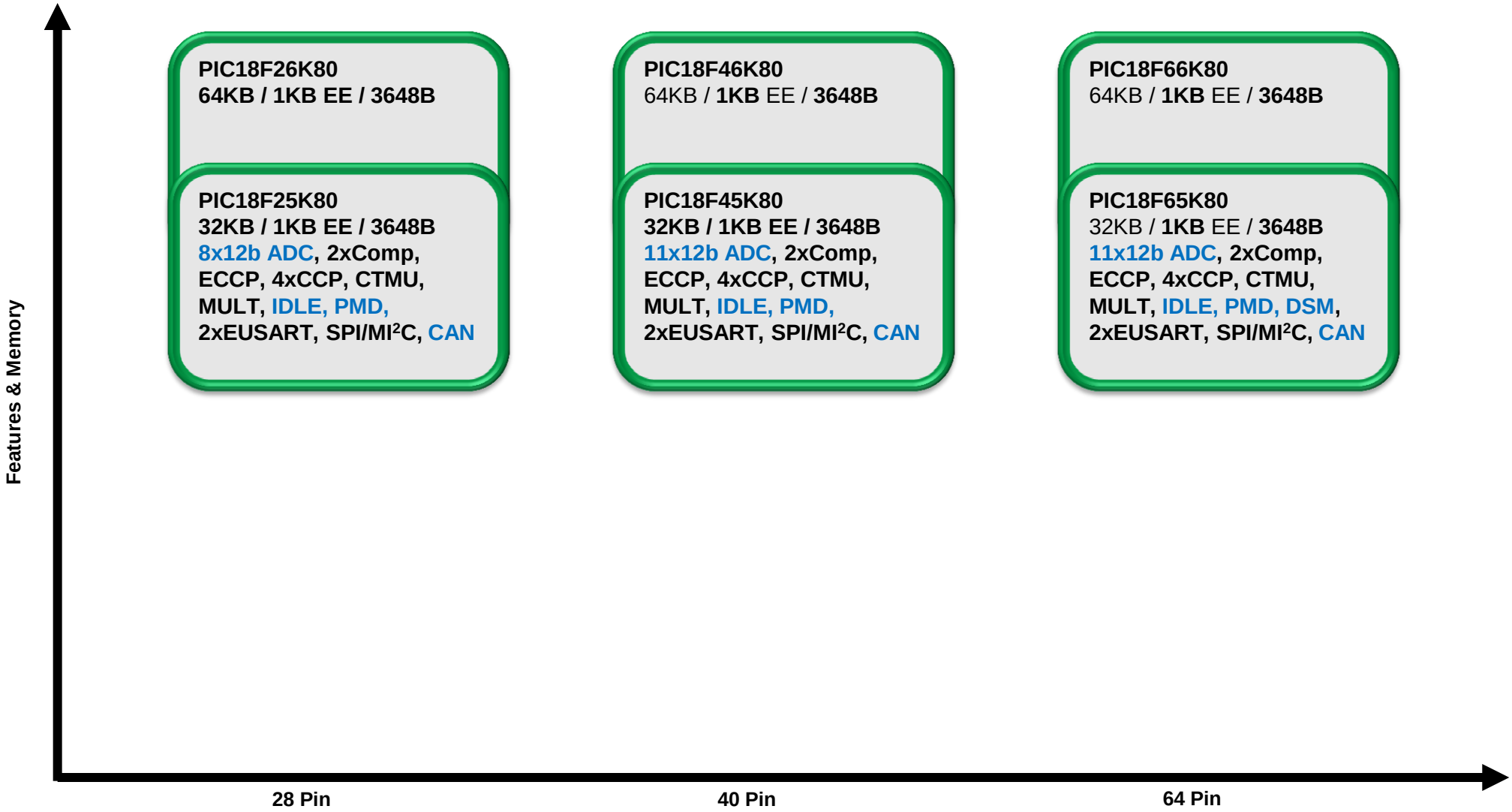
- CAN nodes 可依錯誤的嚴重程度來自動地減少它們對 bus 的控制能力
- 透過以下三種狀態的轉換可以達成這樣的要求
  - Error-Active (正常的操作模式)
  - Error-Passive (控制 bus 的能力被降低)
  - Bus-Off (無法控制 CAN bus)
- 不同 Error-States 間的差異在於錯誤的 node 該如何被限制行為
- CAN 對錯誤時的能力限制 (Fault Confinement) 可防止有缺陷的 node 使得整個網路當掉

Microchip CAN Solutions

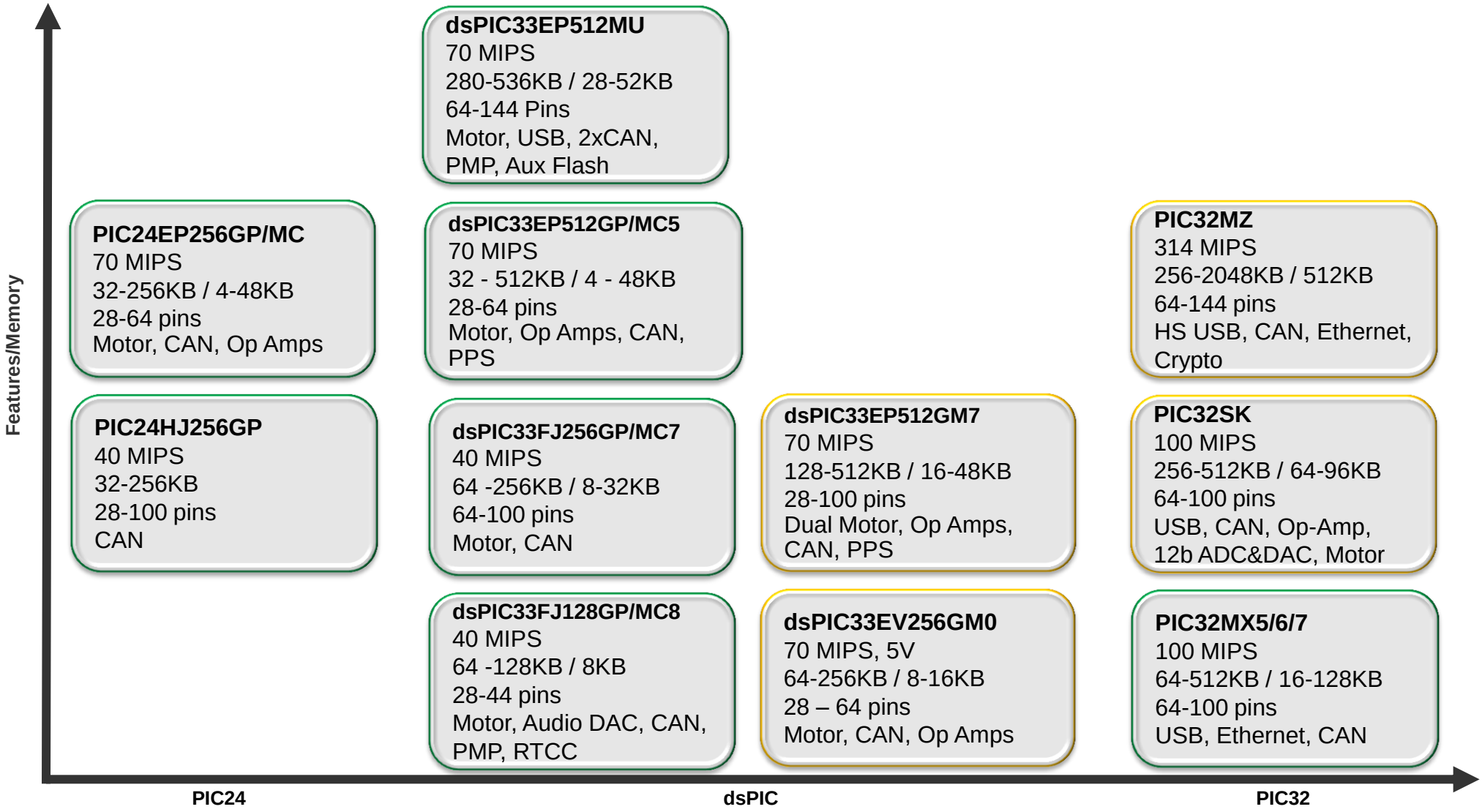
PIC18Fxx80 系列
內含 CAN/ECAN Controller 的產品

PIC18(L)FxxK80

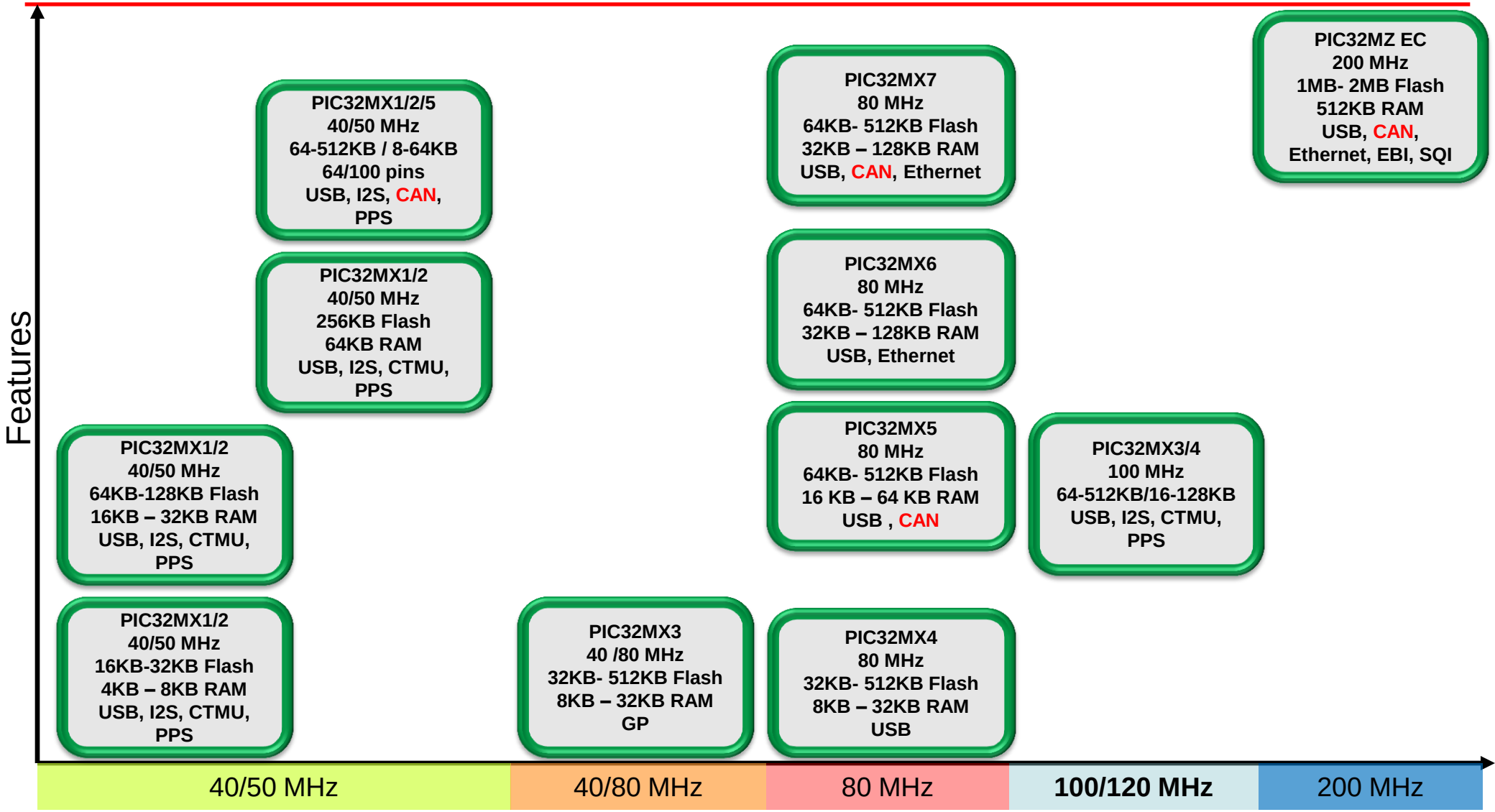
Integrated CAN Controler & 12b ADC



16-bit MCU CAN



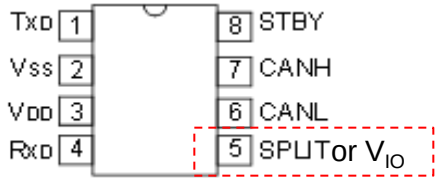
PIC32 Family



CAN Transceiver and Peripheral Products

MCP2561/2

High-Speed CAN Transceivers

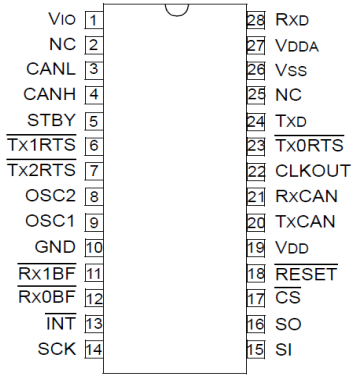


- Automotive Grade[1]
- CAN / ISO 11898 Compliant
- SPLIT Option (common mode stabilization)
- VIO Option (internal level shifting)
- CANH/L ESD ≥ 14kV
- Standby current < 5uA (Typ)
- -40˚C to +150˚C
- DFN/SOIC/PDIP 8L packages

[1]Meets OEM Hardware Requirements for LIN, CAN and FlexRay Interfaces in Automotive Applications", Version 1.3, 2012

MCP25625

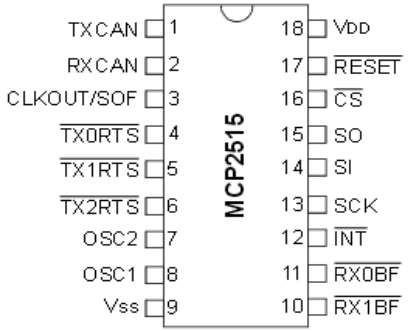
CAN Controller w/HS CAN Transceiver



- Automotive Grade
- CAN 2.0 up to 1 Mb/s Operation
- Very Low Standby Current (10 µA, typ)
- Up to 10 MHz SPI Clock Speed
- Interfaces to MCUs with 2.7V to 5.5V I/O
- Available in SSOP-28L & 6x6 QFN-28L
- -40˚C to +125˚C

MCP2515

Stand-alone CAN Controller



- CAN V2.0B at 1 Mb
- 2 receiver buffers, 6 filters and 2 masks
- SPI interface up to 10MHz
- Vdd 2.7V to 5.5V
- Sleep mode current < 1uA
- -40˚C to +125˚C
- 18L SOIC/PDIP, 20L TSSOP/QFN
- ISO-16845 DIS CAN conformance

MCP256(1/2)FD CAN FD Transceivers

•In Production

- **Ford and GM approved**
  - MCP2561FD – Split Option
    - Common mode stabilization
  - MCP2562FD – Vio Option
    - Level shift digital pins that interface to MCU Vdd
  - Up to 5Mb/s(Max) & 8Mb/s(Char) Data Rate
  - Low TXD to RXD Propagation Delay < 120ns
  - Loop Symmetry:
    - -10%/10% @ 2Mb/s, -20%/10% @ 5Mb/s, -30%/10% @ 8Mb/s
  - VDD=4.5V-5.5V Temp Rage=-40ºC to +150ºC

Microchip MCU RTOS and OSEK Support for Automotive

- Vector CANbedded
  - PIC18F
  - PIC24
  - dsPIC33E/dsPIC33F
  - Drivers based on ISO, OSEK, ASAM standard
- Vector osCAN
  - Preemptive real-time multitasking operating system
  - With properties that are optimized for use in Microcontrollers
  - Certificated to OSEK/VDX specification 2.2

內建於 PIC18F 系列的 CAN Solution (1)

- PIC18F 系列內建的 CAN Solution 分為兩大類
  - PIC18FXX8 - Standard CAN (Legacy Mode)
    - PIC18F248/258 , PIC18F448/458
      - 3 個 Transmitter Buffers
      - 2 個 Receiver Buffers & 1 MAB (Message Assembly Buffer)
      - 6 個 Acceptance Filter & 2 個 Mask

內建於 PIC18F 系列的 CAN Solution (2)

- Enhanced CAN - ECAN
  - PIC18F6680/6585 , PIC18F8680/8585
  - PIC18F2680/2585 , PIC18F4680/4585 (nano Watt)
  - PIC18F45K80/46K80 等新一代的 CAN MCU
    - Max Freq. = 64Mhz, 1.8V ~ 5.5V Operation Range
    - 12-bits ADC, ECCP, CTMU, nano Watt Technology
  - 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

內建於 PIC18FXX8 的 CAN Module

- Support Standard CAN (Mode 0 , legacy Mode)
  - 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

ECAN

Microchip 的 ECAN solution 概要

PICmicro®'s *ECAN* Module

- PIC18F 內建的 ***ECAN*** module 功能以 Mode 0 的操作模式來相容於早期的 PIC18FXX8 MCU 中的 Standard (Legacy) CAN module
- PIC18F 的 ECAN Engine 可運作於以下 3 種模式
  - "Mode 0" Legacy Mode
  - "Mode 1" Enhanced Mode
  - "Mode 2" FIFO Mode

- Modes 1 and 2 可以支援 DeviceNet 的協定

內建於 PIC18F 系列的 ECAN Solution

- Enhanced CAN 如何切換於各不同的 Mode
  - 多出來的 ECANCON 暫存器用於選擇 Mode 0 , 1 or 2
  - Power On 的預設模式為 Mode 0 (與 PIC18FXX8 相容)

R/W-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0
MDSEL1	MDSEL0	FIFOWM	EWIN4	EWIN3	EWIN2	EWIN1	EWIN0
bit 7							bit 0

bit 7-6 **MDSEL1:MDSEL0:** Mode Select bits

00 = Legacy mode (Mode 0, default)
01 = Enhanced Legacy mode (Mode 1)
10 = Enhanced FIFO mode (Mode 2)
11 = Reserved

Note: These bits can only be changed in Configuration mode. See Register 19-2 to change to Configuration mode.

bit 5 **FIFOWM:** FIFO High Water Mark bit(1)

1 = Will cause FIFO interrupt when one receive buffer remains(2)
0 = Will cause FIFO interrupt when four receive buffers remain

Note 1: This bit is used in Mode 2 only.
2: FIFO length of 4 or less will cause this bit to be set.

