MICROCHIP

Regional Training Centers

MCU4101T v2.0

Introduction PIC32 & MPLAB C32

Modify: *Adam Syu*
Microchip Technology Inc.

MICROCHIP
Regional Training Centers

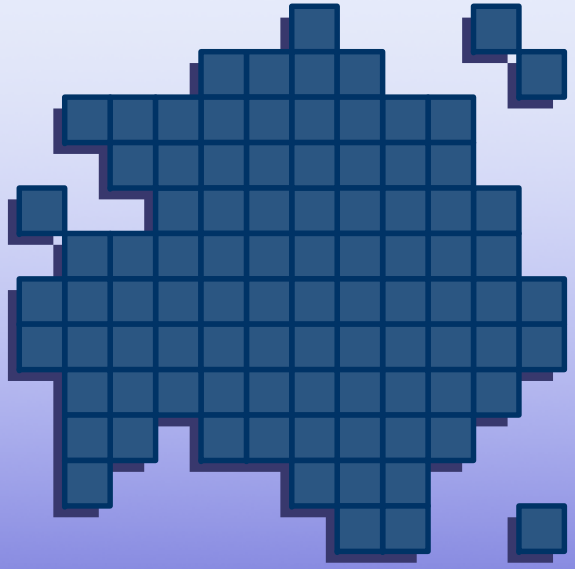
MCU4101T Abstract

- PIC32MX 32-Bits MCU架構描述。
- Development Tools簡介。
- PIC32MX Configuration Bits的設定。
- 基本IO控制與執行效能的設定。
- PIC32中斷架構及控制方式。
- Core Timer的介紹與使用。
- General Purpose Timer的介紹與使用。
- PMP Module的介紹與使用。
- ADC Module的介紹與使用。
- Serial Communication Module的介紹與使用。(Optional)

MICROCHIP
Regional Training Centers

MCU4101T Exercises

- Lab1 Basic IO : 基本的IO控制與System Performance。
- Lab2 Timer : Core Timer與G. P. Timers的控制。
- Lab3 PMP : Parallel Master Port的控制。
- Lab4 ADC Single : ADC單通道取樣的控制。
- Lab5 ADC Scan : ADC多通道掃描的控制。
- Lab6 Timer Trigger ADC : ADC Timer觸發及ISR的控制。
- Lab7 UART Rx Interrupt : UART控制的Demo



MIPS and PIC32 Introduction

MICROCHIP
Regional Training Centers

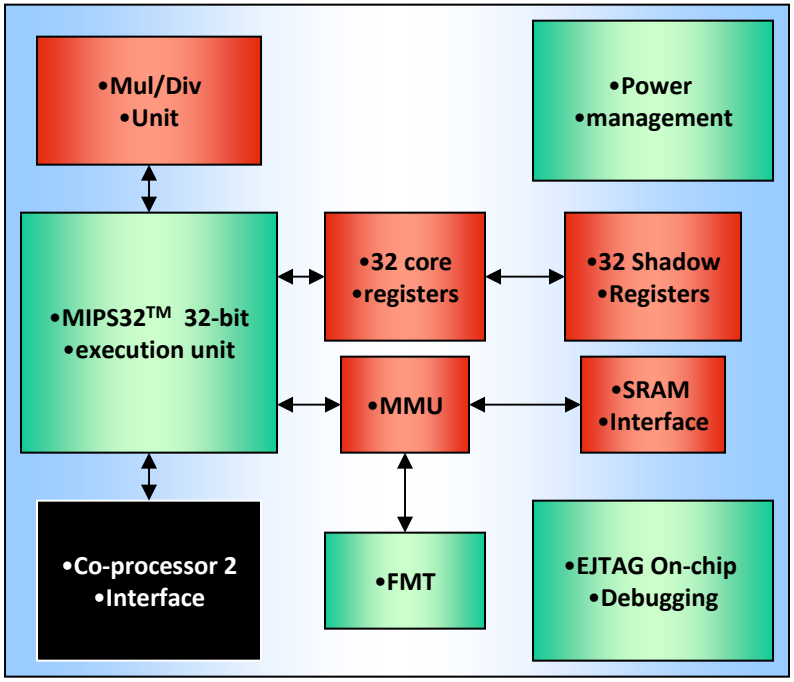
MIPS32® M4K® Core

and Microchip Enhancements

- The MIPS Engine
  - Up to 80 MHz
  - 1.56 DMIPS/MHz measured
  - 32 32-Bit Core Registers with Shadow set
  - 32-Bit ALU, Single Cycle MAC
- Microchip System Enhancements Embedded Performance
  - 256 Byte Pre-fetch Cache
  - Low-latency Vectored Interrupt Control
  - Bus Matrix for Parallelism
- Debugging
  - Hardware Trace with Debug Support
  - JTAG and Boundary Scan

MIPS TECHNOLOGIES

•MIPS32® M4K® Core



MICROCHIP
Regional Training Centers

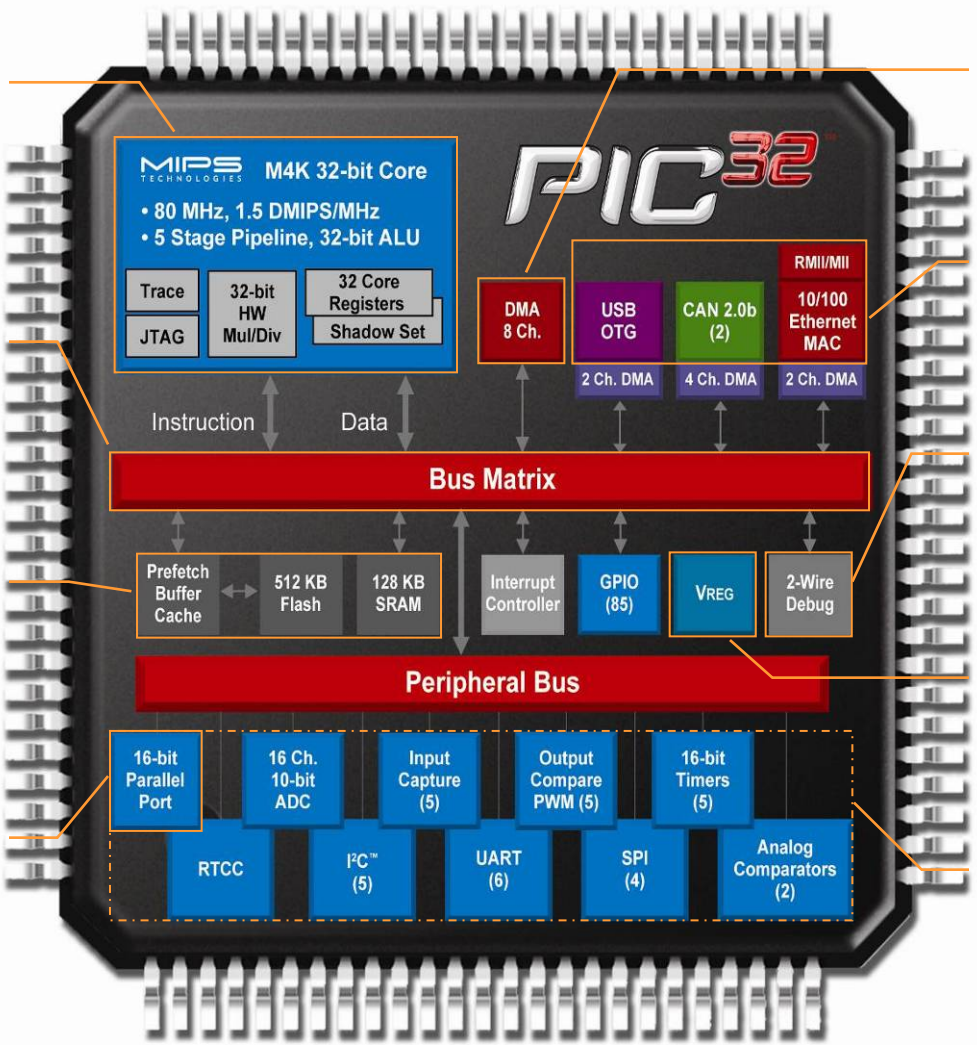
PIC32's Key Features

32-bit MIPS M4K® Core, Harvard Architecture, Single Cycle Hardware MAC, Fast Interrupts & Context Switch.

High Throughput Bus Matrix, which Supports High Speed Concurrent Access to Memories and Peripherals.

512KB 128-bit wide Self-programmable Flash, Predictive Instruction Pre-fetch, 256 Byte Lockable Cache128K RAM.

16-bit Parallel Master Port, Connect SRAM, Flash, QVGA LCDs or other Peripherals.



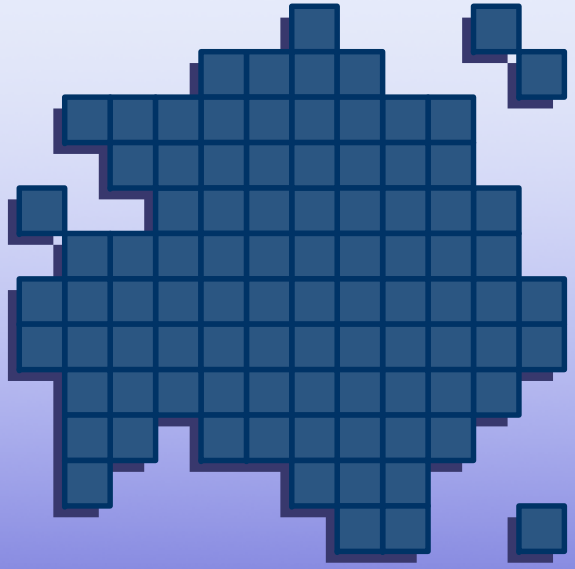
Integrated Connectivity Peripherals for fast cost effective operation:10/100 Ethernet, 2x CAN, USB OTG.

Direct Memory Access Controller, With Integrated CRC, Module Operates in Idle Mode.

Compatible with Microchip Development Tools MPLAB® ICD 3, MPLAB REAL ICE™, PICkit™ 3, PM3

Single 2.3 to 3.6V Supply Power On Reset, Brown Out Reset, Low Voltage Detection

Rich Integrated Analog and Digital peripheral set, Compatible with 16-bit PIC® Microcontrollers



PIC32 Memory Architecture

MICROCHIP
Regional Training Centers

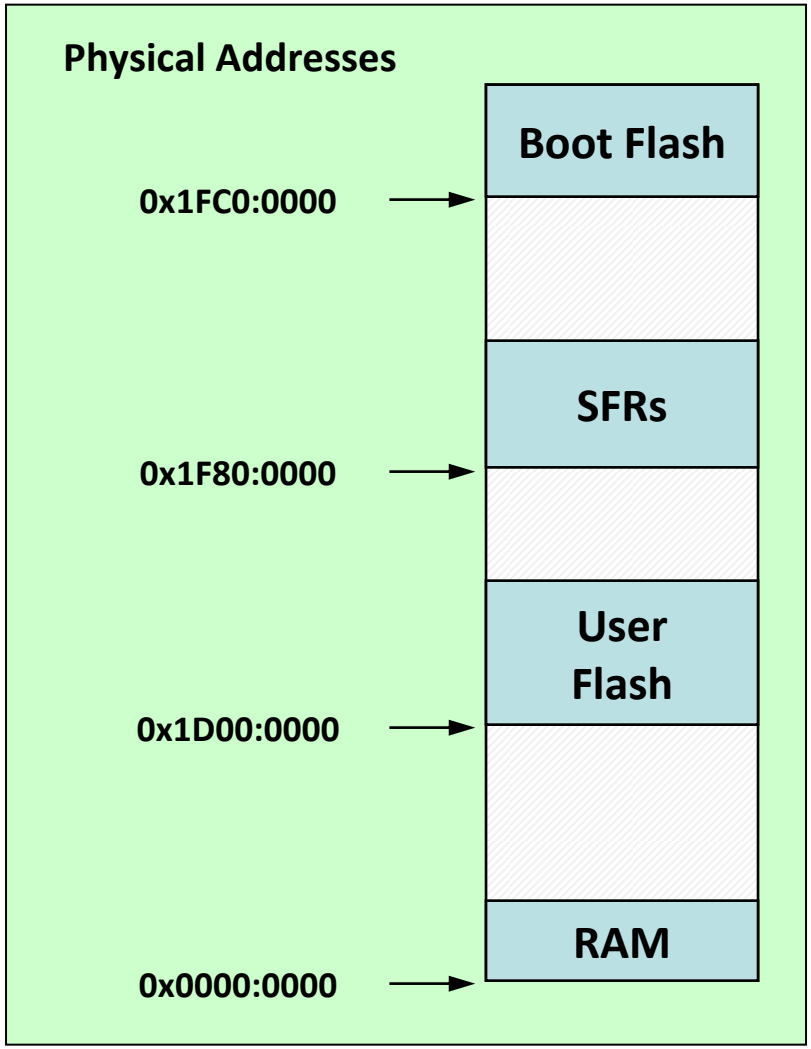
PIC32's Memory

- PIC32與八及十六位元的記憶體架構截然不同。PIC32導入虛擬記憶體架構,將所以有的實體記憶體區塊都映射成一大塊的連續記憶體區塊。
- 以CPU角度在看記憶體時,不管是程式記憶體,資料記憶體,特殊功能暫存器等等,對CPU來看只是位址的不同,存取方式都相同。
- 除了CPU以虛擬記憶體的角度存取資料外,其他的裝置如DMA等,仍以實體記憶體的角度存取資料。

MICROCHIP
Regional Training Centers

PIC32's Physical Memory

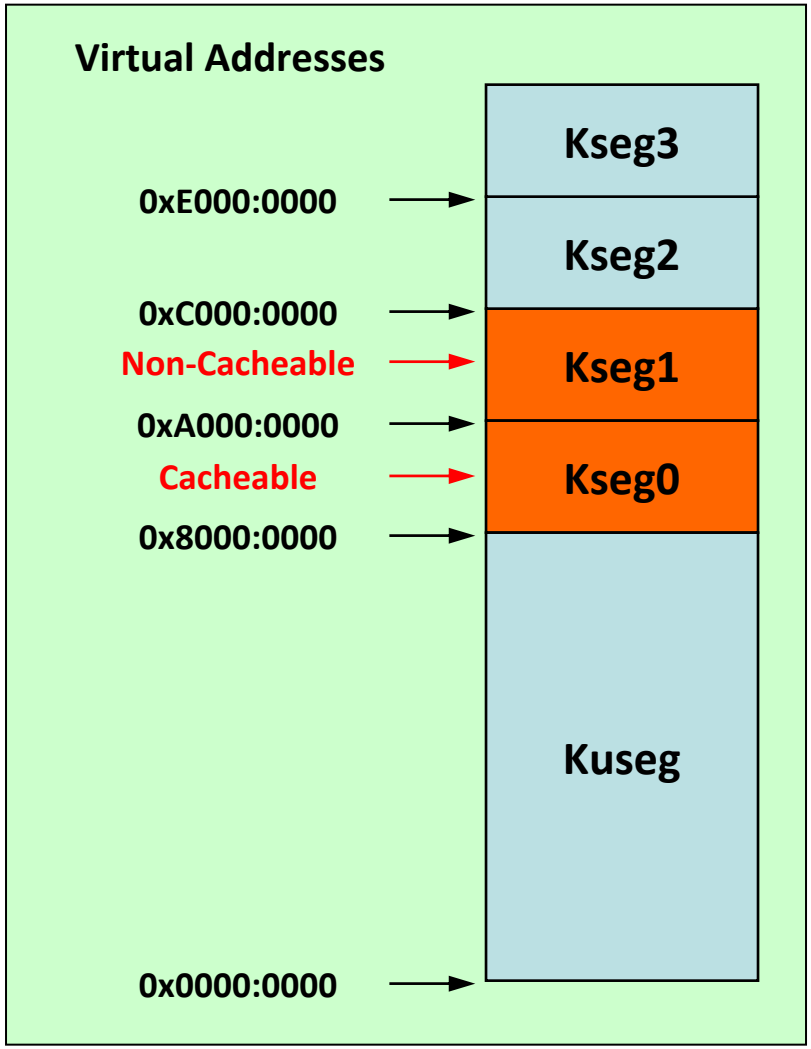
- PIC32的實體記憶體配置如圖所示。所有型號都具有12K Bytes的 Boot Flash。
- 根據不同型號分別擁有32K ~ 512K Bytes 不等的 User Flash 及 8K ~ 128K Bytes 不等的 RAM 。
- 最大定址空間可達4G (2^{32}) Bytes, 實際使用範圍目前為 512M Bytes 。
- 周邊的暫存器採用 Memory Mapping IO的方式。



MICROCHIP
Regional Training Centers

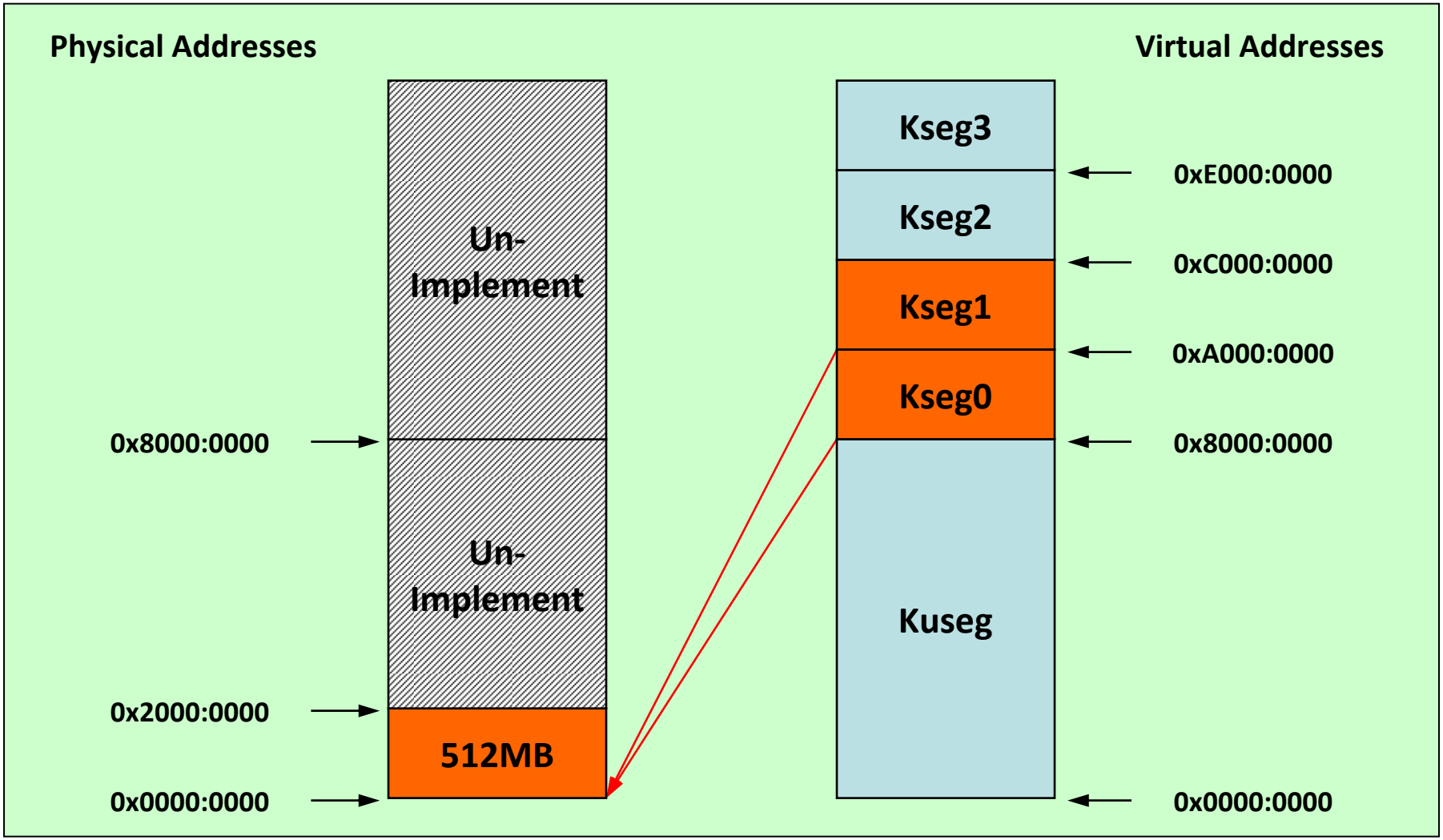
PIC32's Virtual Memory

- Kseg3:JTAG 介面使用。
- Kseg2:未使用區域。
- Kseg1 :核心(Kernel)模式可存取區域。無提供快取機制(Non-Cacheable)。
- Kseg0:核心(Kernel)模式可存取區域。提供快取機制(Cacheable)。
- Kuseg:核心(Kernel)與使用者(User)模式可存取區域。
- 最大定址空間為4G Bytes，目前僅Kseg0,1可使用。