練習一 建立第一個 PIC18F46K80 專案

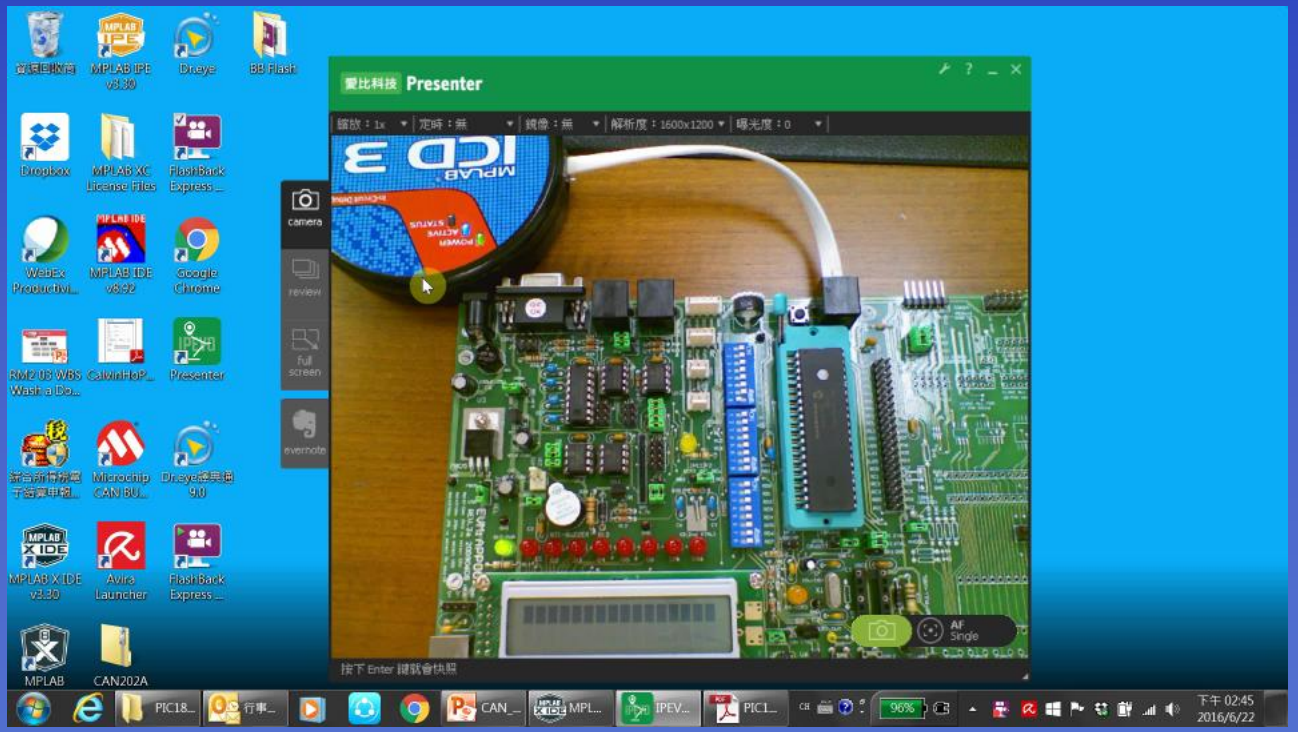
- 學習如何使用 Microchip 提供的 MCC (Microchip Code Configurator)
  - Configure ADC
  - Configure Timer 1
  - Configure I/O
- 以練習一來說明使用 PIC18F46K80 該注意的事項
- 讓學員了解如何使用 APP001 上的資源
  - Connectors (CAN、RS232、ICD)
  - LCD Display
  - LED (D9 @ RC1) 、VR1 & 相關 Jumper 的設定

練習一 建立第一個 PIC18F46K80 專案

- 學習 — 需要完成的功能
  - 一個能夠計時 200ms 並已中斷通知 CPU 的 Timer
    - 使用 Timer 1
  - Configure PIC18F46K80 的 ADC
    - 使用 AN0 來得到 VR1 產生的類比輸入
  - 使用 LED (D9 @ RC1) 來做 Alive signaling
    - 當 Timer1 的 200 ms計時完成後讓 D9 Toggle 一次
  - 加入支援 APP001 LCD Display 的副程式
    - 成功地加入 Microchip 提供的 LCD 副程式
    - 成功地使用 LCD 副程式做 Hello word 的顯示
    - 顯示 ADC 自 VR1 (AN0) 信號轉換的值 (0..4095)

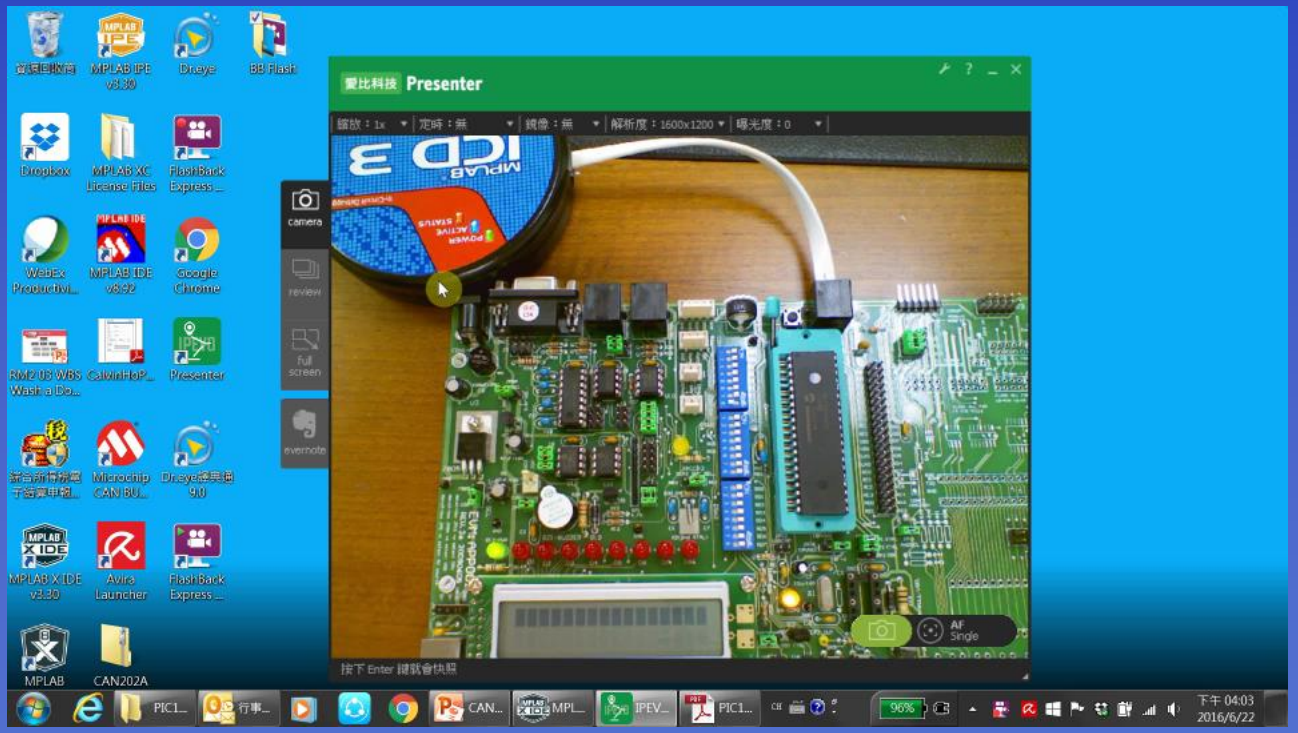
練習一的視訊教學 – Part1

- 有關 MPLAB X IDE, MCC 的介紹及專案建立
- 利用 MCC 規劃 Timer 1 & GPIO



練習一的視訊教學 – Part2

- APP001 LCD 副程式的加入
- 以 MCC configure 及操作 PIC18F46K80 的 12-bit ADC



APP001V3_LCD.c 內建的副程式

- 幾個重要的 APP001 LCD 副程式 (宣告於 APP001V3_LCD.h)
  - openLCD(void)
    - ✓ 設定 LCD 於要求的工作模式 (2-Lines, 5*7 Character)
  - setCursorLCD(unsigned char , unsigned char)
    - ✓ 設定 LCD 的游標位置 (X,Y)
  - putcLCD(unsigned char)
    - ✓ 印出字元
  - putsLCD(char *)
    - ✓ 將置於 Data Memory 中的字串輸出至 LCD
  - putrsLCD(const char *)
    - ✓ 將至於 Program Memory 中的字串輸出至 LCD
  - putNumberLCD(unsigned int, unsigned char)
    - ✓ 將 unsigned int 的數值以 10 進位的格式，顯示於 LCD (可指定要印出幾位數)
  - putHexCharLCD (unsigned char)
    - ✓ 將 char 的數值以 16 進位的方式顯示於 LCD
  - putHexIntLCD(unsigned int)
    - ✓ 將 unsigned int 的數值以 16 進位的方式顯示於 LCD