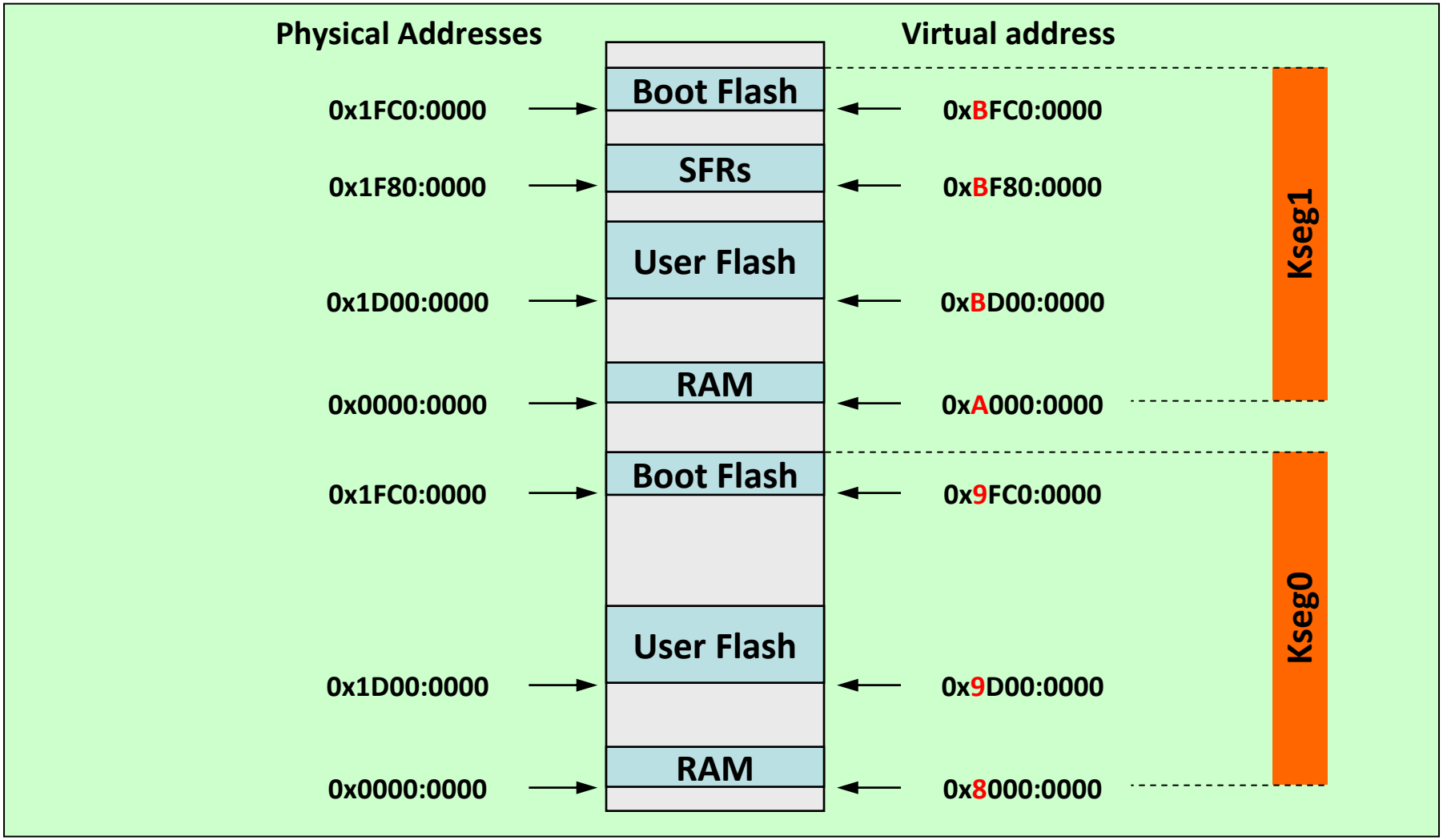
MICROCHIP
Regional Training Centers

Fixed Mapping Translation



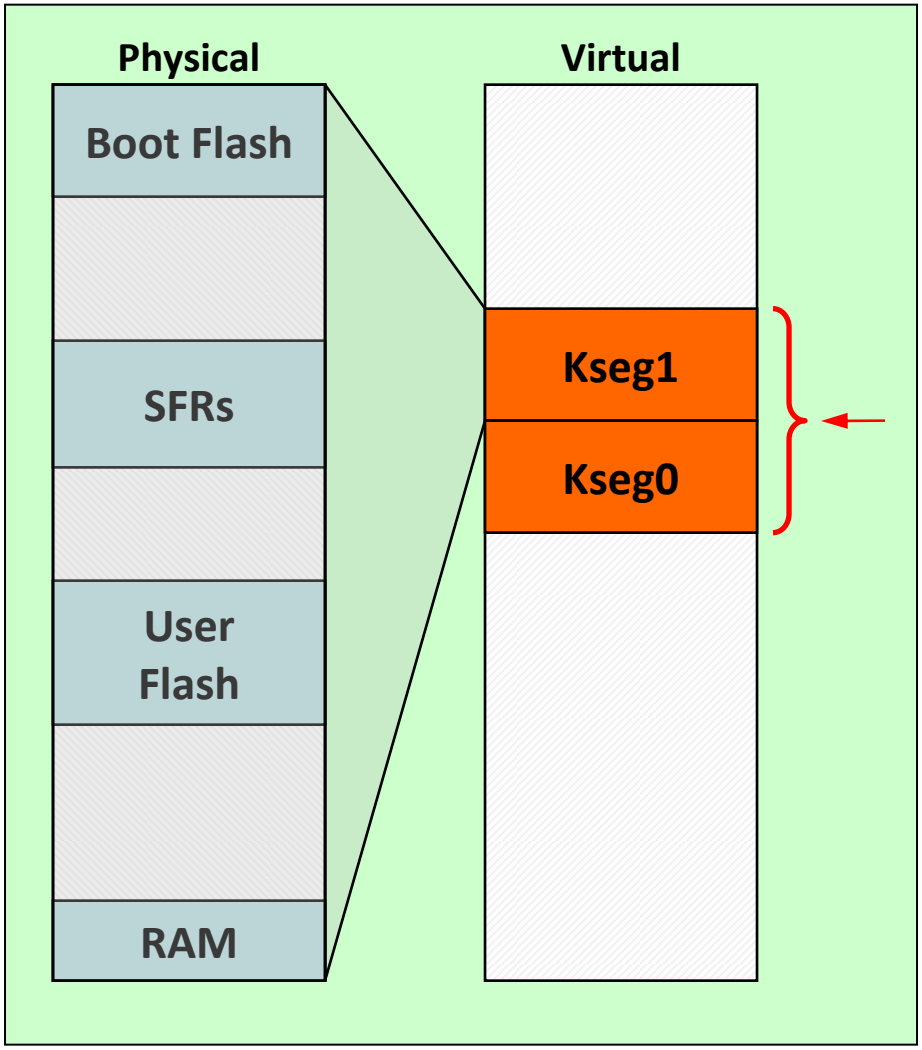
MICROCHIP
Regional Training Centers

Kseg0, Kseg1 Memory Mapping



MICROCHIP
Regional Training Centers

Memory Pointer

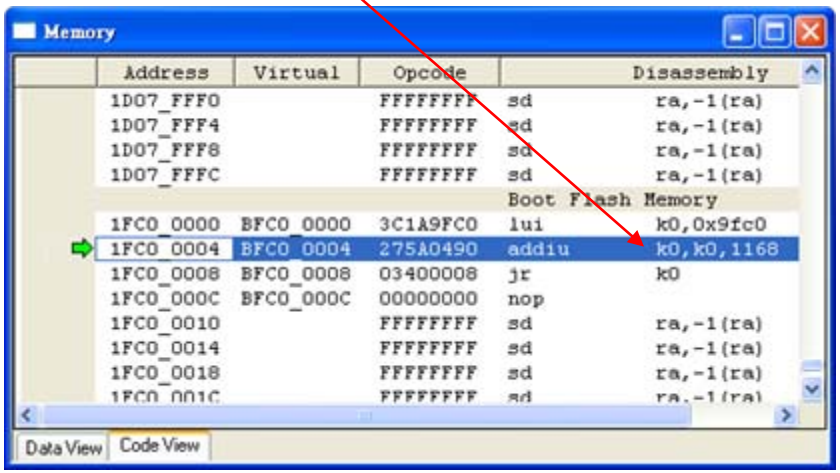


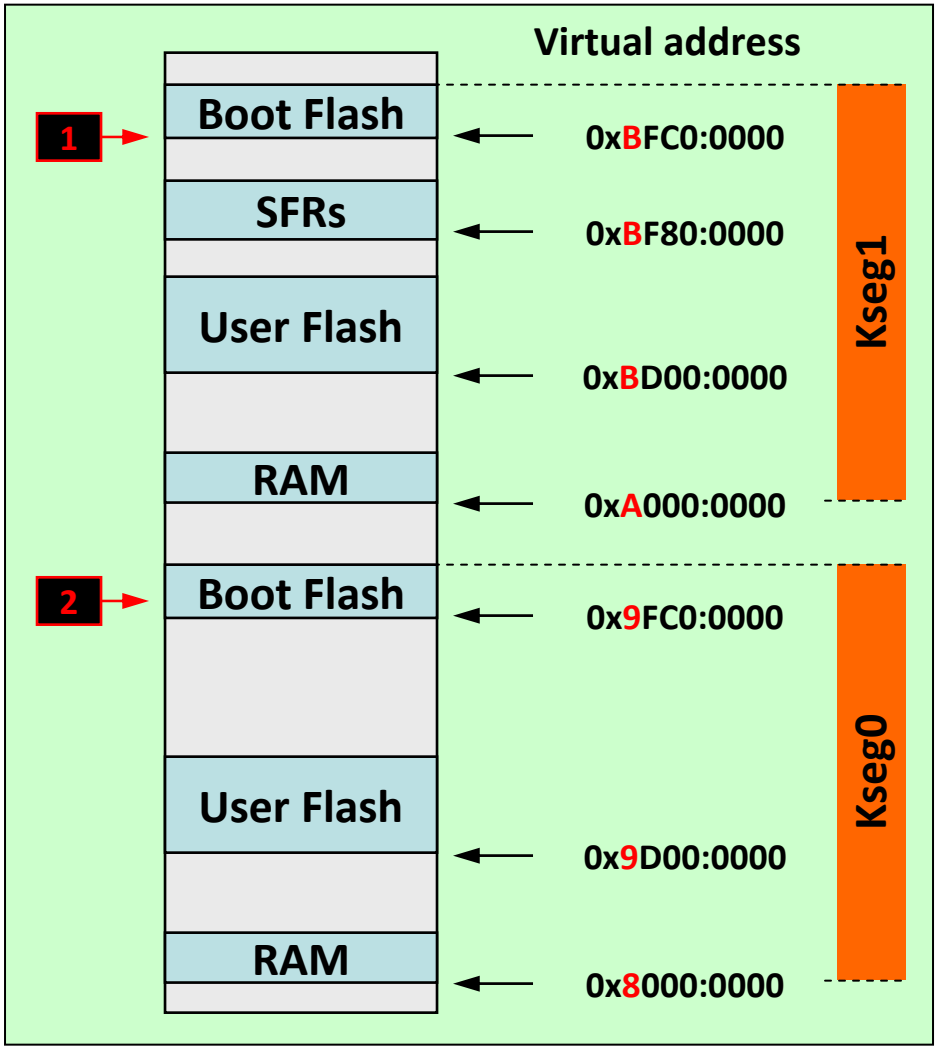
- 記憶體被視為線性空間，採用資料記憶體形式的指標即可定址全部空間。
  For Example:
  unsigned char *Pointer1;
  long *Pointer2;
- 不再需要針對指標做 rom 或 psv 的宣告。
- MPLAB C32 指標長度為32-Bit,可定址空間為4G Bytes。目前只能存取到Kseg1,Kseg0。

MBLAB C32's Boot flow-1

1.Reset 後 PC 跳到 Kseg1 的
Boot Flash (0xBFC0_0000) 執行

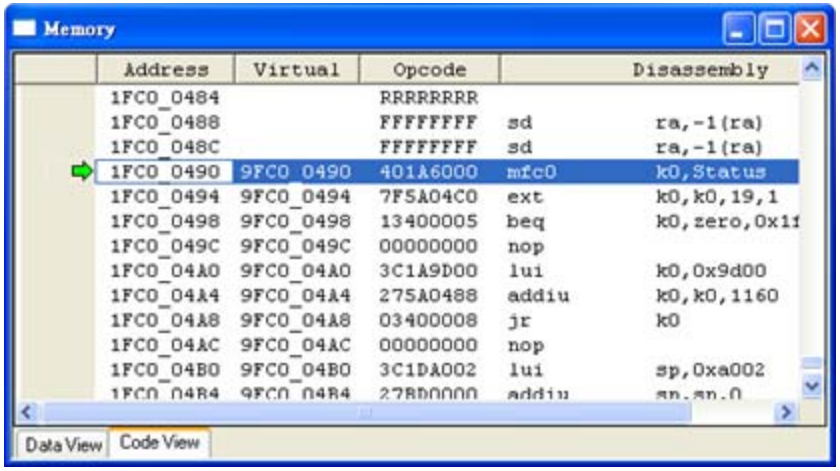
2.Kseg1 切換到 Kseg0 的
Boot Flash (0x9FC0_0490) 執行
@PC = 0x9FC0:0000 + offset
(1168 = 0x490)

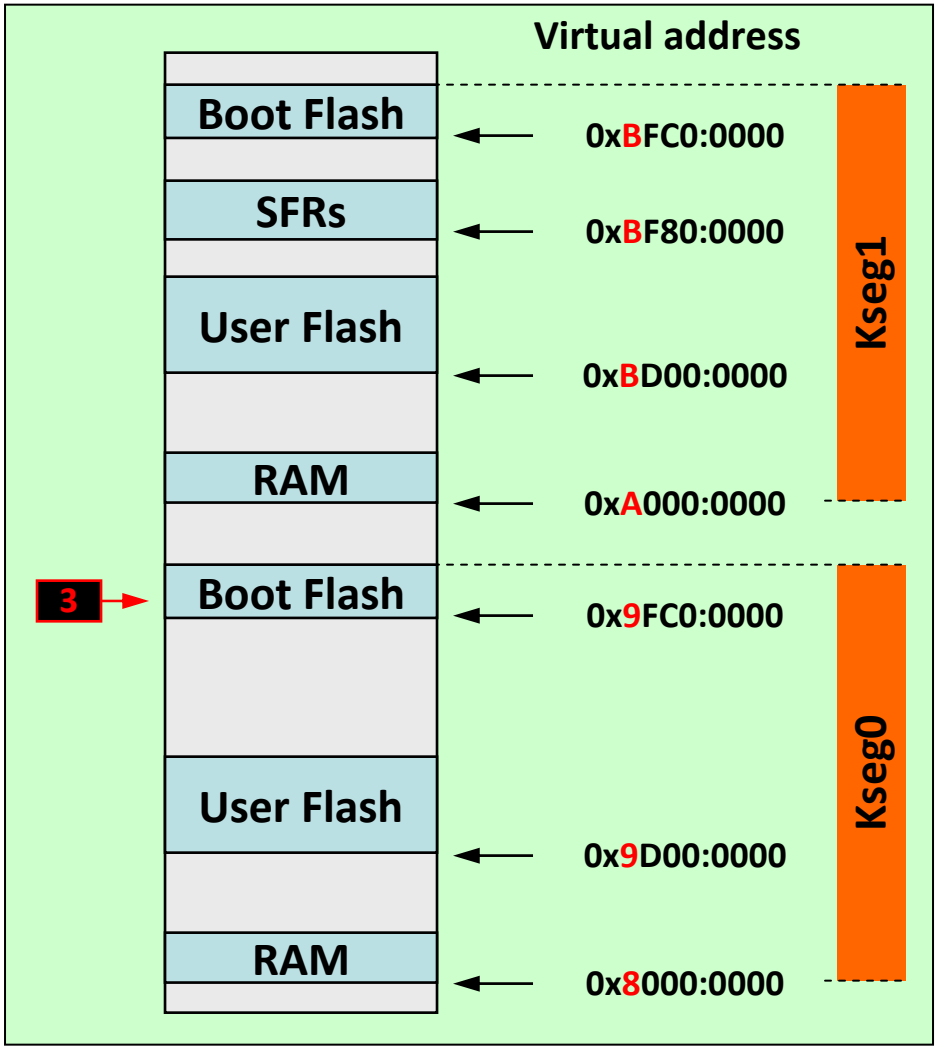




MBLAB C32's Boot flow-2

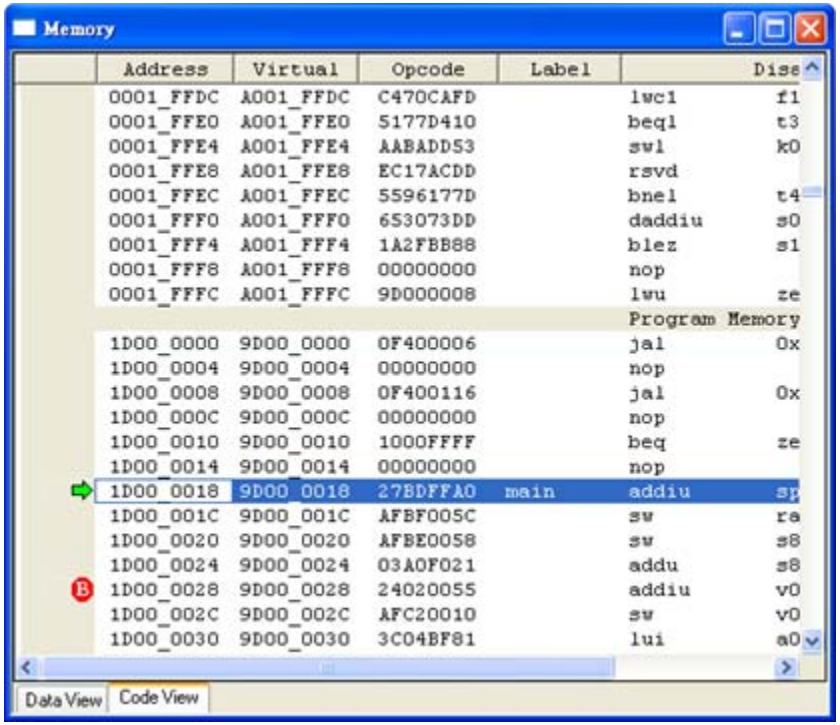
3.程式在 KSEG0 的 Boot Flash
區塊開始執行 C32 的初始設定。

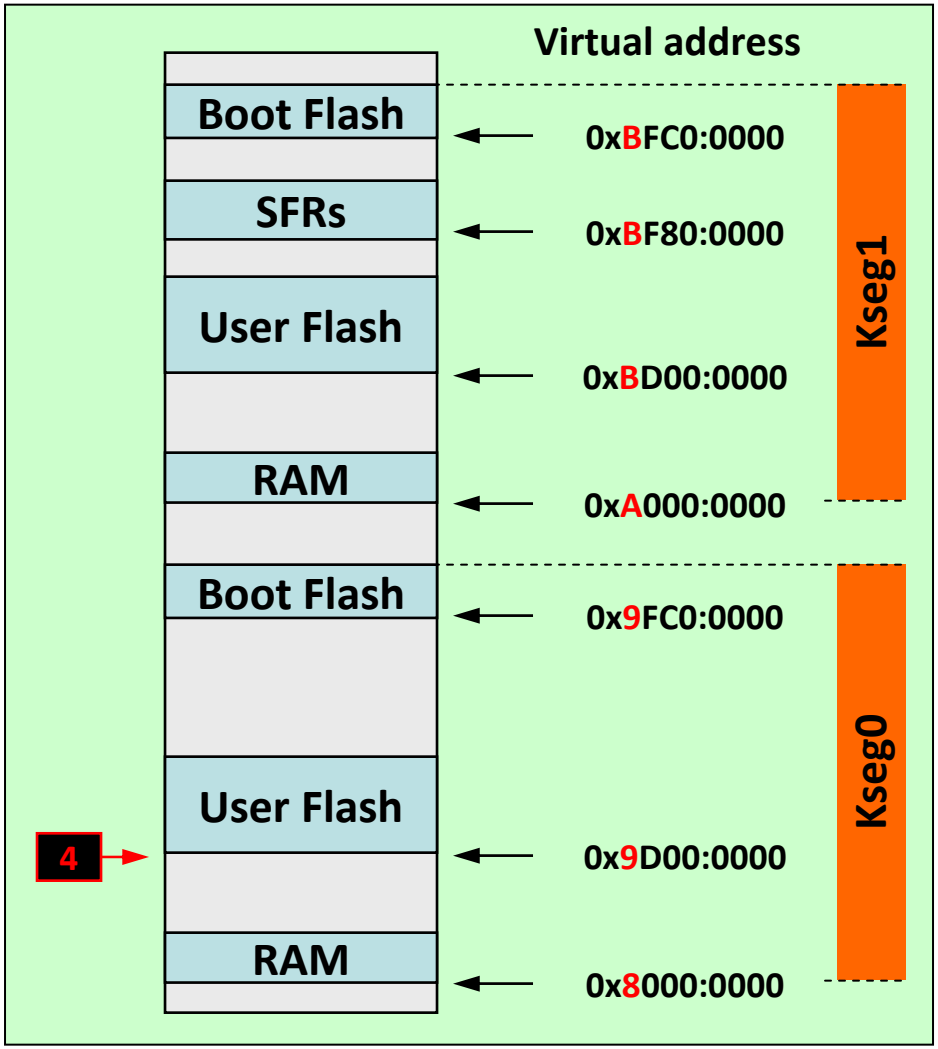




MBLAB C32's Boot flow-3

4.程式再跳到 KSEG0的
Program Flash (0x9D00_0000)
執行然後進入 main( ) 。



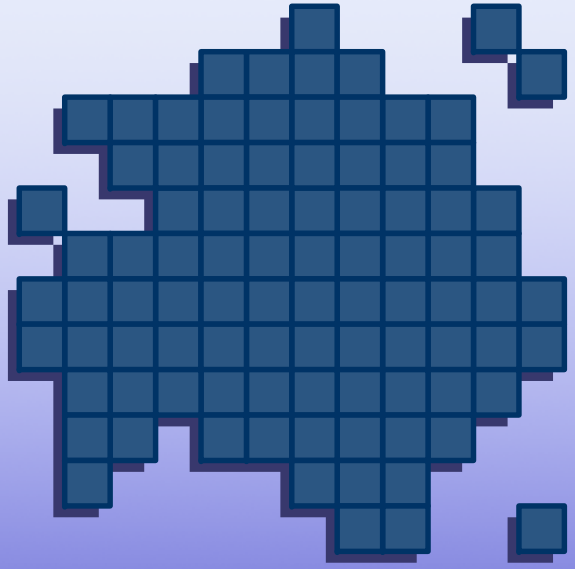


MICROCHIP
Regional Training Centers

MPLAB C32's Linker Script

- MPLAB C32透過Linker Script(*.ld)來描述每個MCU的記憶體大小, 起始位址與長度等等資訊。每個PIC32 MCU都有專屬的Linker Script (procdefs.ld) 。
- 一般情形下是不需要更動Linker Script的, 除非有特殊應用,必須更改程式碼或資料的配置, 例如:Bootloader的應用。

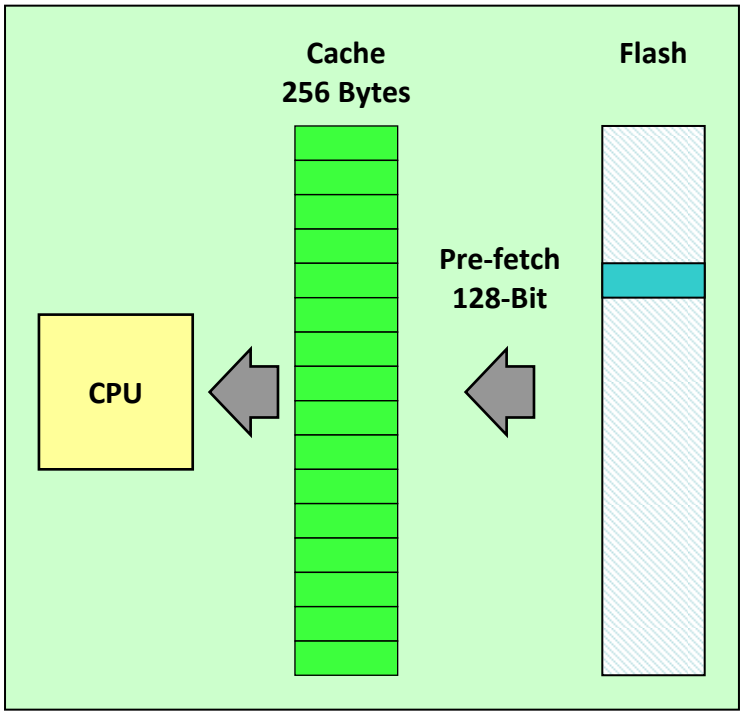
```
_RESET_ADDR = 0xBFC00000;
_BEV_EXCPT_ADDR = 0xBFC00380;
_DBG_EXCPT_ADDR = 0xBFC00480;
_DBG_CODE_ADDR = 0xBFC02000;
_DBG_CODE_SIZE = 0xFF0    ;
_GEN_EXCPT_ADDR = _ebase_address + 0x180;

MEMORY
{
  kseg0_program_mem (rx)  : ORIGIN = 0x9D000000, LENGTH = 0x80000
  kseg0_boot_mem : ORIGIN = 0x9FC00490, LENGTH = 0x970
  exception_mem : ORIGIN = 0x9FC01000, LENGTH = 0x1000
  kseg1_boot_mem : ORIGIN = 0xBFC00000, LENGTH = 0x490
  debug_exec_mem : ORIGIN = 0xBFC02000, LENGTH = 0xFF0
  kseg1_data_mem (w!x) : ORIGIN = 0xA0000000, LENGTH = 0x20000
...
}
```



PIC32 Cache and Pre-fetch

Instruction Prefetch and Cache

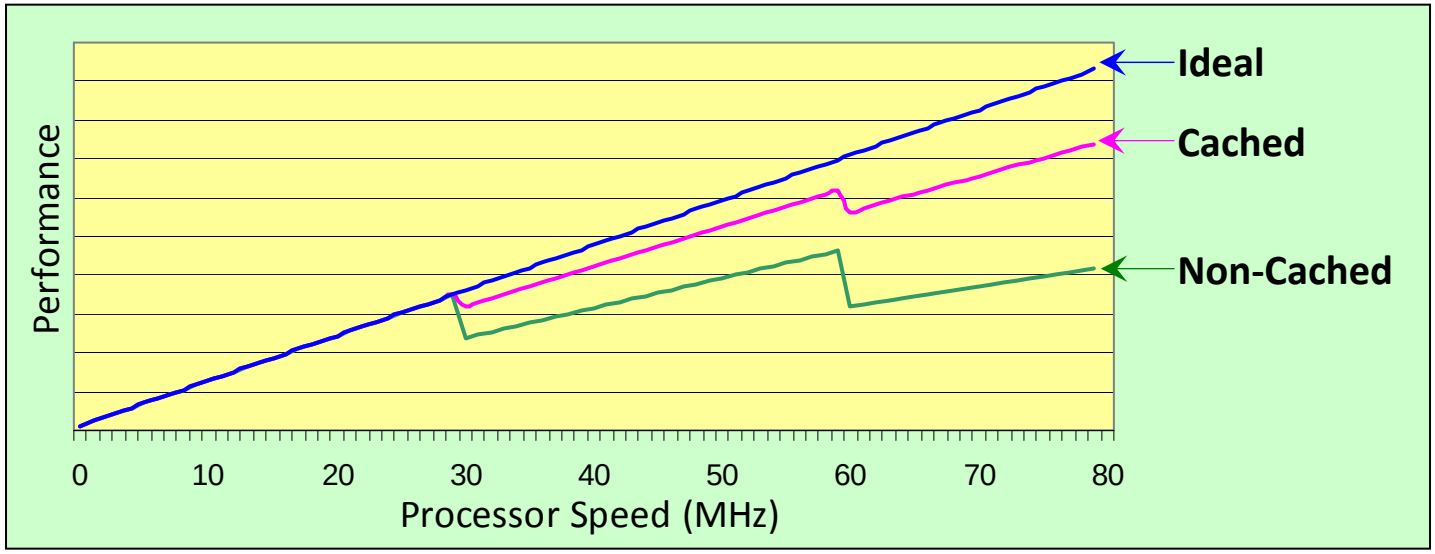
- PIC32具有快取及預提取機制,可以縮短指令擷取所花費的時間,提高效率。
- 預提取(Pre-fetch)機制可同時提取128-Bit資料,可解譯為4個32-Bit指令或8個16-Bit指令。
- PIC32的快取(Cache)結構為十六線(Line),全相關聯式,可同時取用256 Byte的資料。
- PIC32的快取機制預設是關閉的,Wait State則是開到最大。

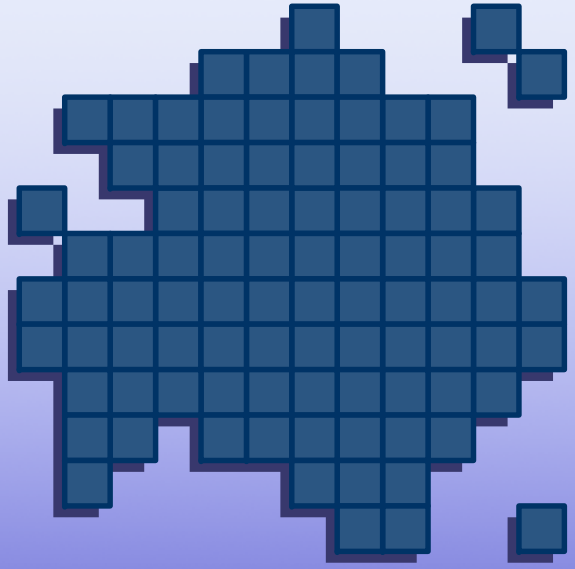


MICROCHIP
Regional Training Centers

Effect of Prefetch Cache

- 理想中的執行效率應與頻率成正比,頻率越快,效率越高。
- 實際上由於Flash的存取速度無法跟上CPU,因此必須加上Wait State來克服,導致效率受限。
- 利用快取機制可以作為Flash與CPU的中介,降低Wait State造成效率受限的問題,提高整體效率。



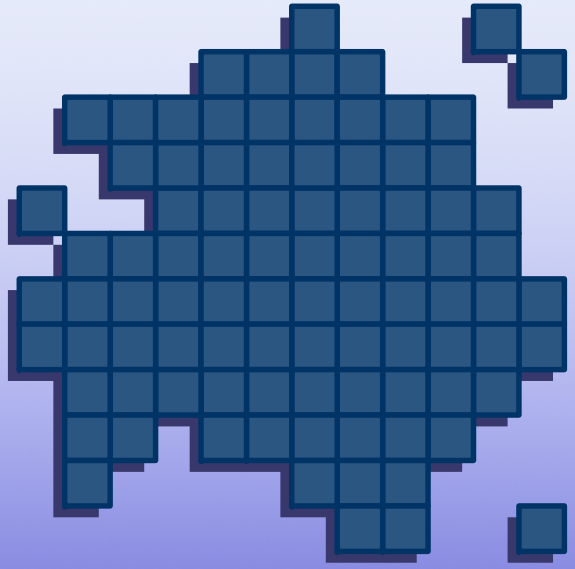


PIC32 Co-processer

MICROCHIP
Regional Training Centers

PIC32's Co-Processer 0

- MIPS具有協同處理器機制,所有MIPS都含有CP0(Co- Processer 0)。
- CP0主要功能用於MIPS的組態與狀態設定,MIPS快取的控制,虛擬記憶體的映射,影子(Shadow)暫存器的管理,例外及中斷控制,除錯控制等。另外還提供一個32位元的核心計時器Core Timer,供系統內部計時使用。
- 由於CP0的許多功能都與MIPS的控制息息相關, 不可隨意更動。因此CP0內部的暫存器區分為兩類。
- 一類為控制(Control)暫存器,此類暫存器必須透過CPU內部暫存器才能存取。
- 另一類則稱為通用(General)暫存器,此類暫存器可以一般記憶體存取,不強制必須透過CPU內部暫存器存取。