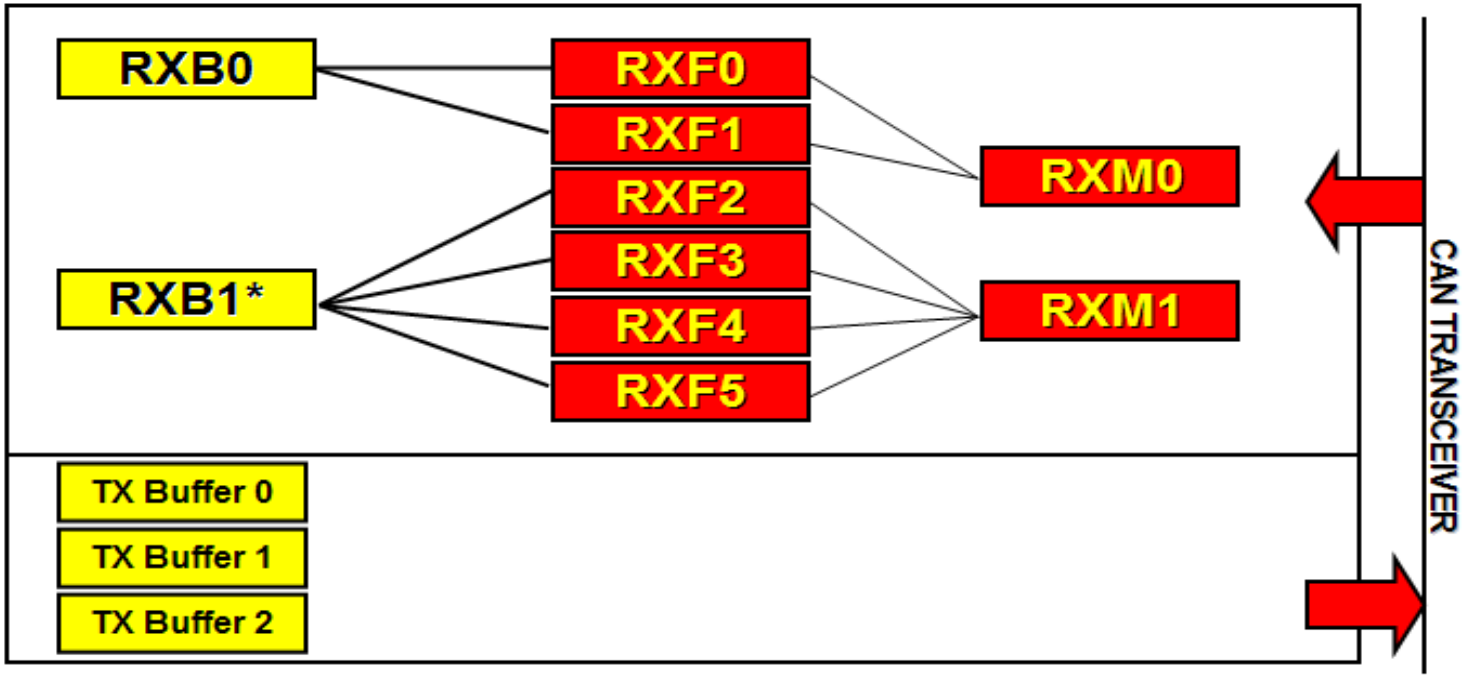
PIC18F CAN Family

Standard CAN 的介紹與操作

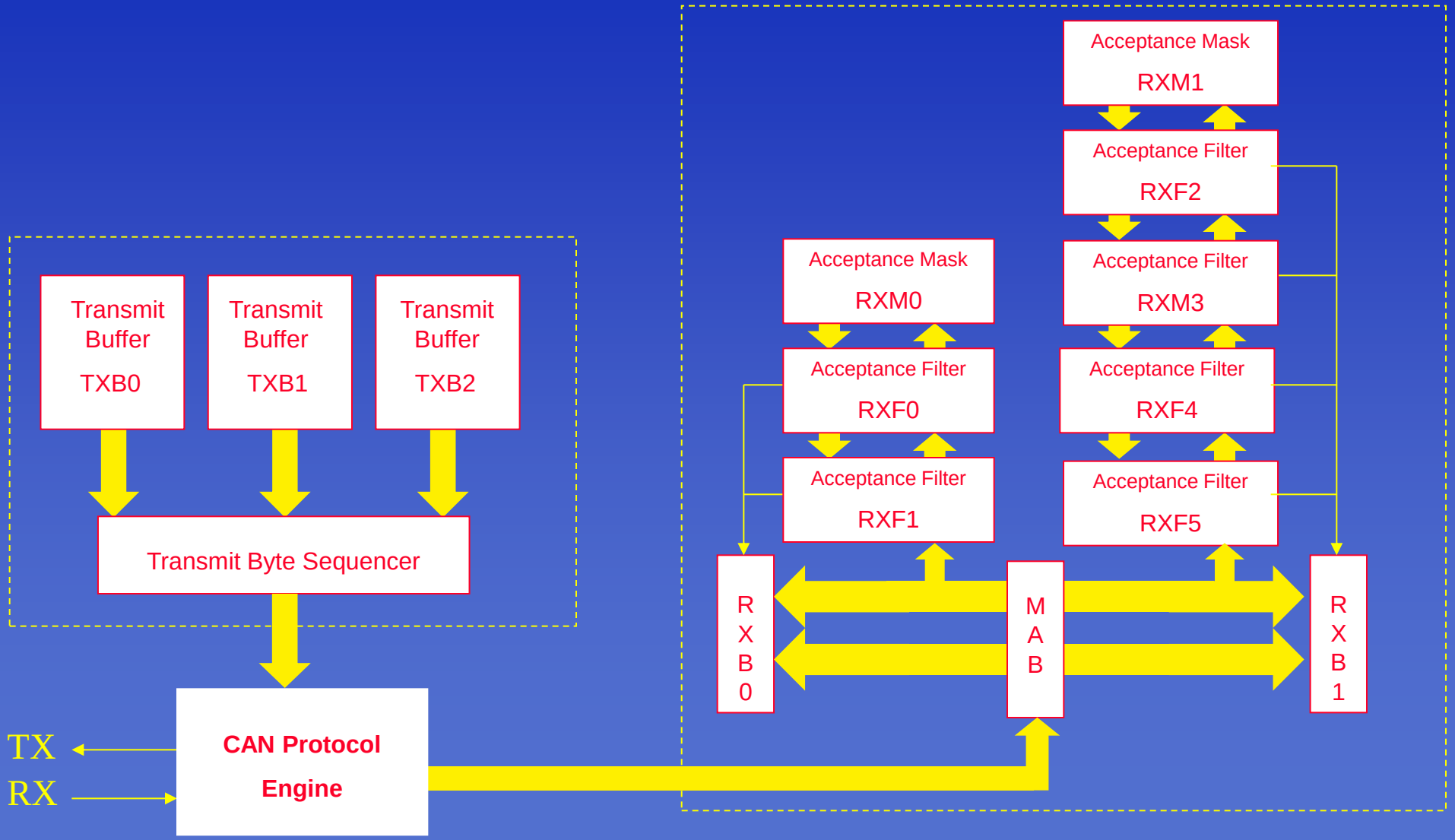
Mode 0

PIC18F ECAN™ Module Mode 0

- **Mode 0 (“Legacy mode”) Resources**
  - 3x TX Buffers (Dedicated)
  - 2x RX Buffers (Dedicated)
  - 1x Message Assembler Buffer
  - 6x Full Acceptance Filters
  - 2x Full Acceptance Masks

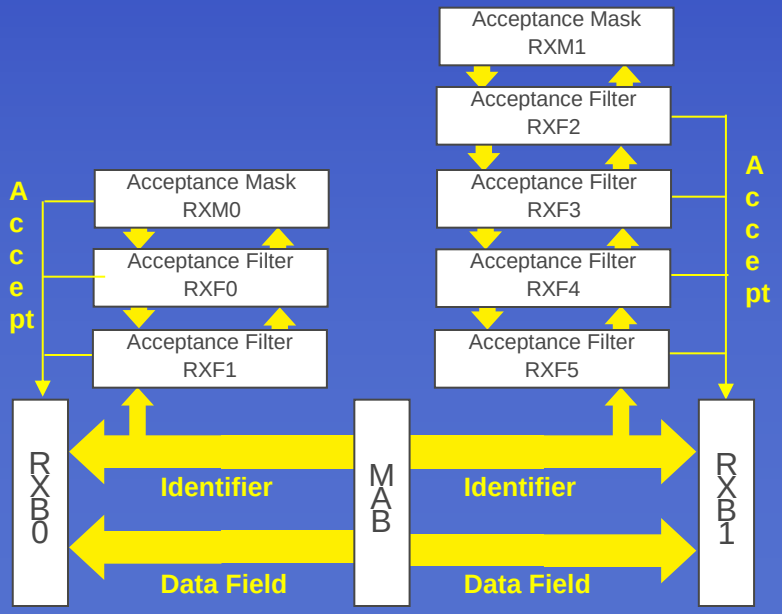


支援 Mode 0 – Standard CAN 的 CAN Buffers



Masks 和 Filters 的概念

- CAN 的 Message 會先被接收至 MAB ，經由 Mask & Filter 來決定在 MAB 中的 Message 是否要被接收 (以 ID – Identifier 來決定)
- 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
    - 在 Identifier 中對應的位元必需與 Filter 中對應的位元相等才被接受
  - '0' = 不使用 Filter 來過濾此位元 (無條件接受)
    - 在 Identifier 中對應的位元不管為 1 或 0 都會被接
- Identifier 通過了 Mask & Filter 的篩選後該筆資料才會被正式接收



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 與程式設計者的介面

- Total of 117 registers (addresses F00h - F78h) in Databank 15 (PIC18F46K80)
- Mode1 & Mode2 support register @ Bank 14
- 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

PIC18F CAN Module 可用的操作模式 (1)

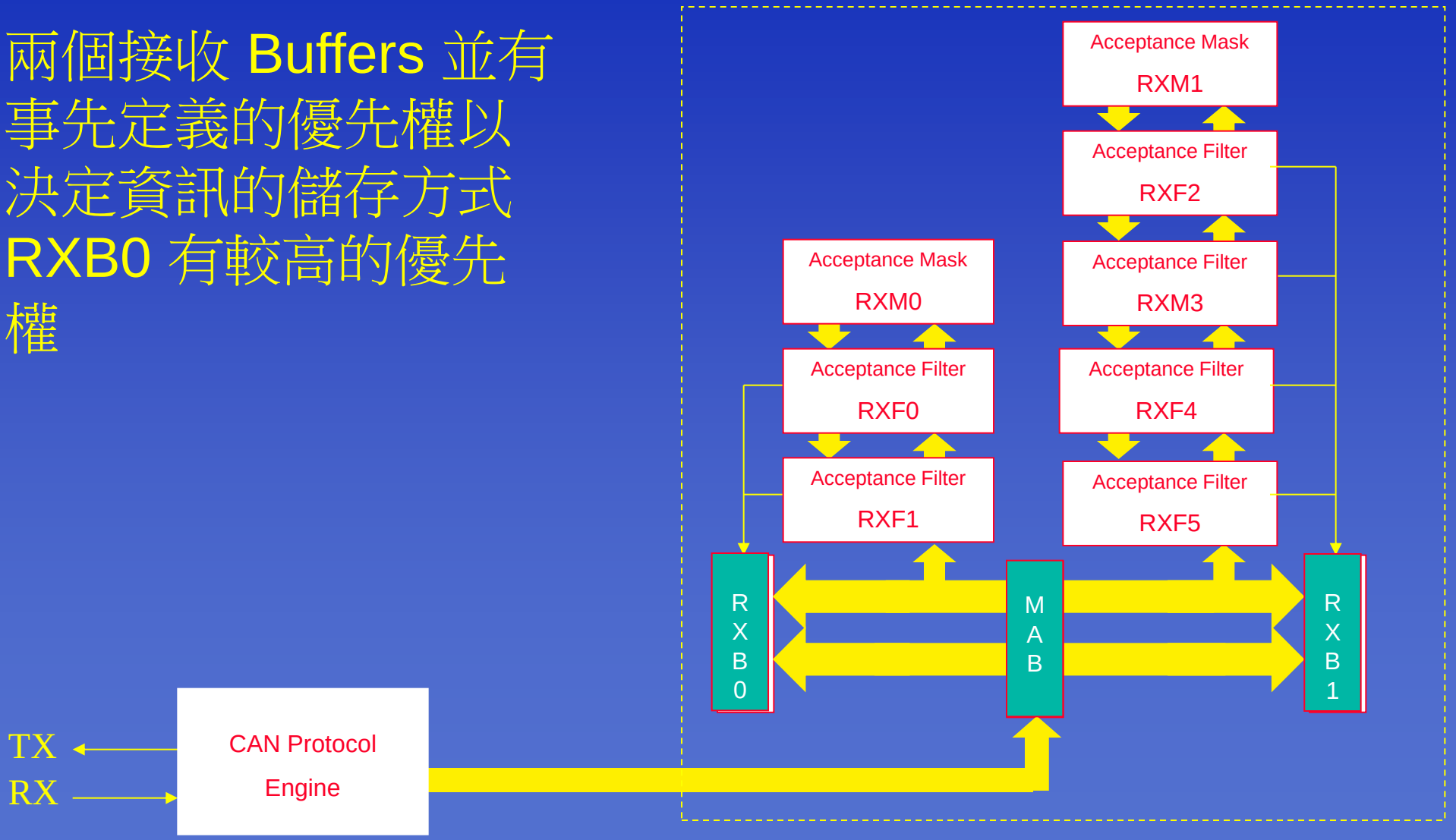
- Configuration Mode
  - Power-up 後 , CAN Module 會被設定於此模式
  - CAN Module 必須在此模式做適當的設定才可正確地工作
  - 與 CAN Module Configuration 相關的暫存器是受保護的
    - 3 Bit Timing Registers (BRGCONn, 1<= n <= 3)
    - 24 Identifier Acceptance Filter Registers (RXFnSIDH, RXFnSIDL, RXFnEIDH, RXFnEIDL (0 <= n <= 5)
    - 8 Identifier Acceptance Mask Registers (RXMnSIDH, RXMnSIDL, RXMnEIDH, RXMnEIDL (0 <= n <= 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 時, CAN Engine 的狀態是 “off-line”

PIC18F CAN Module 可用的操作模式 (2)

- 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 Modem
  - To participate in CAN bus communication

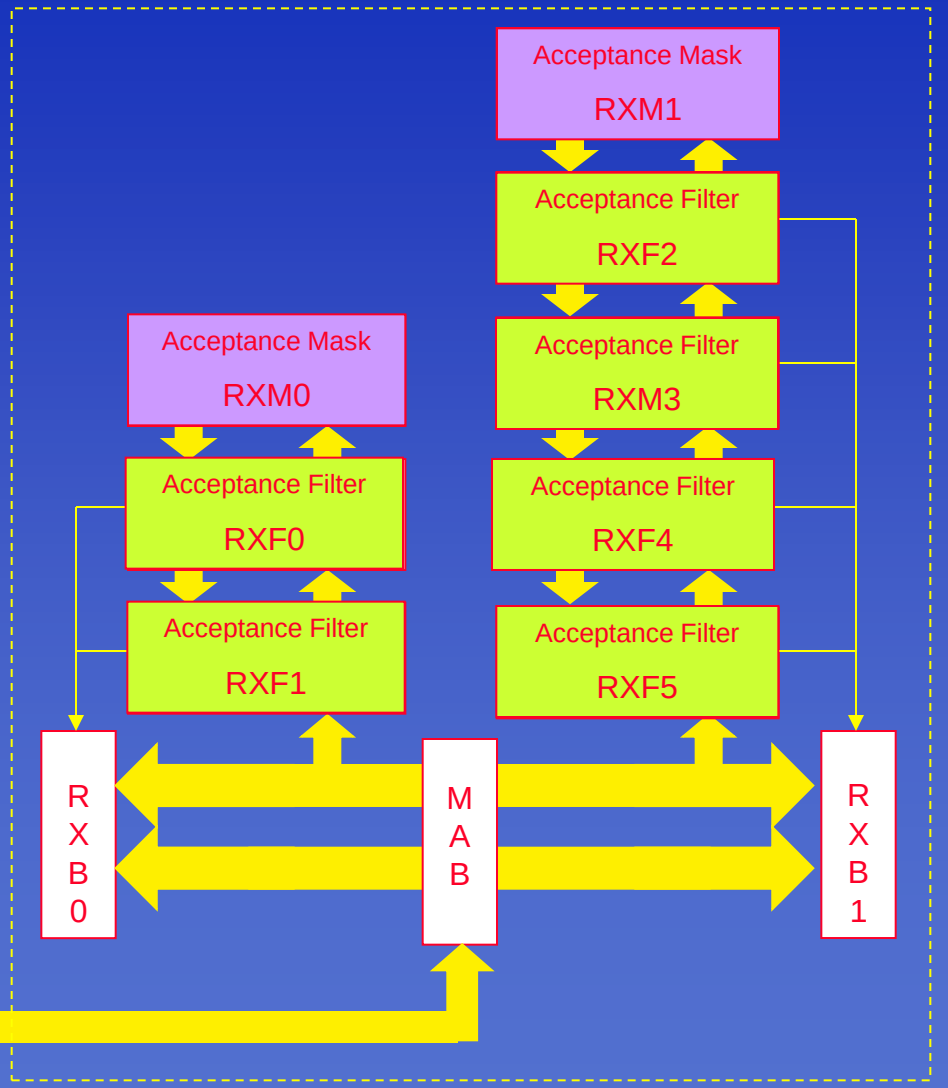
PIC18F Mode 0 的 Receive Buffers (1)

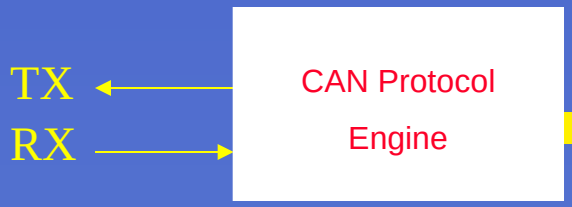
- 兩個接收 Buffers 並有事先定義的優先權以決定資訊的儲存方式
- RXB0 有較高的優先權



PIC18F Mode 0 的 Receive Buffers (2)

- 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
  - ✓ PIC18FXXK80 的中斷致能旗標在 PIE5
- 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

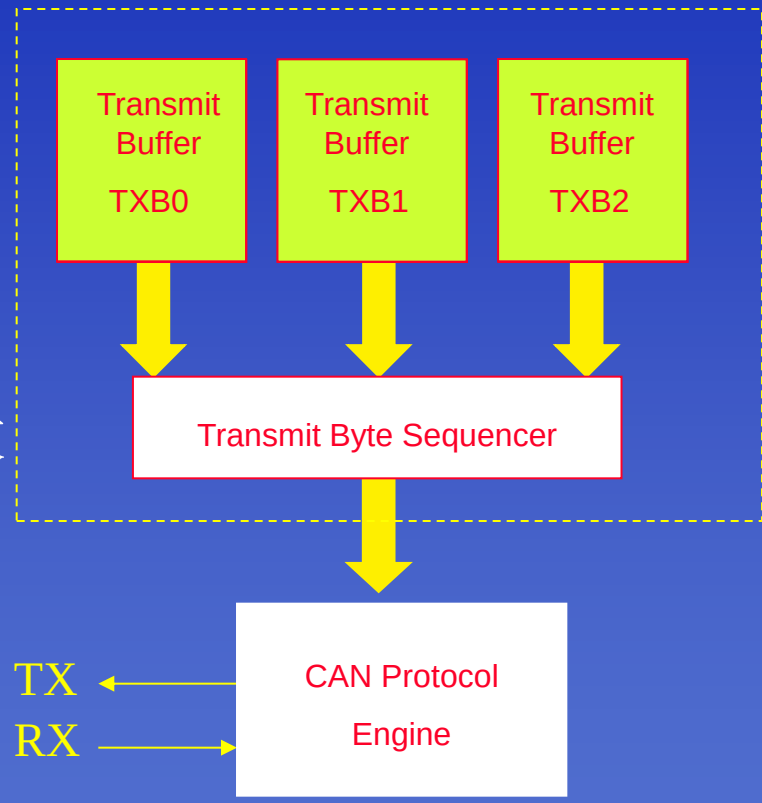




PIC18F Mode 0 的 Transmit Buffers

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 for Mode 0 (1)

- AN738 提供了完整的 Mode 0 - Standard 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
- CAN202B RTC 課程將 AN738 改成 support PICF18FXXK80 以及使用 MPLAB XC8

Microchip 的 AN738 for Mode 0 (2)

- 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 for Mode 0 (3)