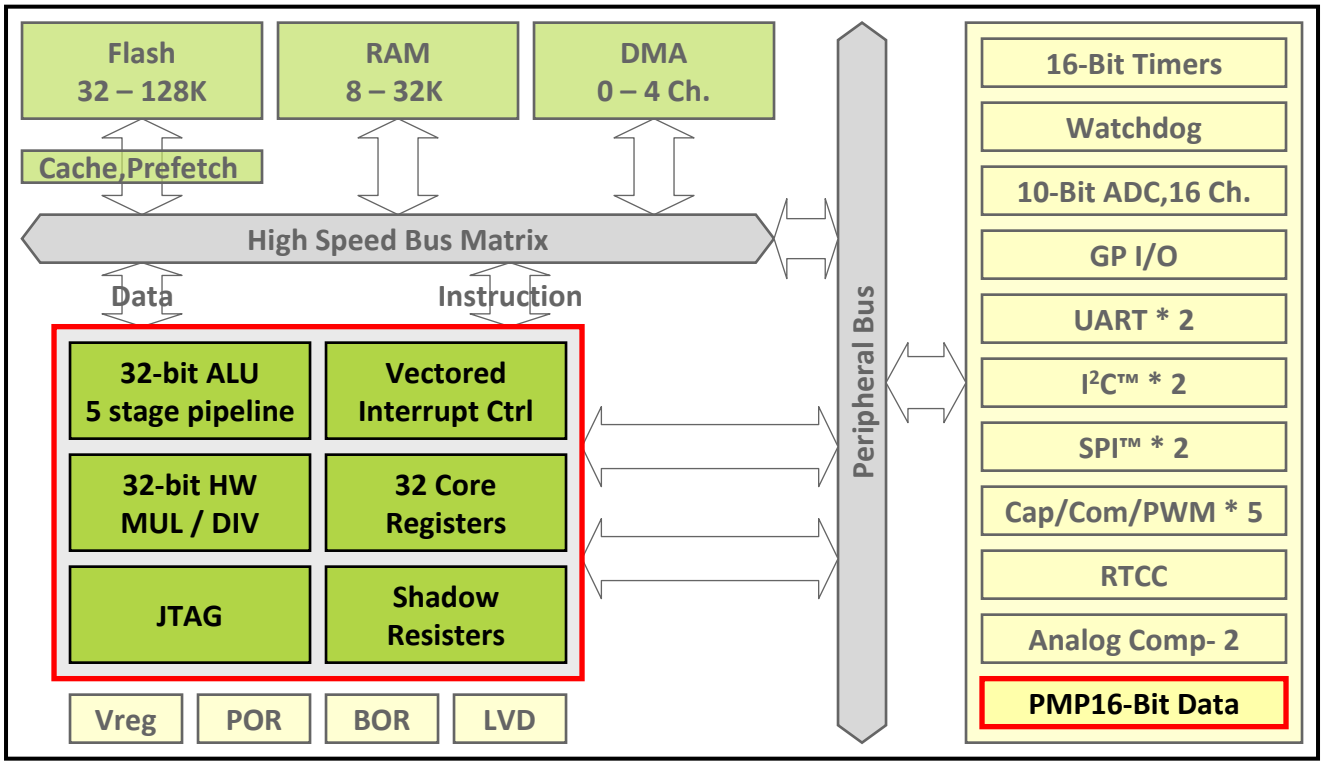


PIC32 Series Introduction

MICROCHIP
Regional Training Centers

PIC32MX3 Series

General Purpose

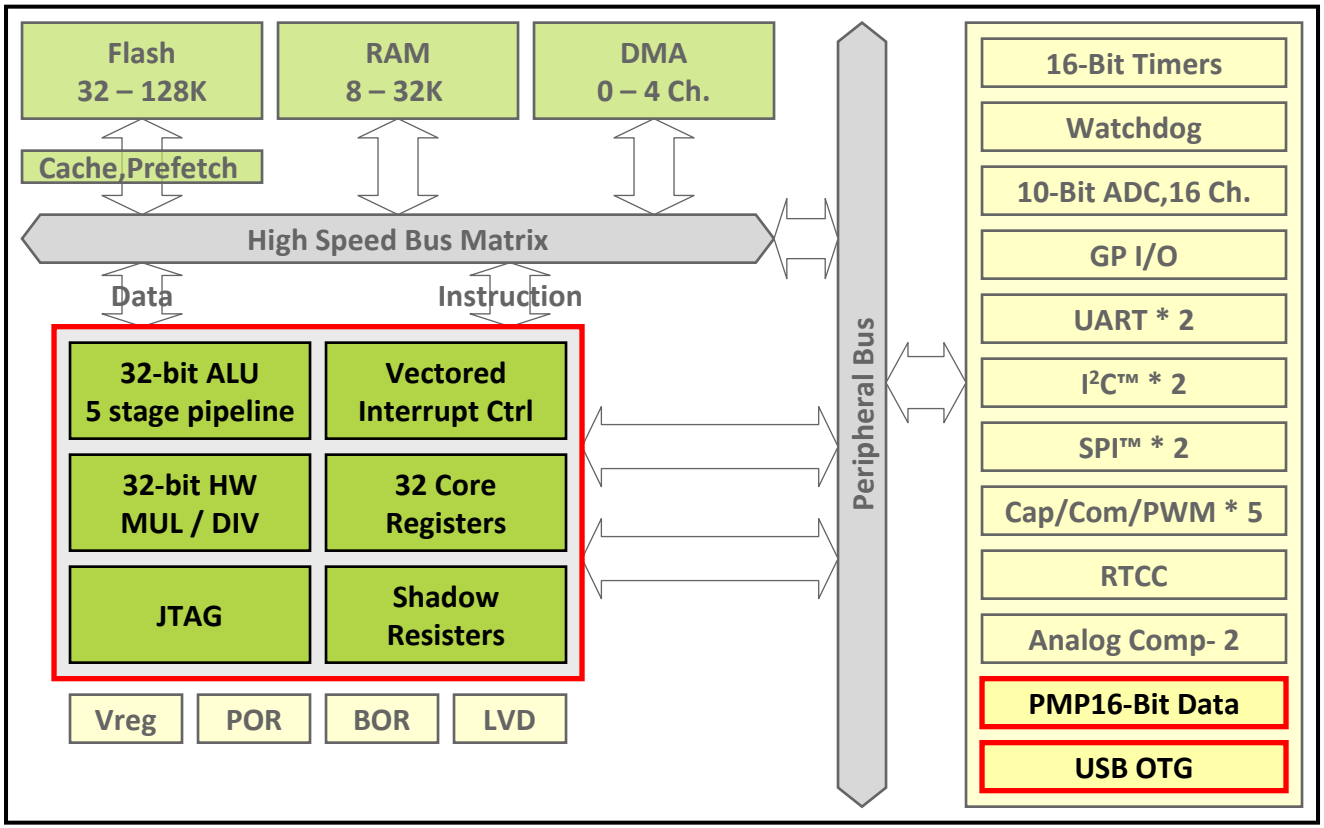


PIC32MX3系列整體架構與PIC24F系列相同,惟獨核心改採用MIPS,PMP的Data Bus擴展為16-Bit。

MICROCHIP
Regional Training Centers

PIC32MX4 Series

USB 2.0 OTG

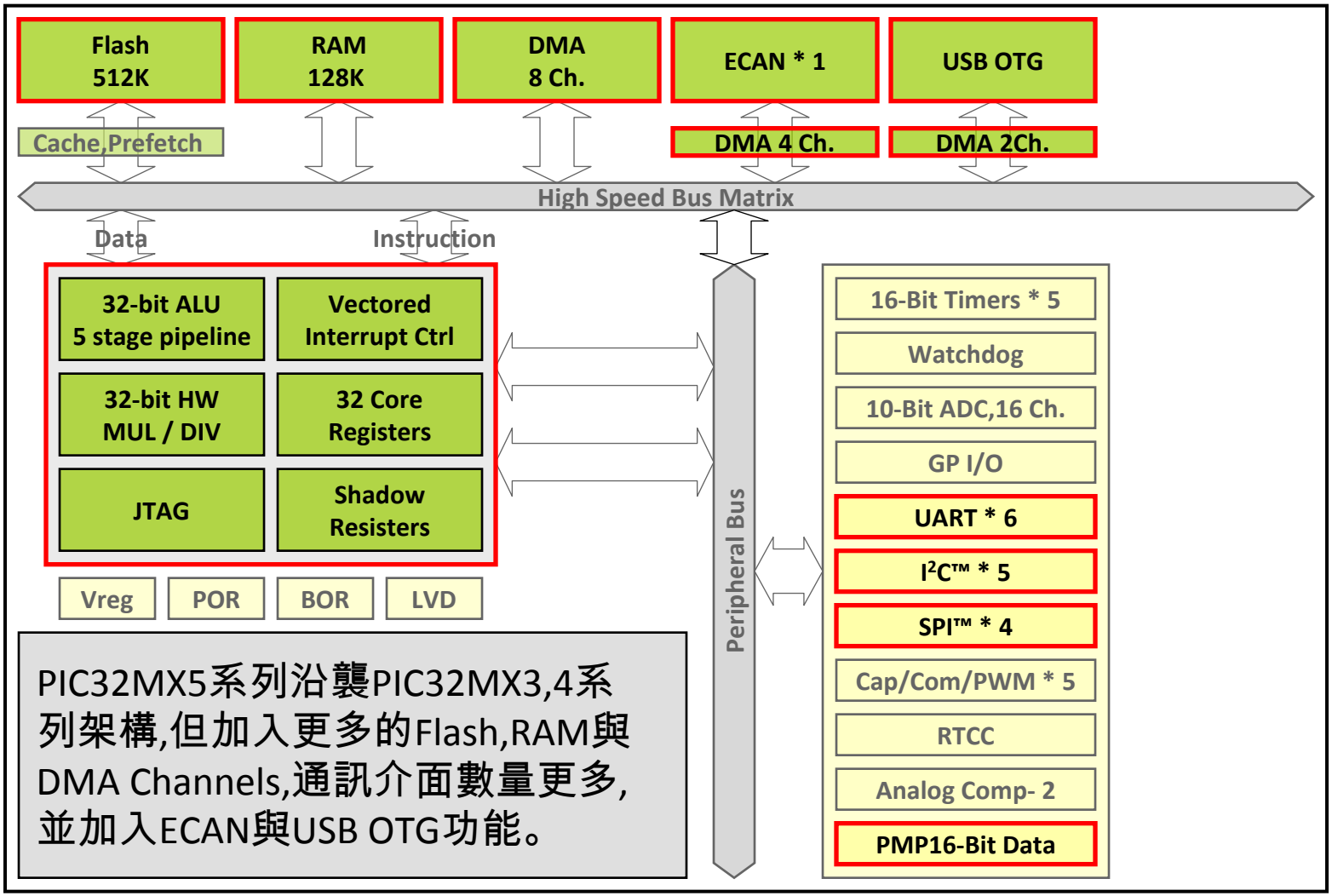


PIC32MX4系列整體架構與PIC24F系列相同,惟獨核心改採用MIPS,PMP的Data Bus擴展為16-Bit,並加入USB OTG功能。

MICROCHIP
Regional Training Centers

PIC32MX5 Series

ECAN,USB OTG

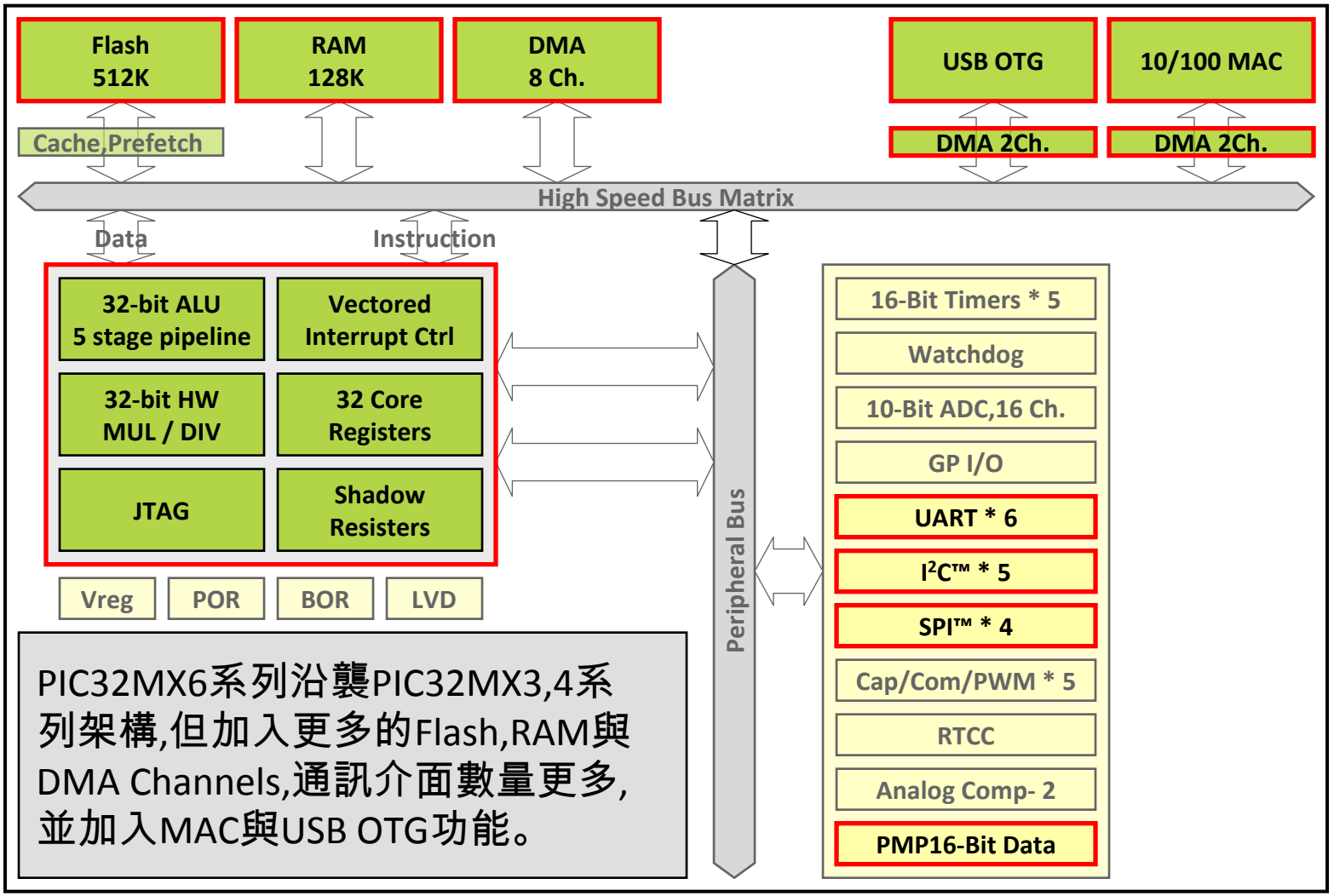


PIC32MX5系列沿襲PIC32MX3,4系列架構,但加入更多的Flash,RAM與DMA Channels,通訊介面數量更多,並加入ECAN與USB OTG功能。

MICROCHIP
Regional Training Centers

PIC32MX6 Series

MAC,USB OTG

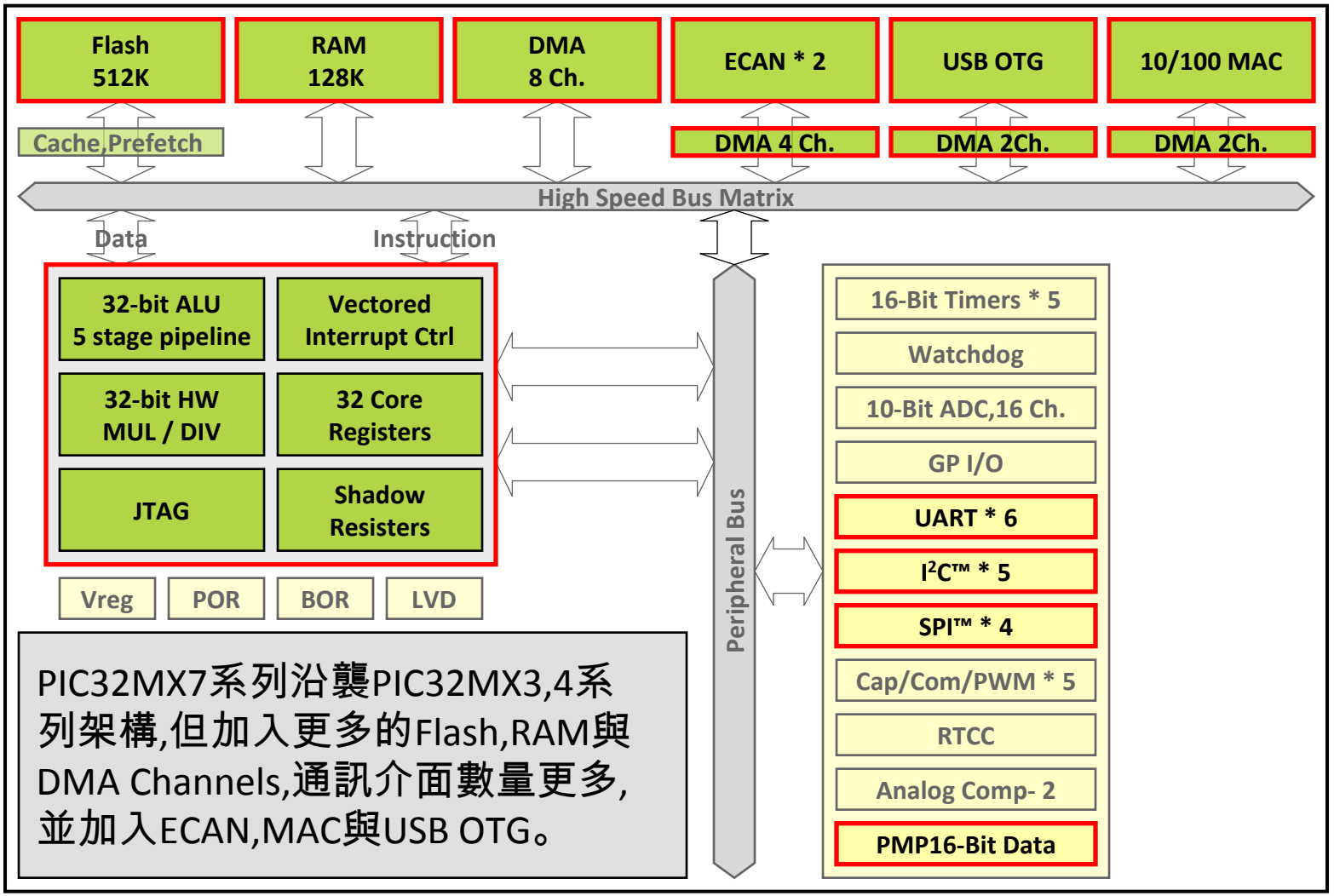


PIC32MX6系列沿襲PIC32MX3,4系列架構,但加入更多的Flash,RAM與DMA Channels,通訊介面數量更多,並加入MAC與USB OTG功能。

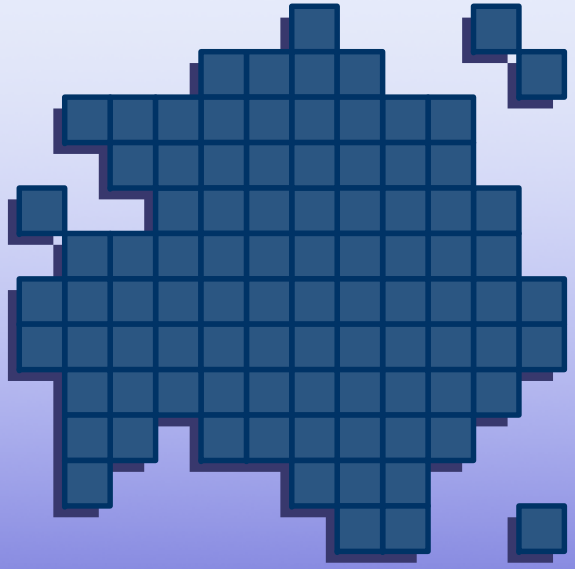
MICROCHIP
Regional Training Centers

PIC32MX7 Series

ECAN,MAC,USB OTG



PIC32MX7系列沿襲PIC32MX3,4系列架構,但加入更多的Flash,RAM與DMA Channels,通訊介面數量更多,並加入ECAN,MAC與USB OTG。

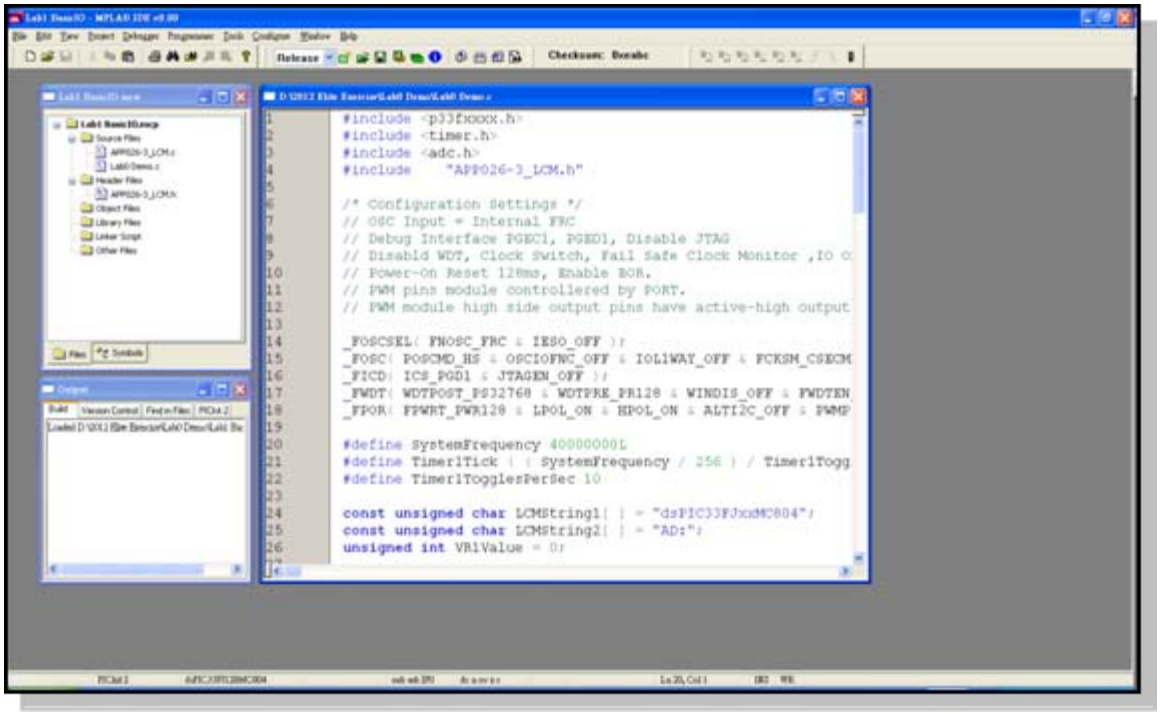


Development Tools

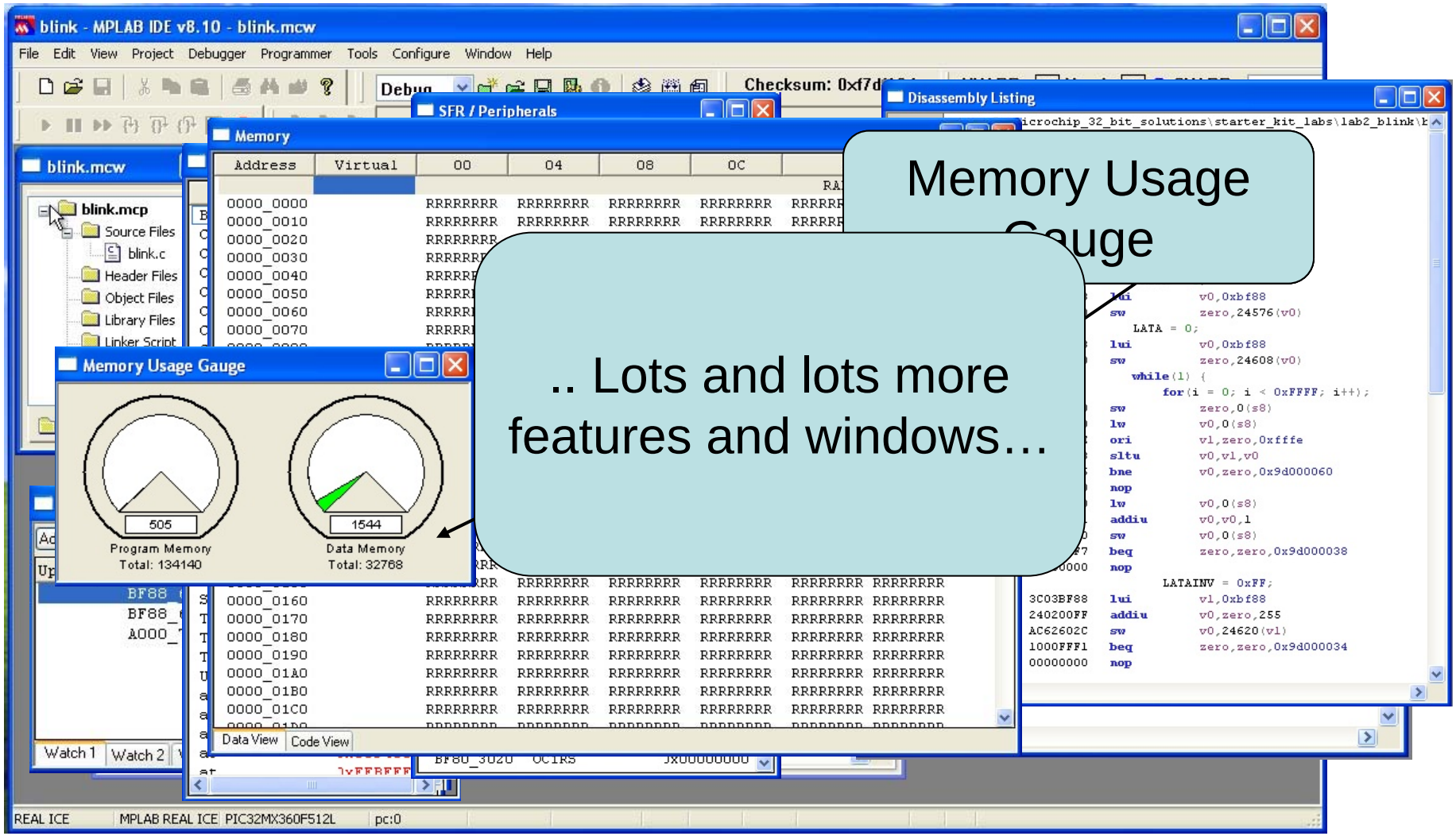
MICROCHIP
Regional Training Centers

MPLAB IDE Introduction

- Microchip提供的整合式開發環境, 支援全系列8-Bits, 16-Bits及32-Bits的MCU。所有的MCU都可透過相同的環境開發。
- 可整合各式的組譯器/編譯器(MPLAB C, Hi-TECH C, etc.),開發工具(PICkit3, ICD3, Real ICE, etc..) 。



MPLAB IDE Introduction



MICROCHIP
Regional Training Centers

MPLAB C32 Introduction

- 支援Microchip 32-Bits MCU的C Compiler 。相容於ANSI規範。
- MPLAB C32可整合於 MPLAB IDE中。
- 提供標準C Functions(printf, scanf, etc..),週邊函式庫,數學運算函數等。
- 可至Microchip網站, 免費取得。
  http://www.microchip.com/c32