- ✓ 使用 CAN Library , PIC18F 系列的 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) ;
```

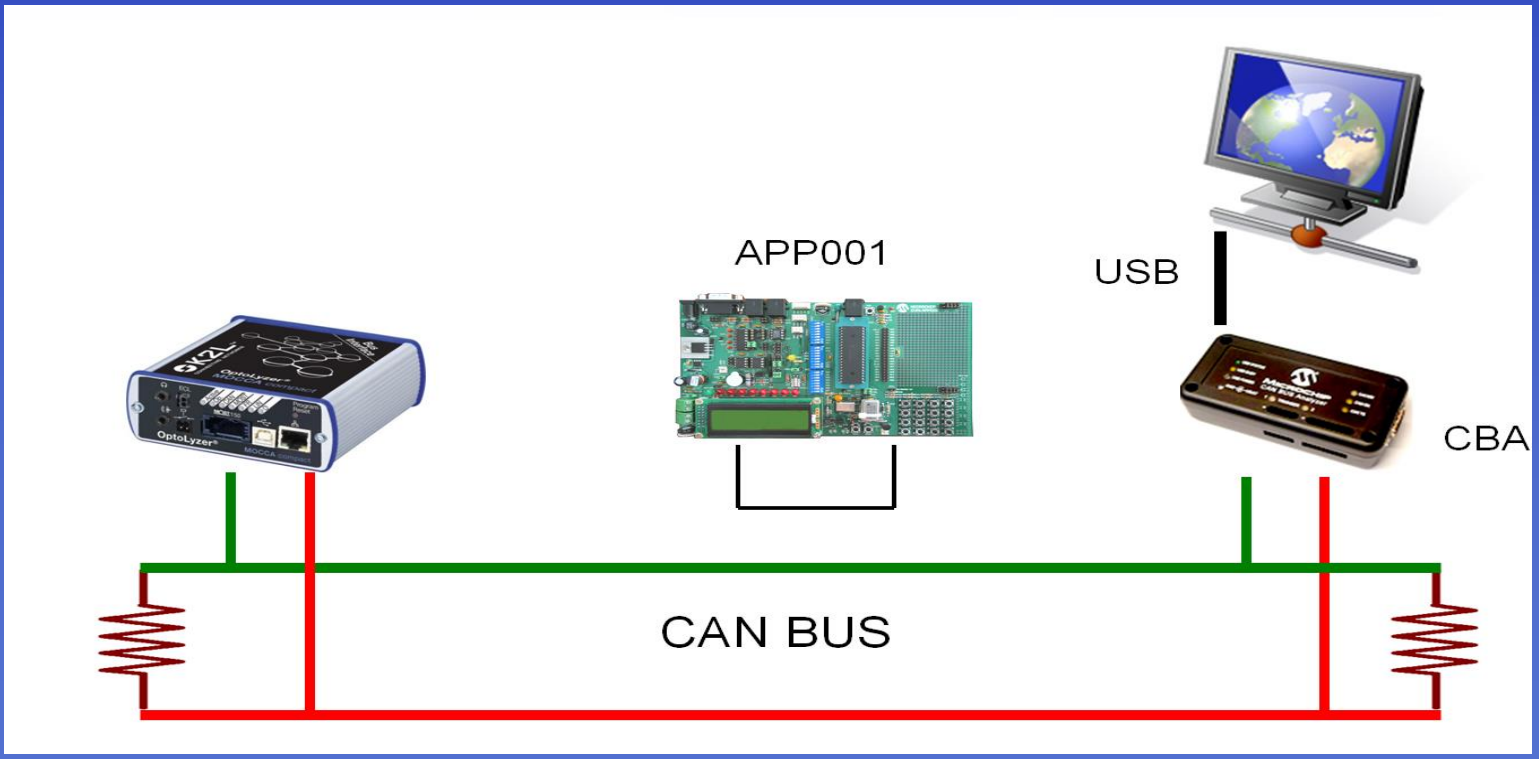
Microchip 的 AN738 for Mode 0 (4)

- 使用 Functions Library 所需注意的事項
  - 除了對 Controller 的規格需有所瞭解外, 並且要參考到 Function Library 已經事先定義好的參數
    - CAN18FXXK80_ECAN_Mode0.h
    - CAN18FXX80_ECAN_Mode0.h
    - CAN18FXX8_CAN_Mode0.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
};
```

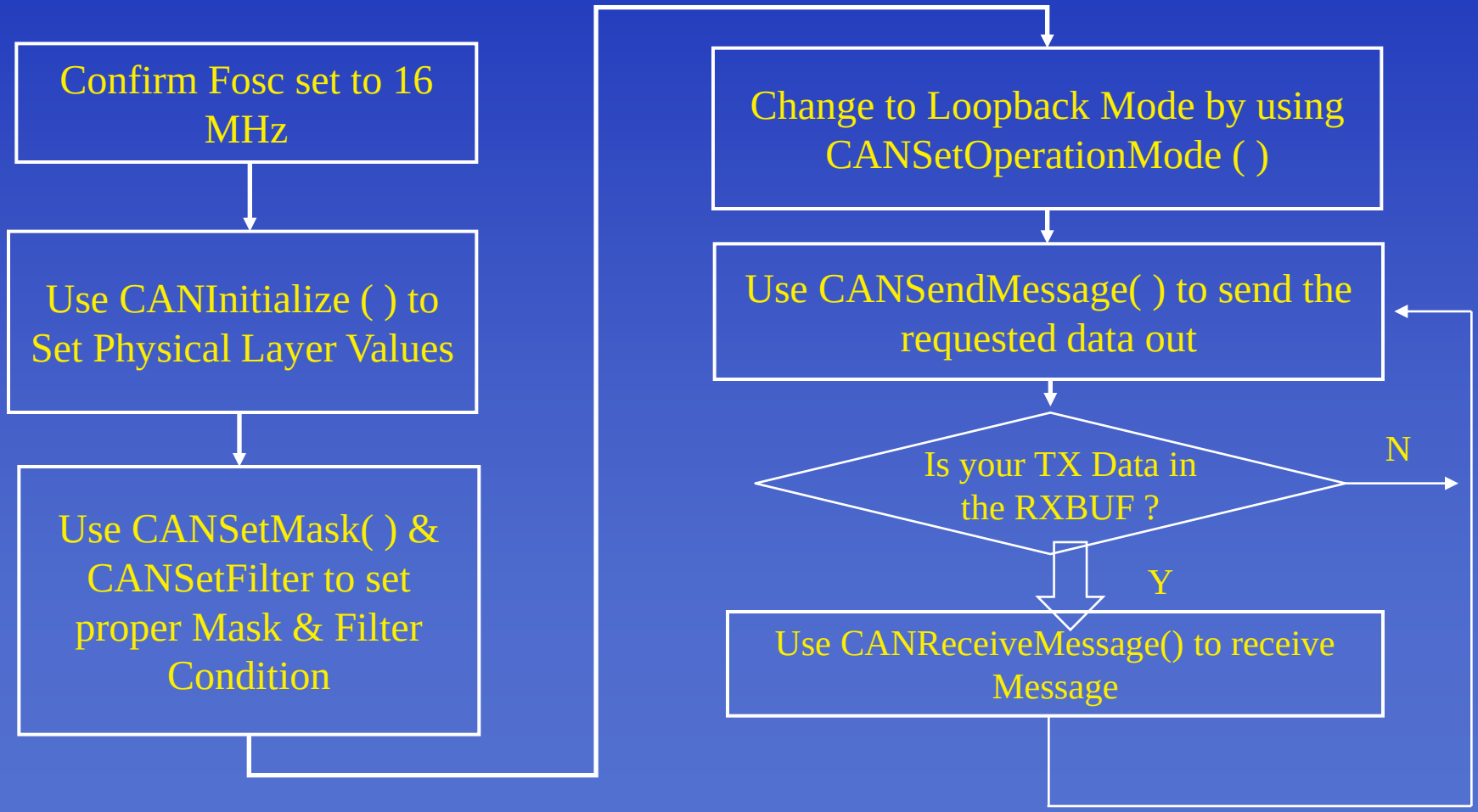
CAN Module 的 Loopback Mode

- 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



Configure PIC18F at 'Loopback Mode'

- Simple Message in Loopback Mode



Configure PIC18F at ‘Loopback Mode’

- AN738 中可用的副程式 (1)
  - CANInitialize( )
    - 用來規劃 PIC18F8680 的工作條件
  - CANSetOperationMode( )
    - 用來設定 PIC18F8680 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

Configure PIC18F at ‘Loopback Mode’

- AN738 中可用的副程式 (2)
  - 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

練習二：將 PIC18F46K80 設定並操作於 ‘Loopback Mode’

- 練習二的目標 : Message 的傳送/接收練習
  - 熟悉 AN738 所提供的 Functions
  - 以 Lookback mode 使得 CAN Module 可以不將資料由 TXCAN 送至外部 , 而是經由內部的路徑轉至 Receive Buffer 的 MAB
  - 將 PIC18F46K80 作正確的設定, 且傳送出一個簡單的 Message
  - 將 PIC18F46K80 的 Mask & Filter 正確設定，讓 PIC18F 能接收到正確且必要的資料
    - 更改 Mask & Filter 的設定，觀察接收的狀況

練習二：將 PIC18F46K80 設定並操作於 ‘Loopback Mode’

- 練習二要求完成之功能:
  - 以 Lookback mode 使得 CAN Module 可以在沒有其他 CAN node 的連線下，將送出的資料經由內部的路徑轉至 MAB
    - 傳送的格式為 STANDARD Data FRAME
    - Identifier 由 RTC 講師指定各個 CAN Node，資料長度為 8 個 Byte
    - 將 8 個 Bytes 的資料由 TX Buffer 送出
      - 01,02,03,04,05,06,ADRESL,ADRESH
  - 進入 MAB 的 Message 經由 Mask & Filter 過濾後，取出預設的有意義資訊 (ADC Value)
  - 將 Message 的 ID (Identifier) 顯示於 APP001 LCD 的第一行 (16 進位)
  - 將接收到的 ADC Value 顯示至 APP001 LCD 的第二行 (10 進位)

練習二：將 PIC18F46K80 設定並操作於 ‘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！

練習二：將 PIC18F46K80 設定並操作於 'Loopback Mode'

- 資料進入 RXB0 後的處置 -> if CANIsRxReady( ) != 0
  - 使用 CANReceiveMessage( ) 將進入 RX Buffer 的資料讀取
  - 將收到的 Message ID -> MESSAGE_ID1 顯示於 LCD Display
  - 將收到的 Message Data 拆解，將 ADC 的資料也顯示於 LCD
- 若一切正常工作, 在 LCD Display 將有以下的資料顯示
  - 第一行左邊顯示傳送出去的 Message ID (十六進位)
  - 第一行右邊顯示10 進位的 A/D 轉換值 [0.. 4095]
  - 第二行左邊顯示收到的 Message ID (十六進位)
  - 第二行右邊顯示收到的 ADC 值，以10 進位的格式顯示

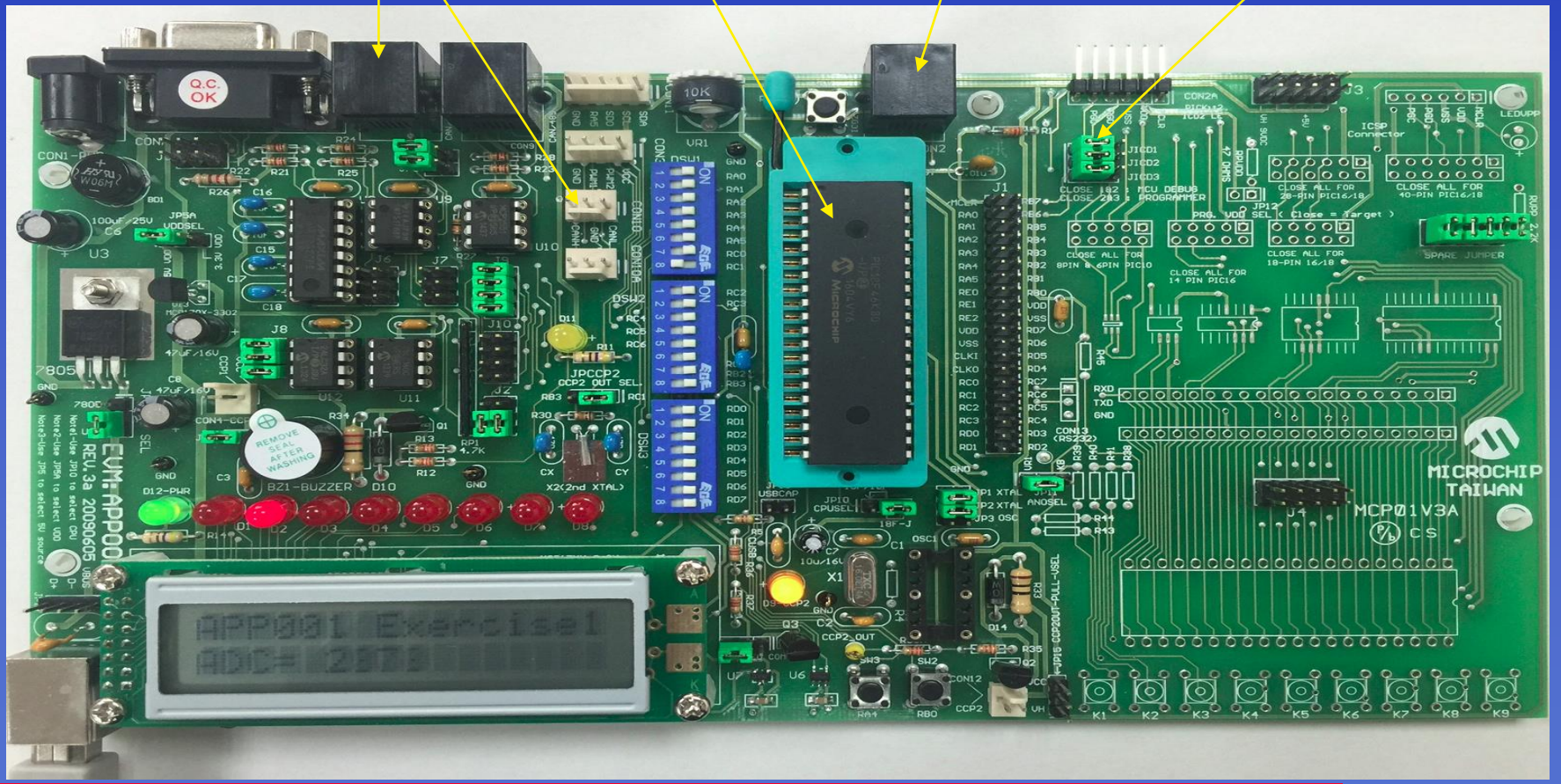
APP001 實驗板

CAN Connector

PIC18F46K80

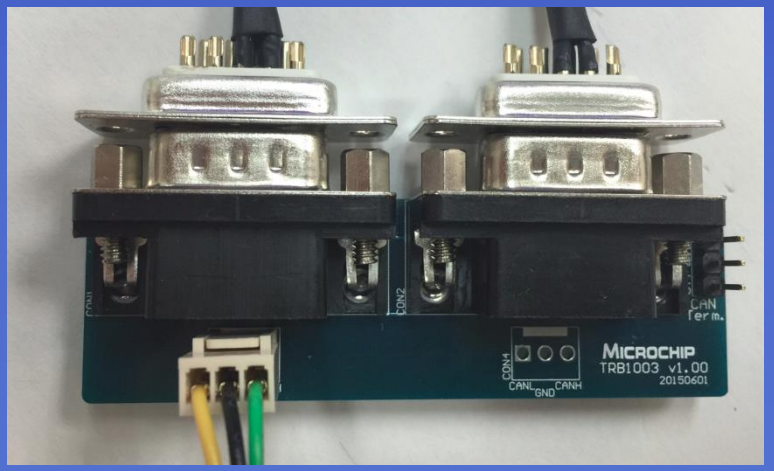
ICD3 Connector

PGC/PGD Select



APP001 to CAN Bus 轉接板

- 早期的 APP001 使用 RJ11 的 Connector – CON8 & CON9 來連接 CAN 網路
- 為了與符合 ISO11898-2 規範，使用 DB9 connector 的裝置連接，故使用 Molex-3 to DB9 的轉換板做信號轉接
  - TRB1003 V1.00
  - 接至 APP001 的 CON10 or CON10A
  - On-Board 120 Ohms 終端電阻，可以 Enable or Disable !



深入探討 PIC18F MCU CAN Module

&

Microchip AN738 for PIC18F46K80

以 PIC18F46K80 為 Target MCU

CAN Module Initialize Registers (Mode 0)

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 Initialize Registers (Mode 0)

CANSTAT- CAN Status Register

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

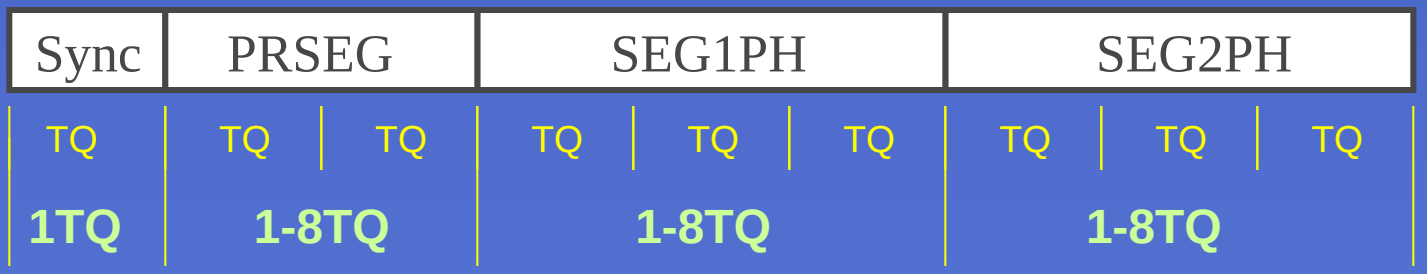
- OPMODE<2:0> 反應 CAN Module 現在所處的操作模式
  - Configuration, Listen Only, Loopback, Disable, Normal
- ICODE<2:0> returns current interrupt code bits
  - 可以將此 3 位元直接 copy 至 CANCON.WIN<2:0> bits 來加速對 buffer 存取的效率

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 Initialize Registers

Bit Timing 的詳細資訊

- 一個位元所佔的時間事實上是由許多Time Quanta (TQ) 組合而成
  - 最大的 TQ 數量 = 25
  - Baud Rate Prescaler (BRP) 用來設定 TQ.
- 每一個位元又可分成四個 segments
  - 每個 Segment 佔用 1 或多個 TQ
- 確實地符合 CAN bit timing 的規格需求，可以實際情況調整
  - Sync segment 永遠只佔用 1 個 TQ



CAN Module Initialize Register

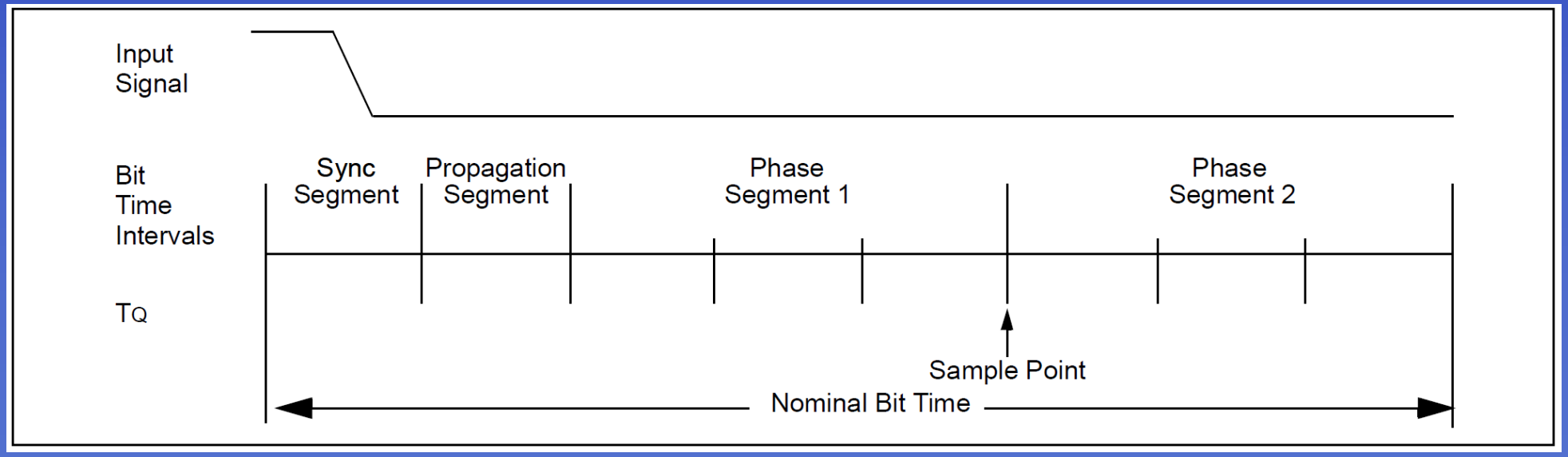
BRGCON1 - Baud Rate Control Register 1

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) /Fosc
  - 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

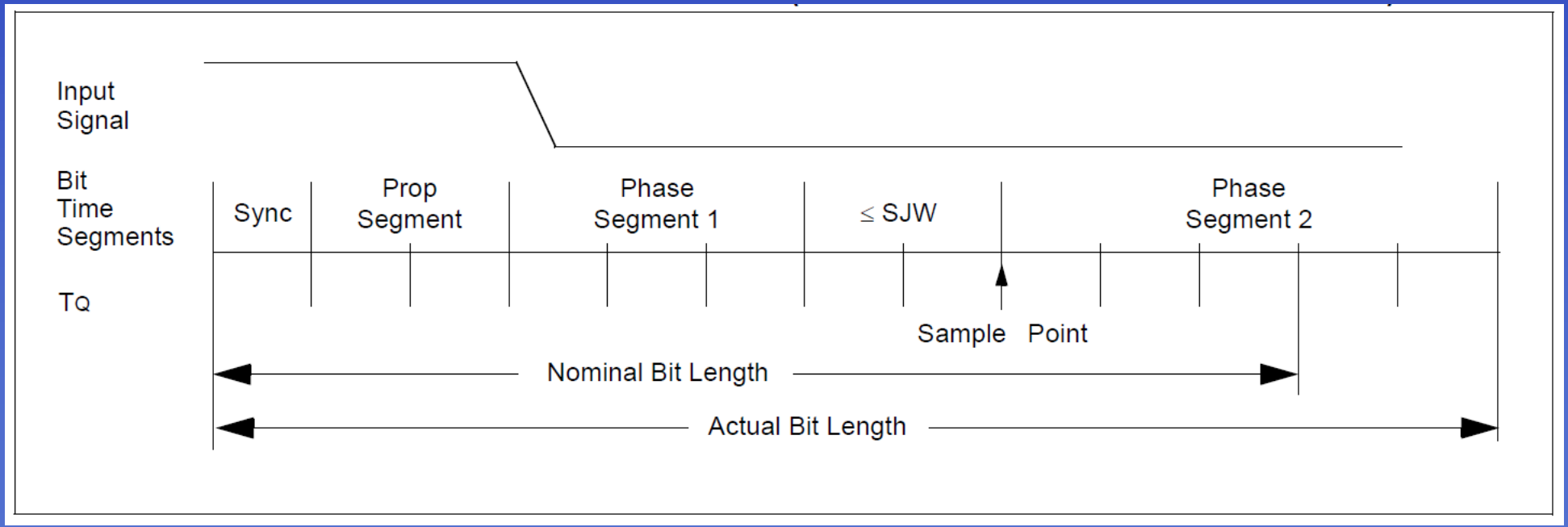
SJW 與位元同步的關係

- 理想的 CAN 資料傳輸如下，輸入訊號剛好落在 Sync Segment 中 (固定為 1 TQ)
- 但是，傳輸線越長，則信號延遲就越多！而且 CAN Engine 彼此的 clock 也有不同的誤差！
- 因為 CAN 沒有傳遞 clock 信號給對方，所以雙方只能藉由信號的 Edge 來同步
- 每個 Message 的 SOF 位元即為 CAN Bus 的硬體同步
- SOF 之後即使用信號的 Edge + SJW 來做同步 (所以 CAN 有位元填充的機制)



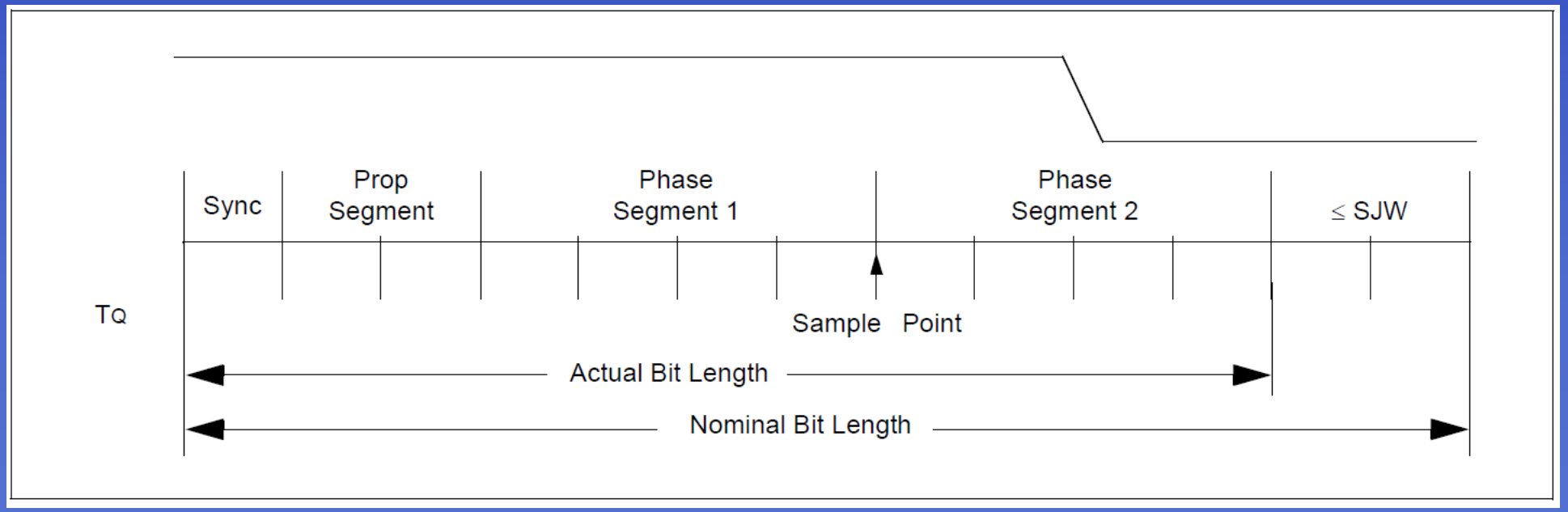
SJW 與位元同步的關係(2)

- 當信號的負緣落到了 Phase Segment 1，代表輸入信號落後了這個 CAN node 的 Timing
  - 此時使用 SJW 來延長 Phase Segment 1
  - 延長時間的最大值為 SJW 設定的時間 (最多4TQ)



SJW 與位元同步的關係(3)

- 當信號的負緣落到了 Phase Segment 2，代表輸入信號超前了這個 CAN node 的 Timing
  - 此時使用 SJW 來縮短 Phase Segment 1
  - 能縮短時間的最大值為 SJW 設定的時間 (最多4TQ)
  - 這也是為何 Phase Segment 2 的時間要 >= SJW 的主因



CAN Module Initialize Register

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 >= 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 >= SEG2PH
  - PRSEG + SEG1PH >= Tdelay

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
WAKDIS	WAKFIL				SEG2PH2	SEG2PH1	SEG2PH0
bit 7							bit 0

- WAKDIS – Wake-up Disable 位元
  - 1 = Disable CAN bus activity 喚醒的功能
  - 0 = Enable CAN bus activity 喚醒的功能
- WAKFIL - Enables/Disables CAN bus line filter 被用於 CPU 的 wake-up
- SEG2PH<2:0> 定義 phase segment 2 的寬度
  - SEG2PH >= SJW
  - PRSEG + SEG1PH >= SEG2PH
  - Overlaps with SJW and shrinks as SJW changes

CAN Module Initialization

計算各 Bit Rate Register 的設定值

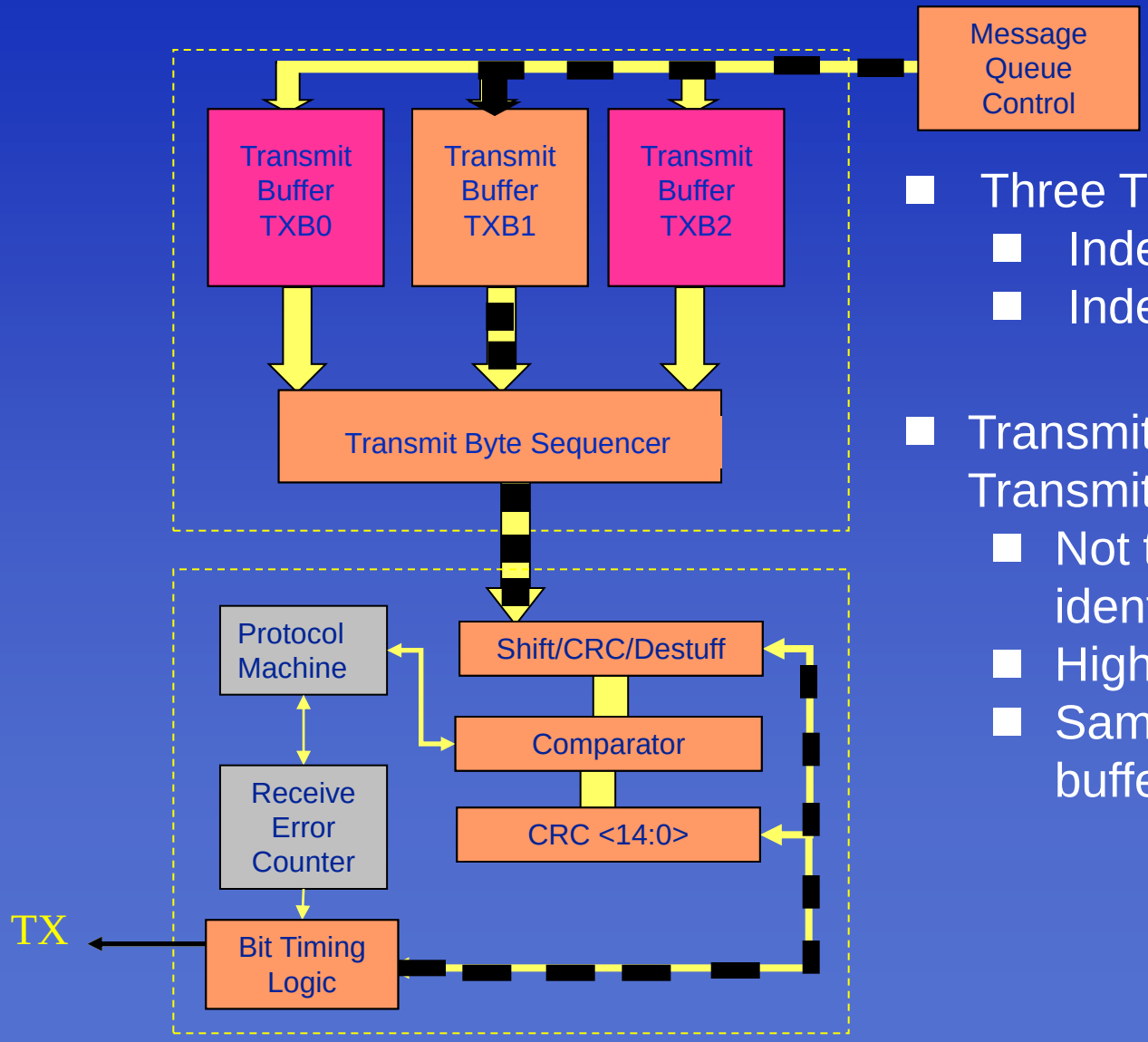
- 計算條件與公式：
  - 需要的 baud rate 為 B
  - 進入 CAN Engine 的頻率為 Fosc, 週期為 Tosc
    - 每一個 bit 所站時間 = Tbit = 1 / B
    - TQ = 2 * (BRP + 1) * Tosc (公式 1)
    - TQ = Tbit / N, N 是每一個 bit 包含的 TQ 數量 (公式 2)
    - BRP = ((Fosc/N) / (2 * B)) - 1 (公式 2 代入 公式 1)
    - 在 8 <= N <= 25 的條件下，選擇一個最大可能的 N，並且滿足讓計算後的 BRP 值為整數
    - 將 BRP 寫入 BRGCON1
      - Phase Segment 2 = SJW = N * 0.30 (取整數)
        - SJW 的最大值為 4 TQ
      - Prop Segment = 2 (依據傳輸線的長度可能需要調整)
      - Phase Segment1 = N - 1 - 2 - Phase Segment 2