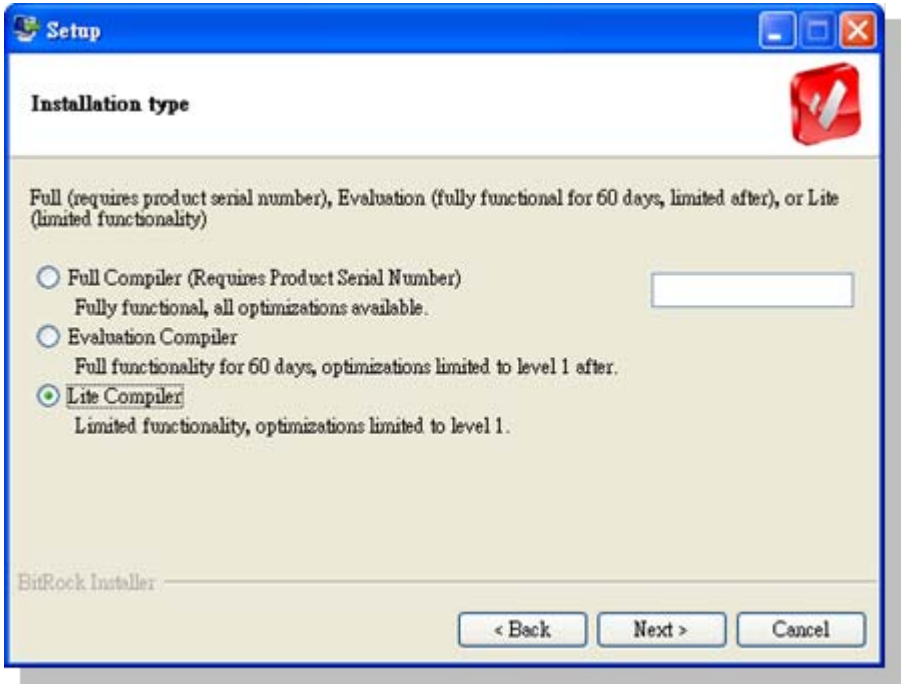
MICROCHIP
Regional Training Centers

About MPLAB C32 Version

- MPLAB C32提供三種不同版本的Compiler:
  正式版(Standard), 精簡版(Lite), 評估版(Evaluation) 。
- 正式版(Standard) *SW006015 - MPLAB C Compiler for PIC32 MCUs 895 USD*
  付費版本,擁有完整編譯功能及最佳化(Optimizations)的功能,可跟精簡版相比,最高可以減少40%的程式碼空間(Code Size)。
- 精簡版(Lite)
  免費版本,擁有完整編譯功能, 與正式版唯一的差異,就是沒有完整的最佳化功能,只擁有Level 1最佳化功能。
- 評估版(Evaluation)
  免費版本, 安裝後60天內,具有標準版的所有功能,60天後自動變成精簡版。

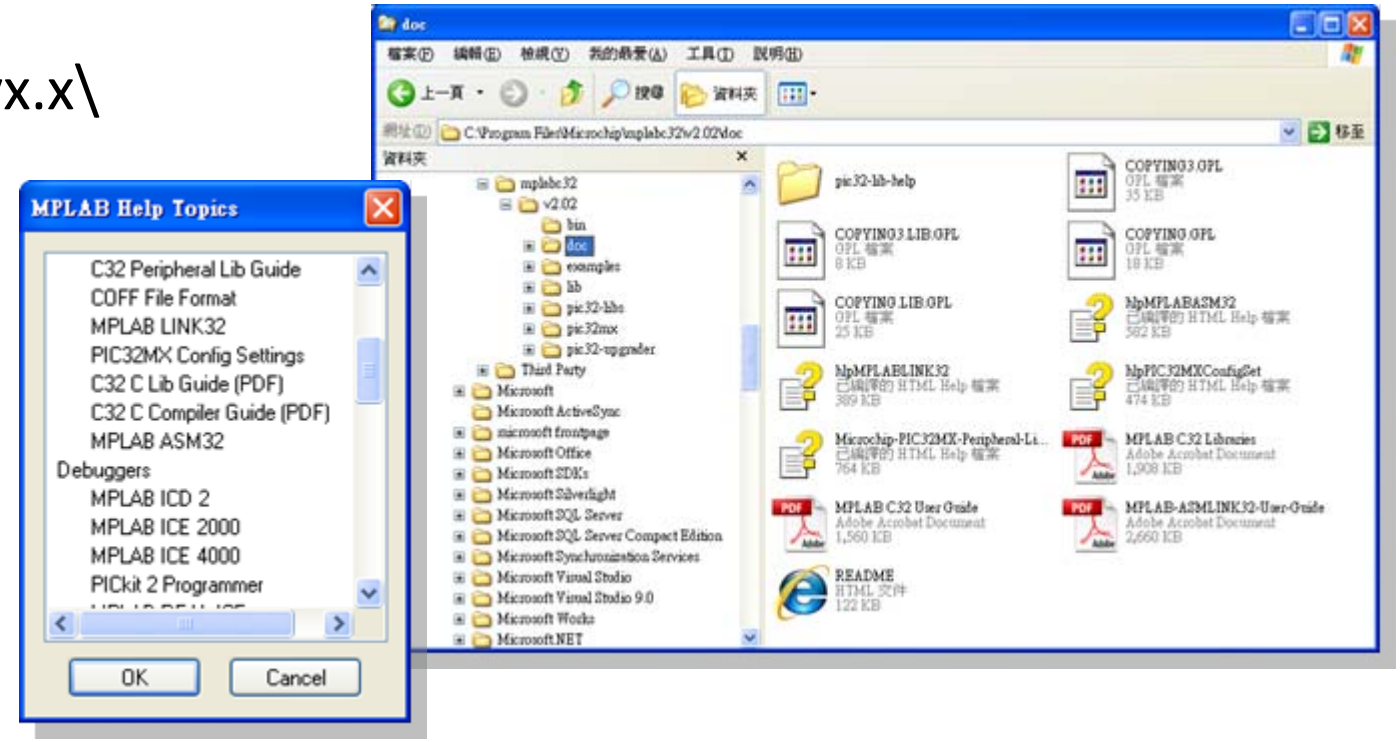
MPLAB C32 Install

- MPLAB IDE安裝時,會一併MPLAB C32。以MPLAB IDE v8.80版本為例, 會一併安裝v2.01版的C32。
- 預設路徑為C:\Program Files\Microchip\MPLAB C32 Suite。
- 目前版本安裝流程,已將正式版(Standard), 精簡版(Lite), 評估版(Evaluation)整合在同一安裝檔中。版本選擇在安裝過程中指定即可。
- 預設安裝路徑 (C:\Program Files\Microchip\MPLABC32\vx.x\) 。



MPLAB C32's Documents

- MPLAB C32的相關文件,可以在MPLAB IDE中的功能表選擇Help\Topics,再由Topics的列表中尋找MPLAB C32的相關文件。
- 也可以在C:\Program Files\Microchip\mplabc32\vx.x\docs\裡找到許多相關的文件。



MICROCHIP
Regional Training Centers

MPLAB C32's Data Type

Type	Bits	Max	Min
char, singed char	8	-128	127
unsigned char	8	0	255
short, singed short	16	-32768	32767
unsigned short	16	0	65535
int, singed int, long, singed long	32	-2^{31}	$2^{31}-1$
unsigned int, unsigned long	32	0	$2^{32}-1$
long long, singed long long	64	-2^{63}	$2^{63}-1$
unsigned long long	64	0	$2^{64}-1$
float	32		
double	64		
Long double	64		

MICROCHIP
Regional Training Centers

C32's Header and Libraries

- MPLAB C32不需要特別引入與MCU相符的MCU標頭檔。與MPLAB C30相同,只需要引入通用標頭檔<p32xxxx.h>,Compiler會根據專案檔所選定之MCU自行判斷並所需載入之MCU標頭檔。
- MPLAB C32使用周邊函式時,不需要特別引入與周邊函式相關的標頭檔,這點與MPLAB C30不同。MPLAB C32簡化了引入的程序,只需引入<plib.h>,即會將所有周邊函式的標頭檔全部載入。
- 以MPLAB C32撰寫程式時,檔案開頭非常簡單,幾乎只會看到*#include* <p32xxxx.h>跟*#include* < plib.h >。

MICROCHIP
Regional Training Centers

C32's Header and Libraries

- <p32xxxx.h> 的實際內容

  (C:\Program Files\Microchip\MPLAB C32\pic32mx\ include\P32xxxx.h)

  …

  #elif defined(__32MX795F512H__)

  #include <proc/p32mx795f512h.h>

  #elif defined(__32MX795F512L__)

  #include <proc/p32mx795f512l.h>

  …

- <plib.h>的實際內容

  (C:\Program Files\Microchip\MPLAB C32\pic32mx\ include\plib.h)

  …

  #include <peripheral/adc10.h>

  #include <peripheral/pmp.h>

  #include <peripheral/cmp.h>

  …

MICROCHIP
Regional Training Centers

Debug/Program Tools

Features/Speed/Trace

PICkit™ 3
Full Speed USB HID,
Run, Halt, SS,
Simple Breakpoints,
Program, Read
All of Microchip's Flash PIC®
MCU and dsPIC DSCs

USD $70.0

MPLAB® ICD 3
High Speed USB 2.0,
Run, Halt, SS
Software Breakpoints,
Complex
Breakpoints,
Program, Read,
All of Microchip's Flash PIC®
MCUs and dsPIC® DSCs

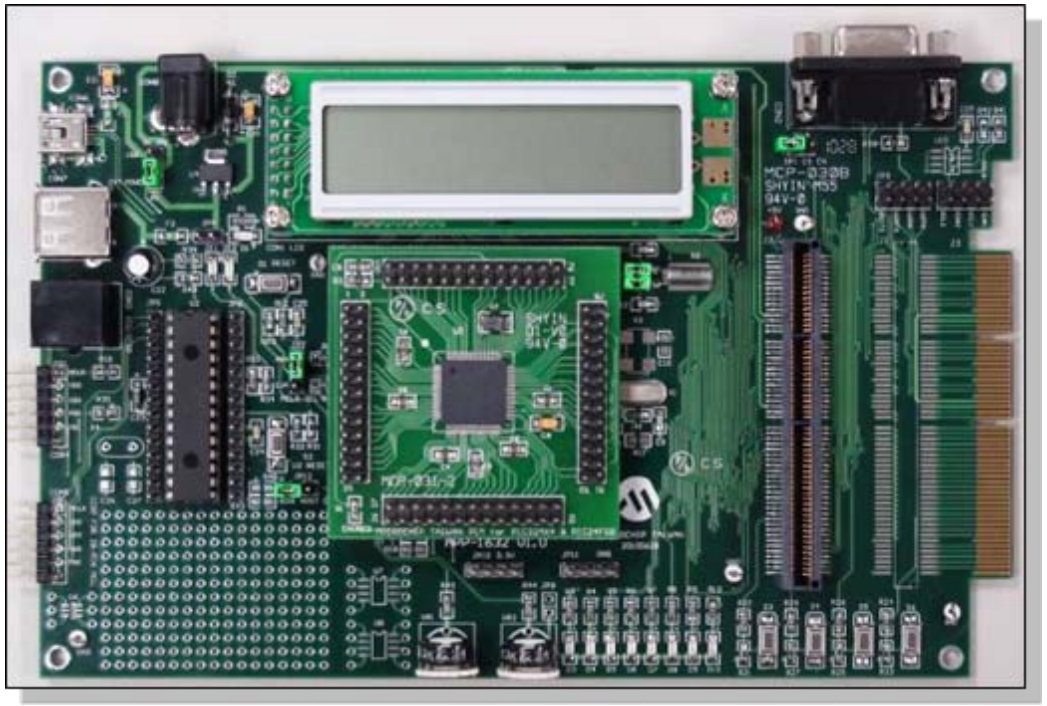
USD $220.0

MPLAB REAL ICE™ Emulator
High Speed USB2.0
Run, Halt, SS
Software Breakpoints,
Complex
Breakpoints/Triggers,
Stopwatch,
Program, Read,
Real-Time Watch,
Log, Trace,
Logic Probes,
Performance Pak
All of Microchip's Flash PIC
MCUs and dsPIC DSCs

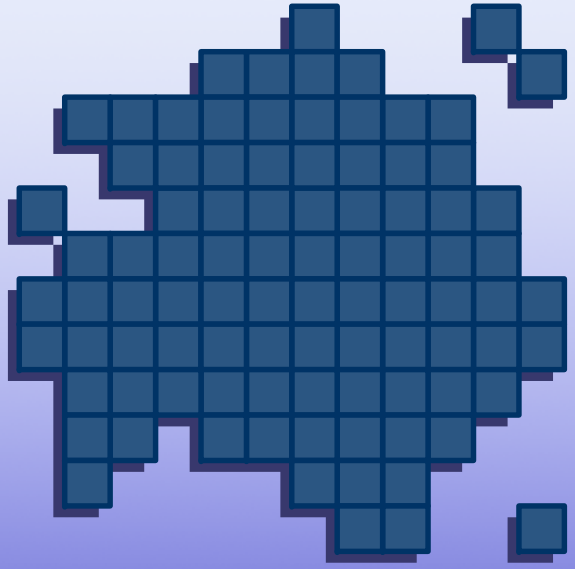
USD $500.0

APP1632 EVM Board

- APP1632實驗板為Microchip台灣公司為PIC32MX系列及十六位元微控制器所設計之多功能實驗板。其設計相容於Explorer 16開發板,並增加功能使其相容於PIC32MX系列。
- 提供LED,Switch, LCD Module,VR,USB and DB9 Connecter 。
- 提供SPI,I^2C EEPROM, Second MCU Socket。
- 支援PICtail+ Socket。
- 排針使用2.54mm Pitch,容易跳線作實驗。