CAN Module Initialization

計算各 Bit Rate Register 的設定值- 範例

- 在 CPU 執行頻率為 16 Mhz 的條件下規畫出 125 kbps 的 CAN Bit Rate
  - Tbit = 1 / 125 kbps = 8 uS
  - 選取 N = 16,
    - (Fosc/N) / (2 * B) = (16MHz/16)/(2 *125kbps) = 4
  - BRP = 4 - 1 = 3 (剛好整數)
  - Phase Segment 2 = SJW = 16 * 0.30 = 5
    - SJW 的最大值為 4
  - Prop Segment = 2
  - Phase Segment 1 = 16 - 1 - 2 - 5 = 8
- 最後寫入 Bit Rate 控制暫存器的結果:
  - BRP<5:0> = 3, SEG1PH = 7, SEG2PH = 4, SJW = 3

Transmitter Buffers @ PIC18F46K80 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 (Mode0)

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

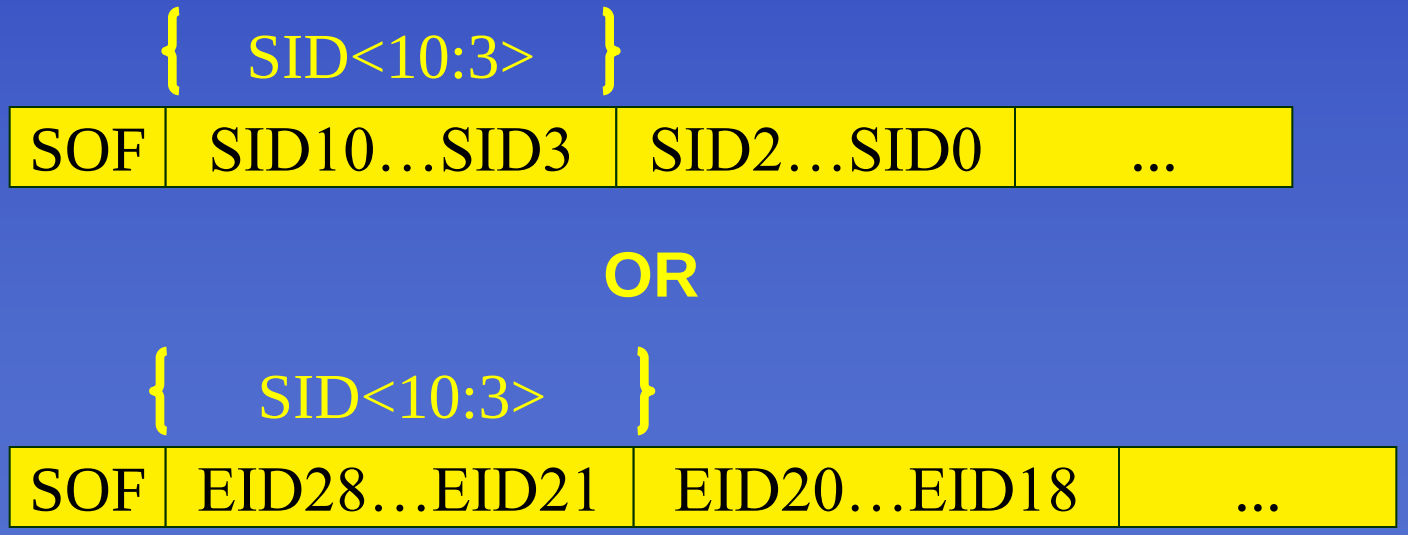
- Three transmit buffers (0 <= n <= 2)
- TXABT returns transmission abort status
- TXLARB returns arbitration status
- TXERR returns bus error status
- TXREQ requests transmission of current message
- TXPRI<1:0> defines priority of current message
  - Higher the number, higher the priority

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

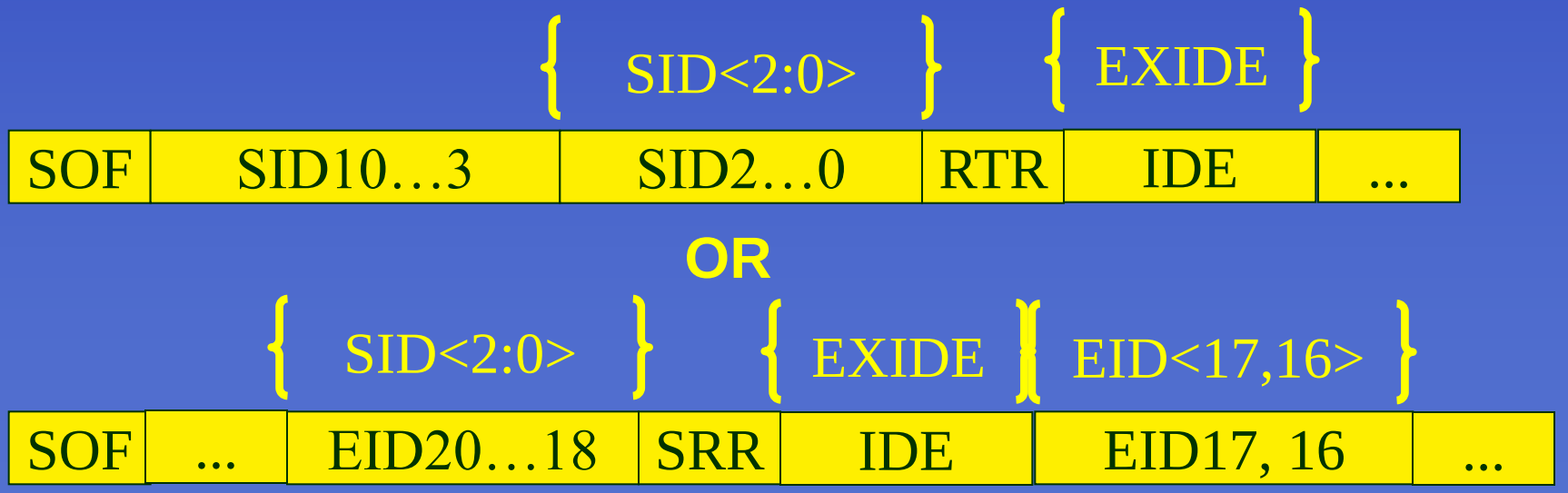


CAN Module Buffer Access

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



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

- ✓ PIC18F46K80 在 Mode-0 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列舉值傳入
- 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( )

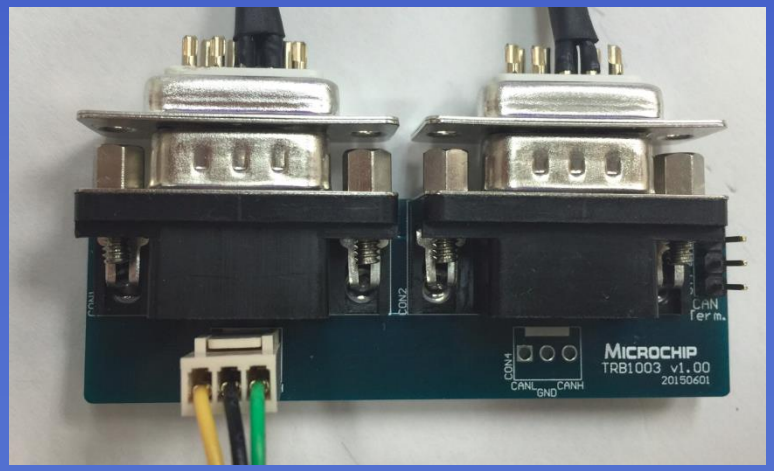
- BOOL CANIsTxReady(void) ;
  - 檢查是否有任何 TX Buffer 是可被寫入資料的
  - 傳回值 :
    - ✓ TRUE : 至少有一個 CAN Transmit Buffer 是空的
    - ✓ FALSE : 沒有任何 CAN Transmit Buffer 是空的

- 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( )) ……..

練習三：GO On the Bus

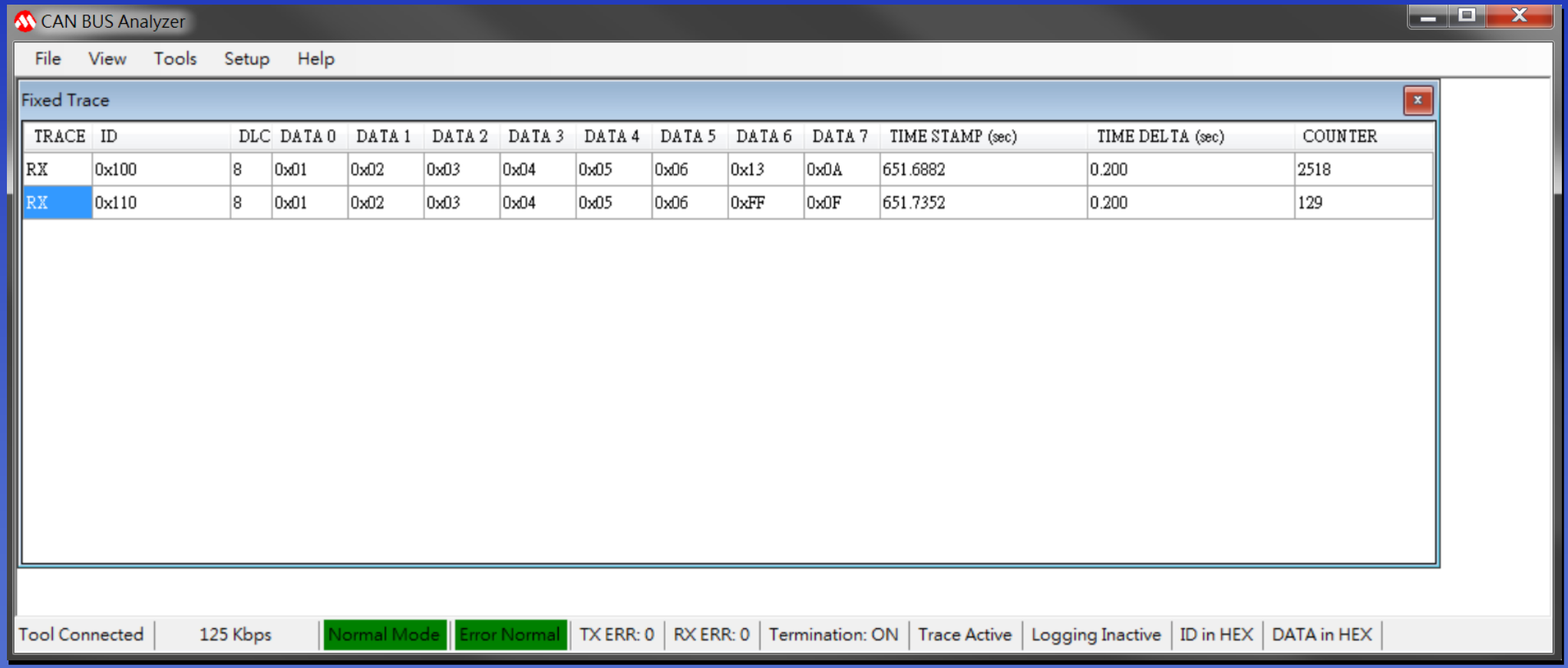
- 將 練習二 中 CANInitialization( ) 中的內容作適當的修改
  - ✓ 將參數 CAN_OP_MODE_LOOP 改成 **“CAN_OP_MODE_NORMAL”**
  - ✓ 將 MESSAGE_ID1 依照指定作修改 (0x100 .. 0x150)
  - ✓ 將 TRB1003 Molex-DB9 轉接板接至 APP001 的 CON10
  - ✓ 將 CAN cable 接至 TRB1003
  - ✓ 記得最後一個 node 要將終端電阻設定為 “ON”



練習三：GO On the Bus

- 重新 Compiler 並執行後，PIC18F8680 TXB0 的資料將以程式中定義的 MESSAGE_ID1 為 Message ID 被送出
- 因為在 NORMAL MODE 中, 故 PIC18F46K80 將不再收到自己送出的資料
- 使用 CAN Bus Analyzer (APGT002)，在講師 PC 的螢幕中將可以觀察到各個 CAN Node 傳來的資料 (每個 CAN node 的 MESSAGE_ID1 由講師指定)
  - 0x100, 0x110, 0x120, 0x130, 0x140, 0x150, ….
  - 0x200, 0x210, 0x220, 0x230, 0x240, 0x250, ….
  - 0x300, 0x310, 0x320, 0x330, 0x340, 0x350, ….

Microchip CAN Bus Analyzer
觀察到之通信內容



Receive Buffers @ PIC18F46K80 ECAN Module

PIC18F8680 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(Mode 0)

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		RXRTRRO	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)
  - JTOFF 為 RXB0DBEN 位元的複本
  - 若 RXB0DBEN = 1，則 RXB0CON<0..2> 的值為 6 or 7
  - 如此可與 RXB1CON<0..2>原本的 0..5 的值共用狀態
- FILHIT0 回傳引發此次接收的資料由哪一個 filter 過濾進來
  - 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 (Mode 0)

RXB1CON - Receive Buffer 1 Control Register

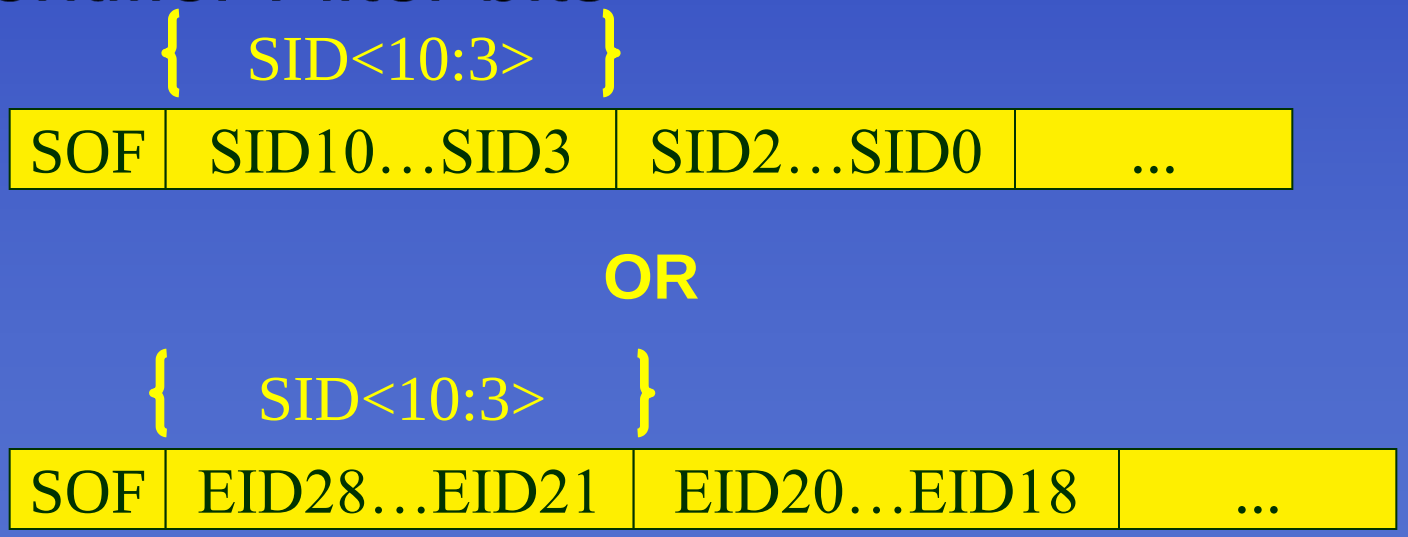
R/C-0	R/W-0	R/W-0	U-0	R-0	R-0	R-0	R-0
RXFUL	RXM1	RXM0		RXRTRRO	FILHIT2	FILHIT1	FILHIT0
bit 7							bit 0

- 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

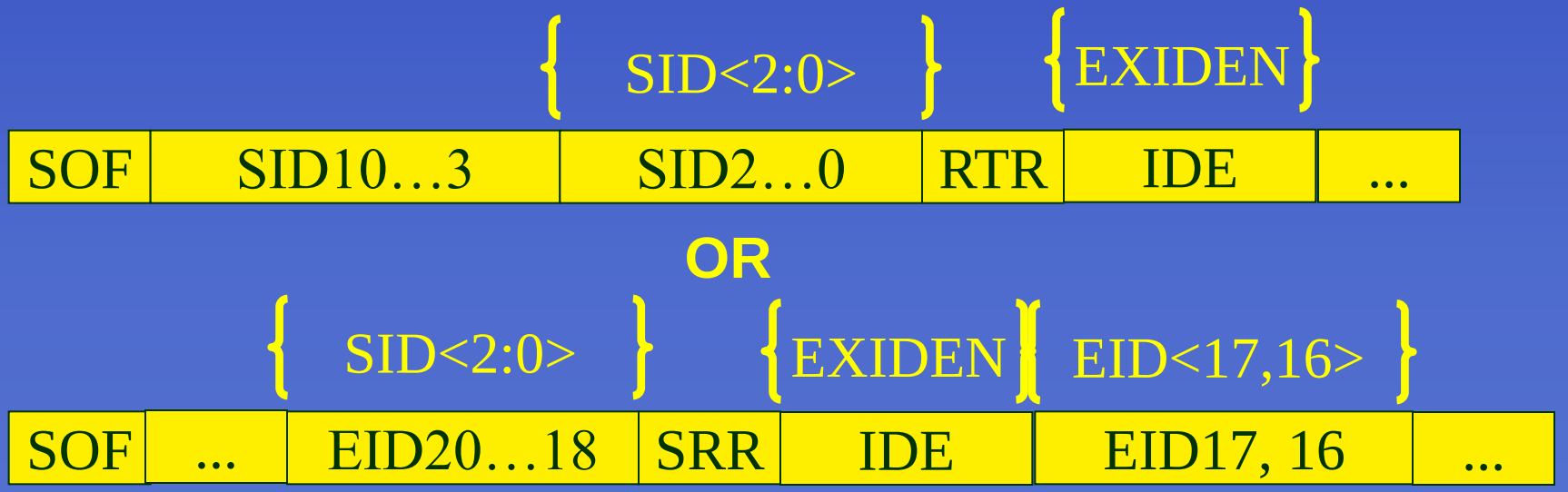


CAN Module Initialization

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



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

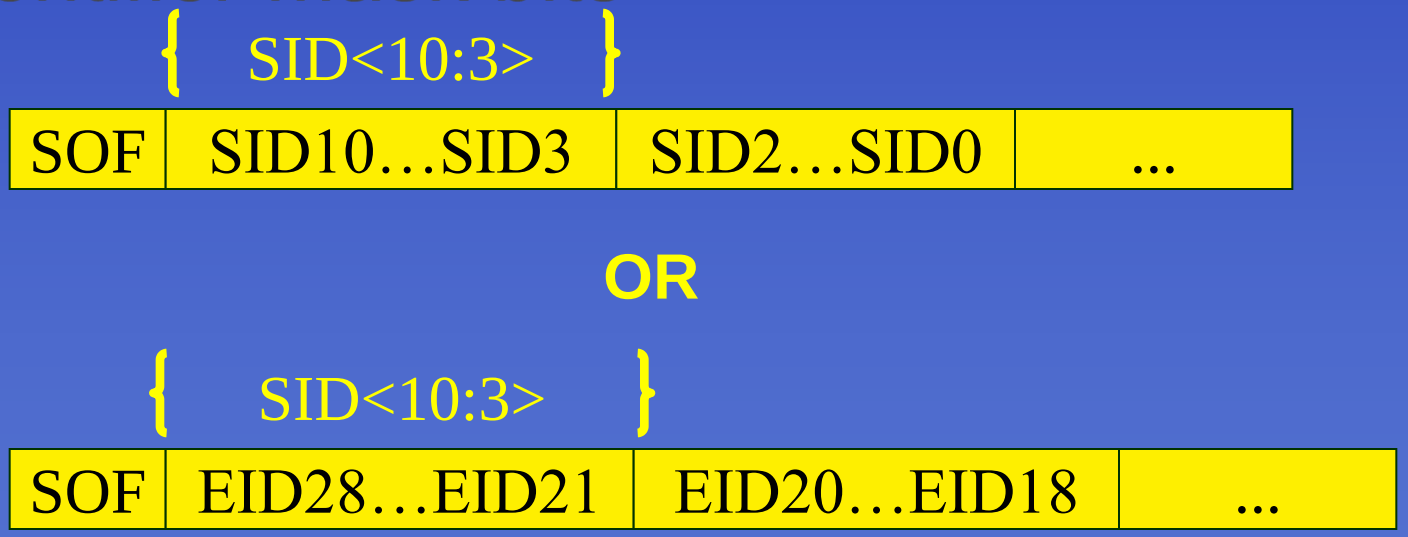
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0
bit 7							bit 0

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

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

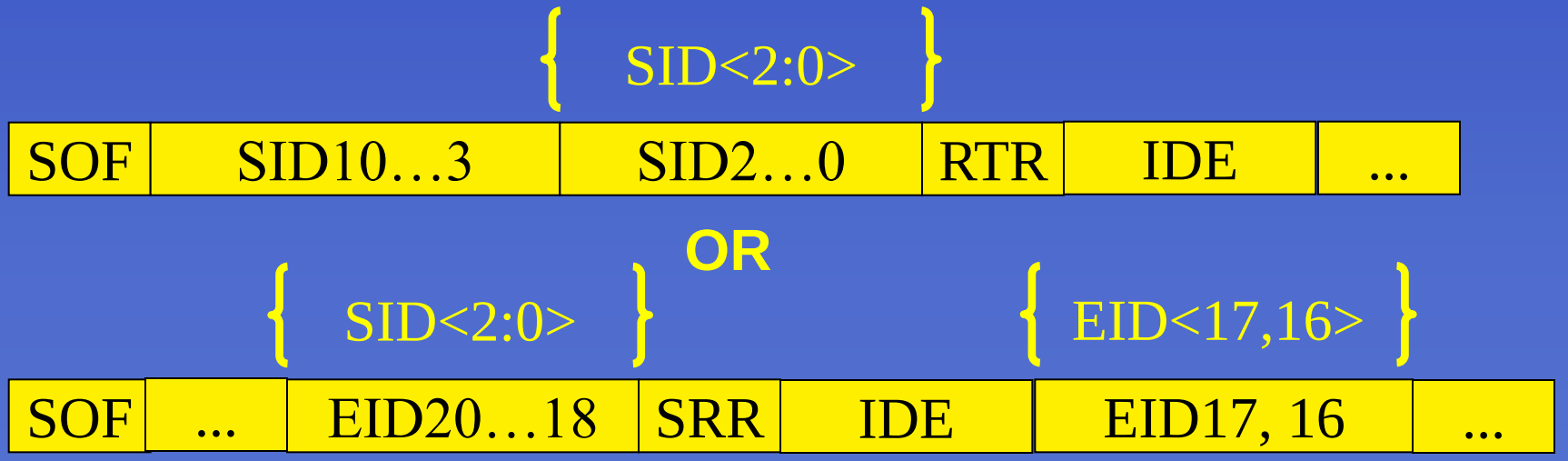


CAN Module Initialization (Mode 0)

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

- SID<2:0> defines Std./Extd. Identifier Masks, EID<17,16> defines Extd. Identifier Masks only



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

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

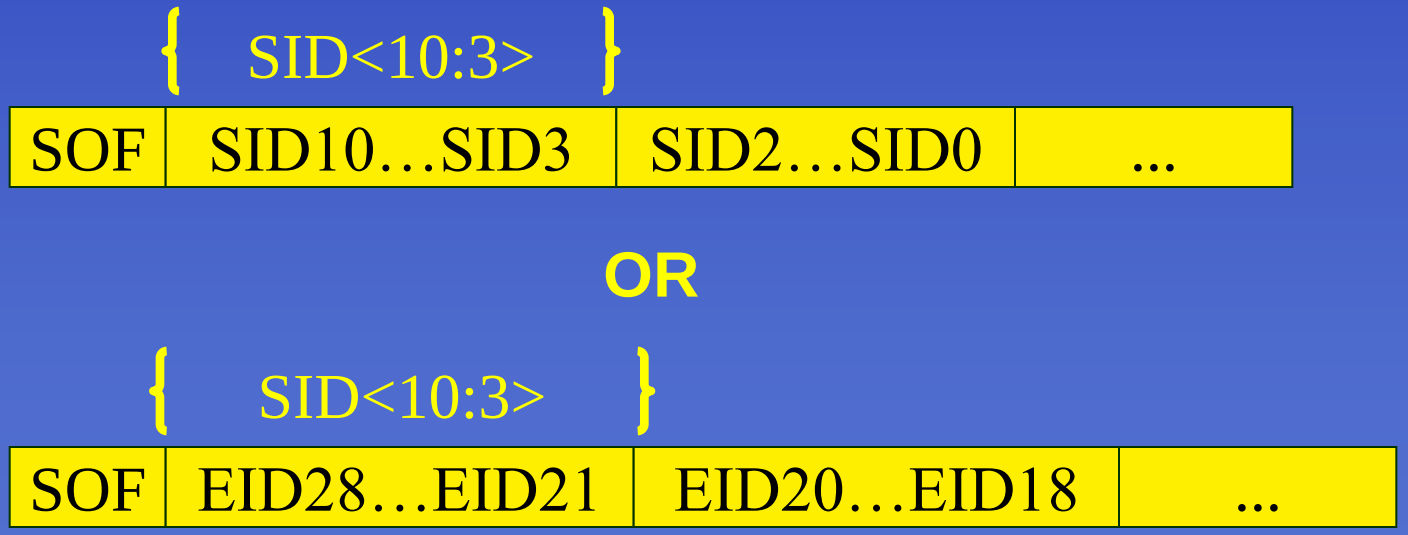
- EID<7:0> defines Extended Identifier Mask bits
  - Used for Extended Identifier Mask value only

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

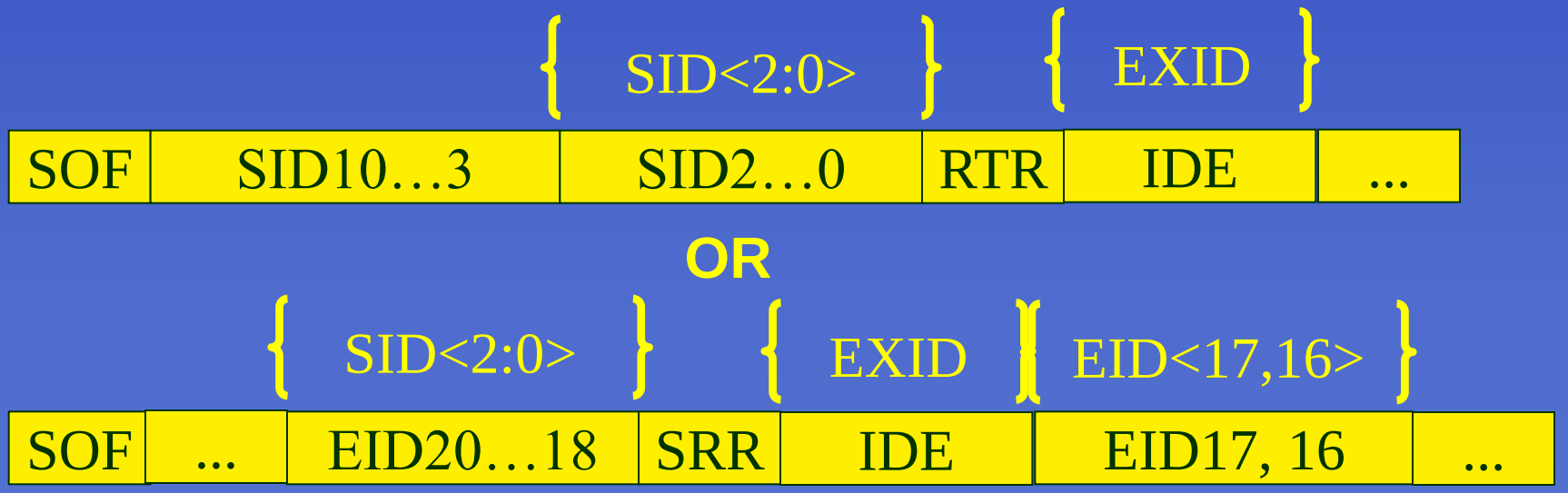


CAN Module Buffer Access

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



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 PIR5.RXBnIF = 1
- 若 PIE5.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

練習 4 : GET Data From CAN Bus

- 使用練習 3，修改 RX_FILTER0 中的內容，使其成改為 0x400
  - 也可以直接開啟 CAN_202B_EX4
- 因為主控台的 Message ID 為 0x400，故各站皆可收到主控台的資料而在 LCD Display 上有所顯示
- 若動作正確, 可將 RX_FILTER0 改成鄰站的 Message ID，然後嘗試是否可互傳 ADC 的資料

練習 5：正確的 On Bus 程序

- 在送出 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 CAN Module 設定成 NORMAL MODE 並開始設定送出資料
- 使用練習 4 的結果, 於 CANSetOperationMode( ) 中將 PIC18F46K80 設定為 LISTEN ONLY mode
  - 也可直接打開練習 5
  - 寫一個檢查 CANIsRxReady( ) 的程式迴圈, 一直到接收到 Message ID = 0x400 的資料才跳出
  - 跳出檢查迴圈後進入將 CAN module 設為 NORMAL mode
    - 在 LISTEN ONLY mode 時資料是送不出去的，此時也不要送資料，
  - 重設 PIC18F46K80 為 NORMAL mode 後，資料即可正常地收送 !!

練習 6：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
- 使用 CAN_202B_EX4 這個專案, 當 TMR1 Overflow 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

練習 6：測試 REMOTE Transfer Request

- RTR Frame 中的 ID 將分別為 0x400 .. 0x4??，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

補充教材

Microchip PIC18FXXX with ECAN

ECAN

64/80 Pin ECAN Solution

Device Definition for 80-pins

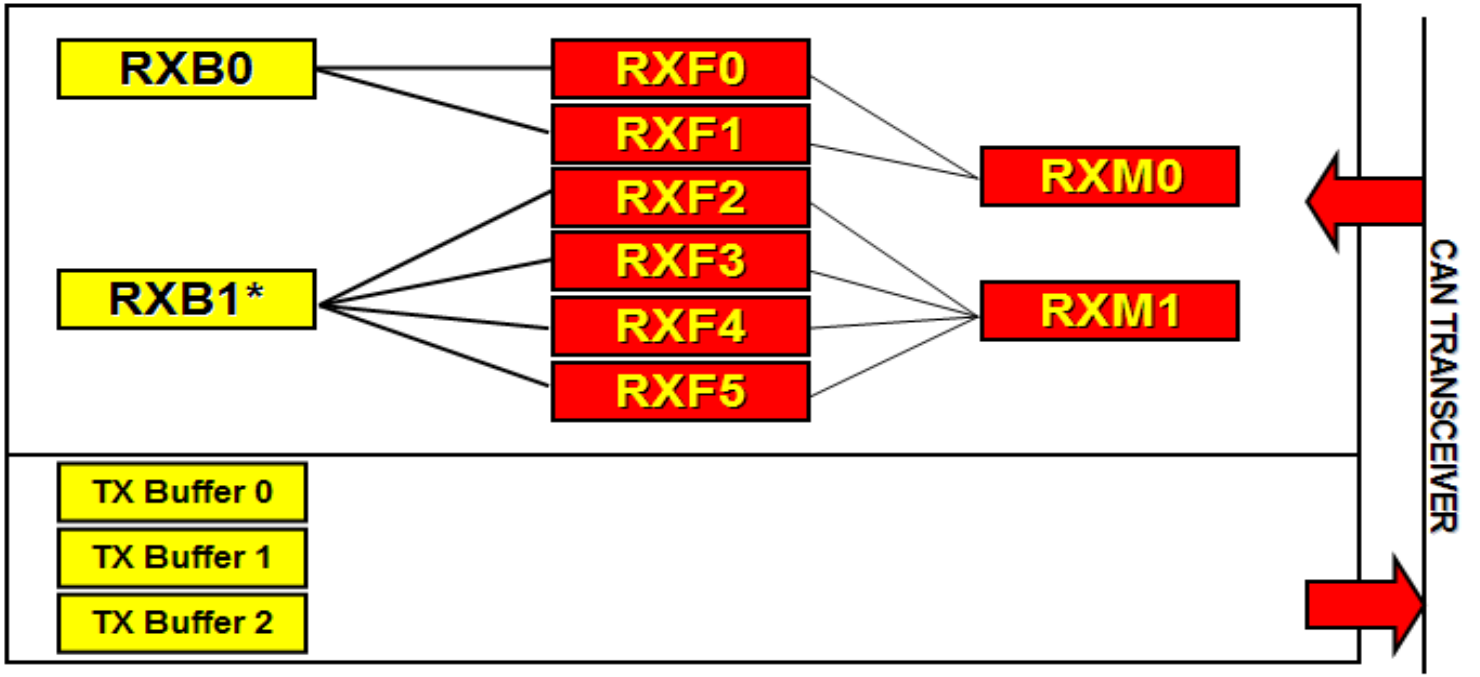
- **PIC18F8680**
  - 64K bytes Enh. Flash
  - 3Kbytes Data RAM
  - 1024 bytes Data E^2
- **PIC18F8585**
  - 48Kbytes Enh. Flash
  - 3K bytes Data RAM
  - 1024 bytes Data E^2
- Packages
  - 80 TQFP (68 I/O)
- Core
  - 16-bit instruction
  - 8x8 hardware Multiply
  - Multi-level Interrupts
  - 10MIP at 10MHz
- Peripherals
  - Analog
    - 10-Bit ADC
    - 2xComparators
  - Communication
    - AUSART (LIN)
    - SPI / I^2C
    - 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 E^2
- **PIC18F6585**
  - 48Kbytes Enh. Flash
  - 3K bytes Data RAM
  - 1024 bytes Data E^2
- 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 / I^2C
    - Parallel Slave Port
    - *ECAN*
  - Timers
    - 3x16, 1x8 Bits
    - 1xCCP & 1 xECCP

PIC18F ECAN™ Module Mode 0

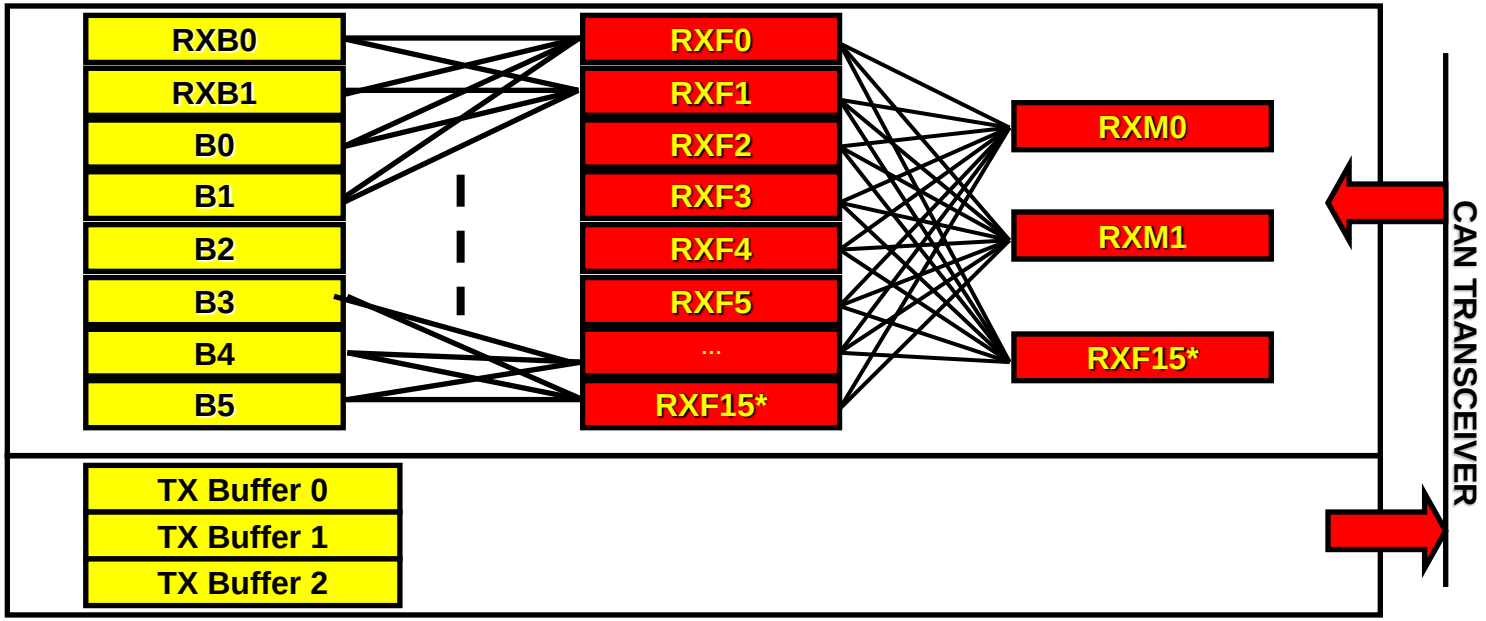
- **Mode 0 ("Legacy mode") Resources**
  - 3x TX Buffers (Dedicated)
  - 2x RX Buffers (Dedicated)
  - 1x Message Assembler Buffer
  - 6x Full Acceptance Filters
  - 2x Full Acceptance Masks



*NOTE: RXB0 can overflow into RXB1

PIC18F ECAN™ Module Mode 1

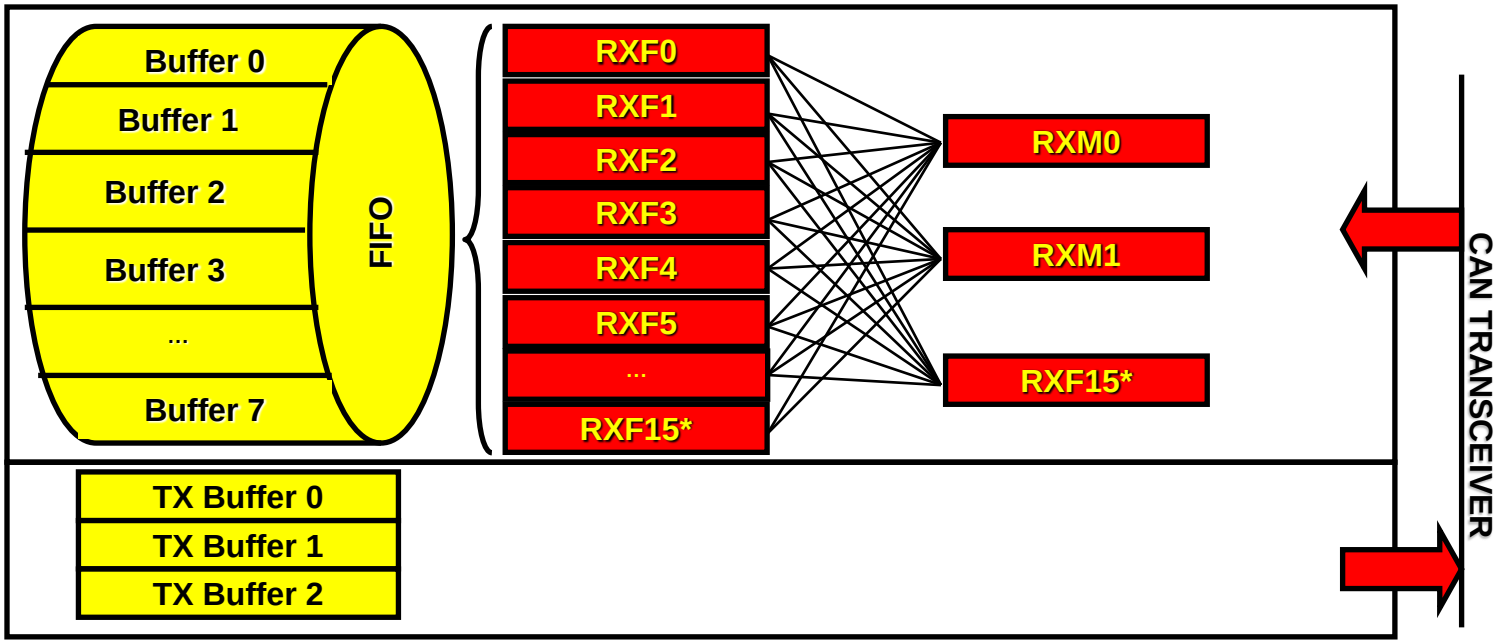
- **Mode 1: "Enhanced Legacy Mode" Resources**
  - 3x TX Buffers (Dedicated)
  - 2x RX Buffers (Dedicated)
  - 6x TX or RX buffers (Programmable)
  - 1x Message Assembler Buffer
  - 15x or 16x Full Acceptance Filters
  - 2x or 3x Full Acceptance Masks
  - Auto RTR Handling
  - Programmable data filtering on Standard Messages for DeviceNet™ Support



*NOTE: RXF15 can be used as a Mask or a Filter

ECAN™ Module Mode 2

- **Mode 2: "Enhanced FIFO Mode"**
  - RX buffers form up to a 8 level circular receive FIFO (First In First Out)
  - Filters and Masks are not assigned to individual buffers but are assigned to the FIFO
  - Mode 2 Resources match that of Mode 1 (Enhanced Legacy)



*NOTE: RXF15 can be used as a Mask or a Filter

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 E^2
- **PIC18F4585**
  - 48Kbytes Enh. Flash
  - 4K bytes Data RAM
  - 1024 bytes Data E^2
- Packages
  - 40 DIP (36 I/O)
  - 44 TQFP/QFN (36 I/O)

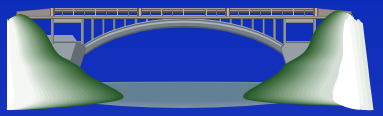
- **Core**
  - Extended Instruction Set
  - 8x8 hardware Multiply
  - 10 MIPS at 10MHz
  - nanoWatt
- **Peripherals**
  - Analog
    - 10-Bit ADC (11 ch)
    - 2xComparators
  - Communication
    - EUSART (LIN)
    - MSSP (SPI /MI^{2}C)
    - ECAN
  - Timers
    - 3x16, 2x8 Bits
    - 1xCCP & 1xECCP

Enhanced Flash Devices in 28-pins

- **PIC18F2680**
  - 64K bytes Enh. Flash
  - 4Kbytes Data RAM
  - 1024 bytes Data E^2
- **PIC18F2585**
  - 48Kbytes Enh. Flash
  - 4K bytes Data RAM
  - 1024 bytes Data E^2
- 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 /MI^{2}C)
    - ECAN
  - Timers
    - 3x16, 2x8 Bits
    - 2 x CCP

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”

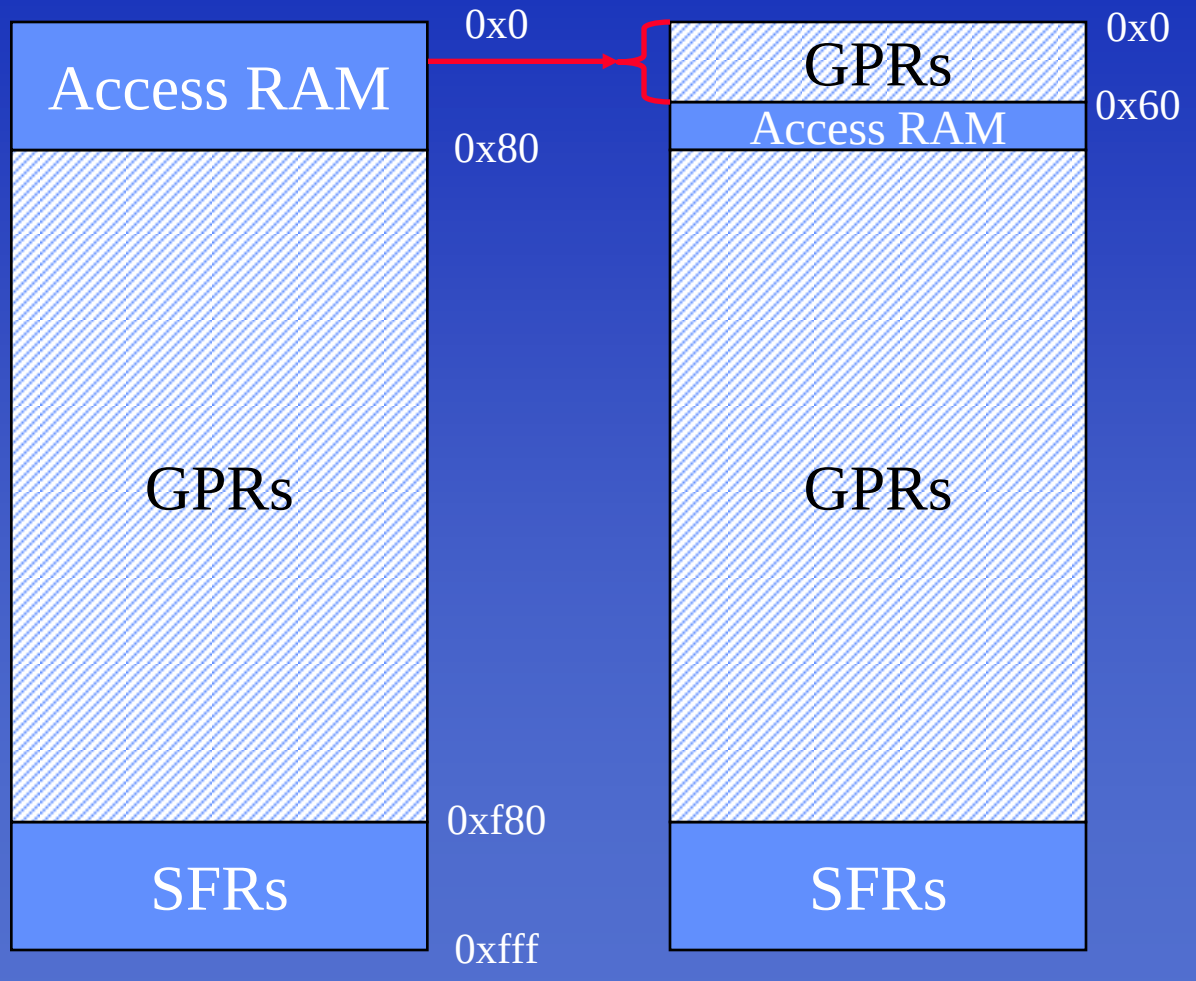
These extensions greatly simplify the code that is needed for software stack accesses.

New Indexed Addressing

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

= *banked addressing*

= *unbanked addressing*



Extended Instruction Set

- EIS consists of 8 new instruction
  - 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

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