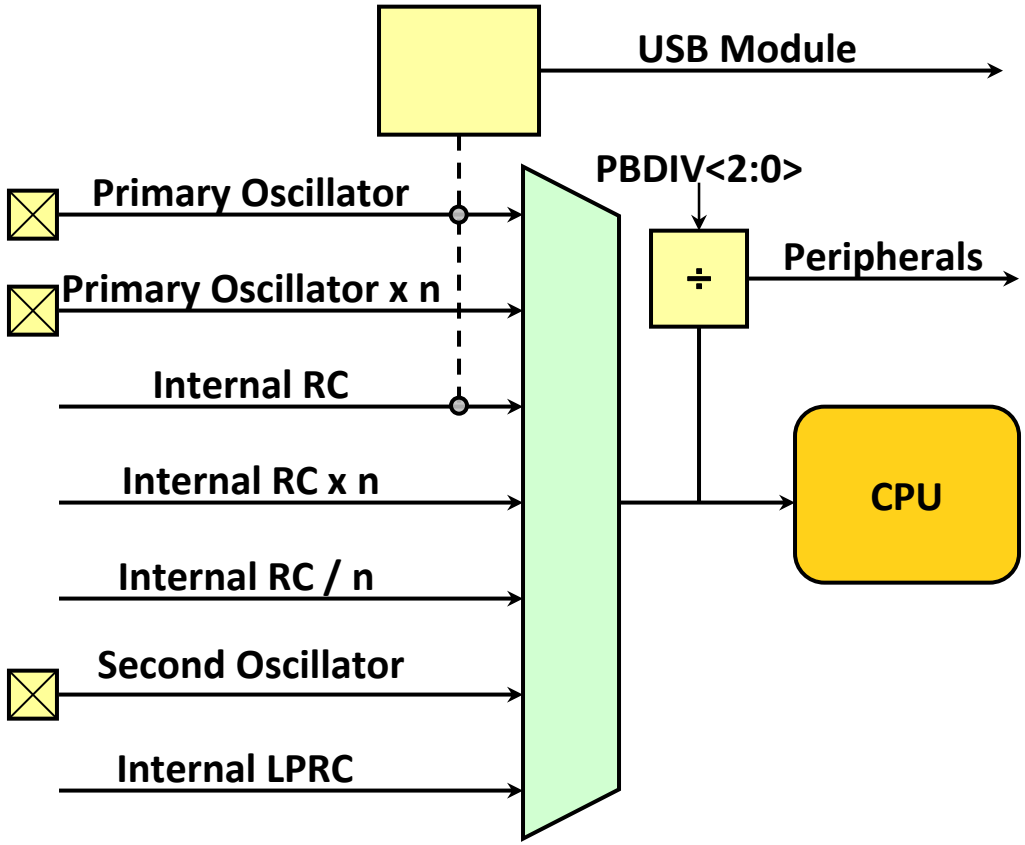
更多細節請參考APP1632使用說明書。



Oscillator and Configuration Bits

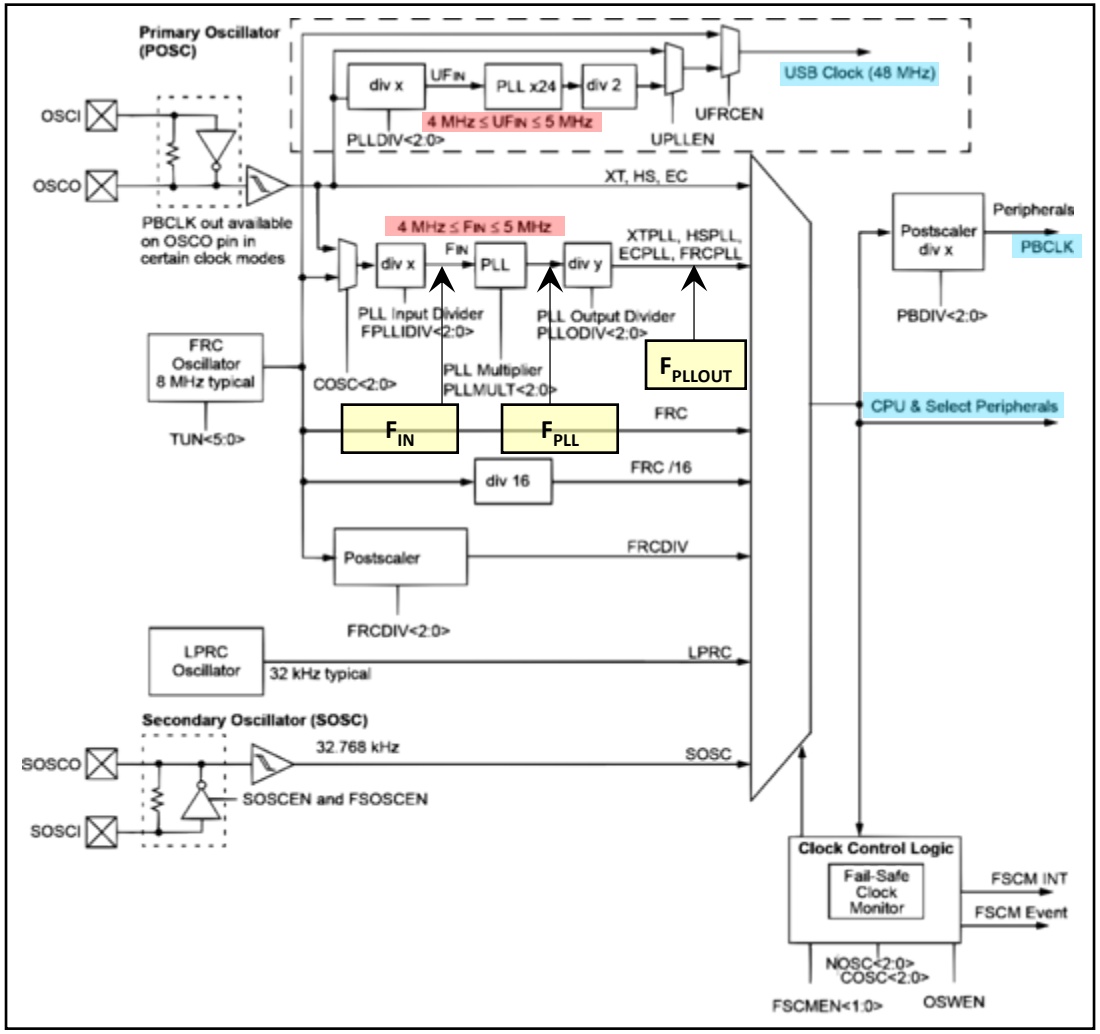
PIC32's Oscillator

- PIC32具有多種時脈來源可選擇。可以使用內部RC或者外部的Crystal, Oscillator。輸入的時脈也可透過內部電路進行倍頻(PLL)或者除頻。
- 時脈來源的設定, 必須正確如果設定錯誤, CPU會接收到錯誤的頻率, 導致程式運作上的時序不對, 或者完全沒有接收到時脈, 導致CPU無法運作。
- PIC32MX4/5/6/7系列具有USB模組,具有獨立的時脈線路。



PIC32's Oscillator

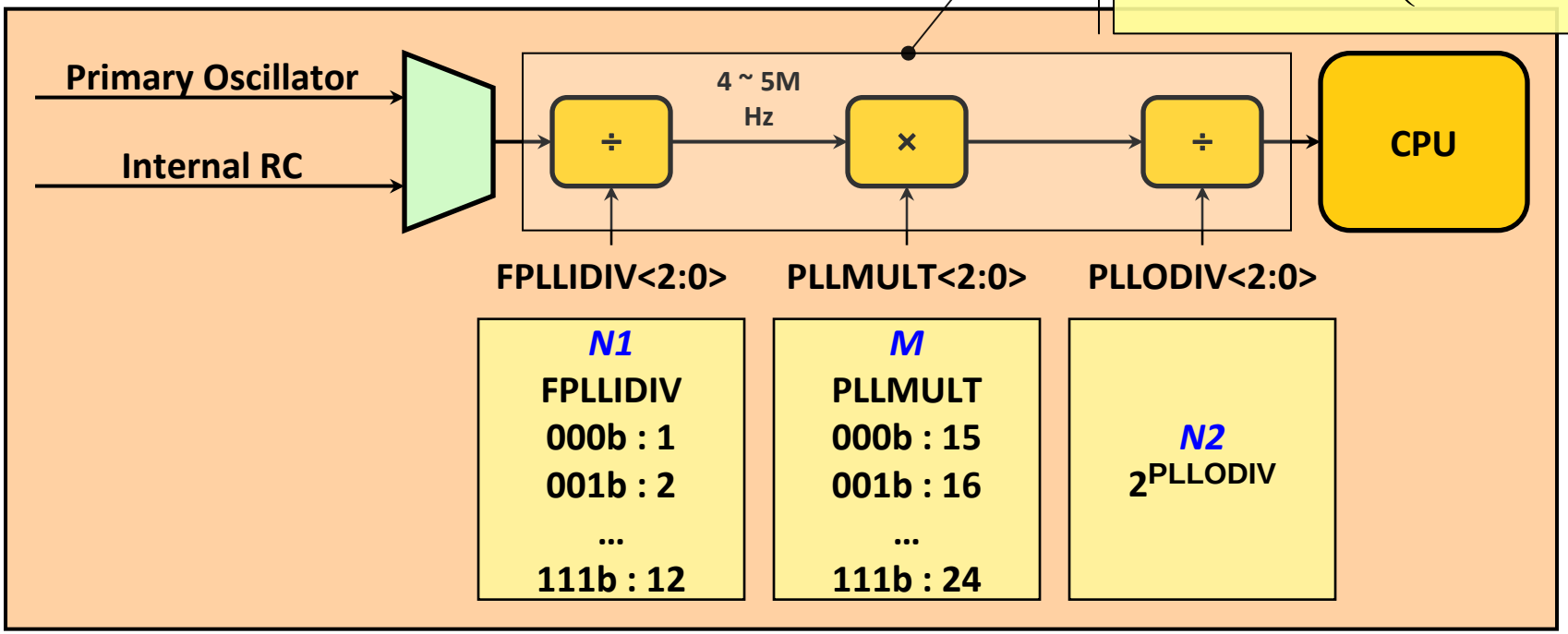
- PIC32有三個時脈系統
  System Clock, B Clock
  USB Clock。
- System Clock:
  供CPU及Core Timer使用。
- PB Clock:
  供其他週邊模組使用,
  例如:PMP,ADC等。
- USB Clock:
  專供USB模組使用。
- 時脈來源的設定必需透過Configuration Bit設定。



Phase Locked Loop

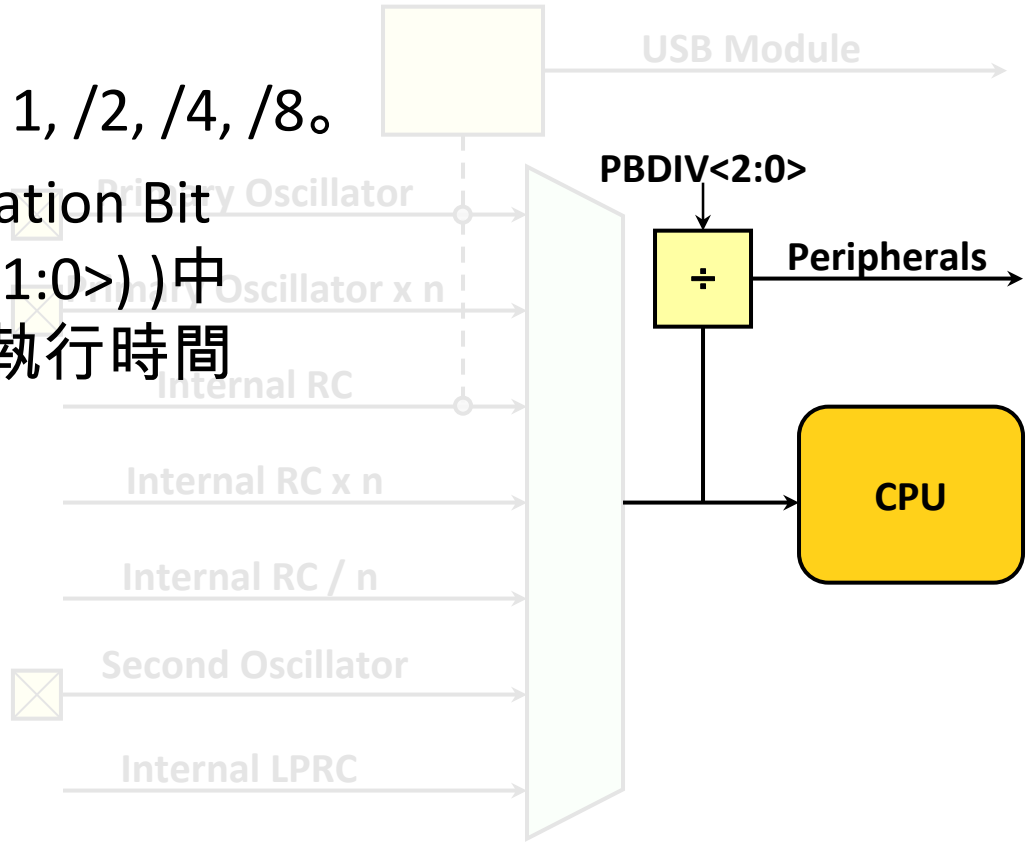
- PIC32可以透過PLL線路,提高供給CPU的頻率(FOSC)。
- Primary Osc.跟FRC可以透過PLL線路倍頻。
- PLL的架構分為三個區塊:

$$F_{OSC} = F_{IN} \times \left(\frac{M}{N1 \times N2} \right)$$



System Clock and PB Clock

- PIC32的系統時脈(System Clock)與周邊時脈(PB Clock)可以運作不同頻率,兩者間存在一除率關係。
- 周邊時脈(PB Clock) =
  系統時脈(System Clock) / 1, /2, /4, /8。
- 除率的設定可在Configuration Bit (OSCCON<20:19> (PBDIV<1:0>))中設定,或在程式中設定於執行時間(Run Time)動態設定。



MICROCHIP
Regional Training Centers

Primary Osc. Mode

- 主要震盪器的震盪模式必須依外部的石英晶體來決定:

Oscillator	Description
HS	10 – 40 MHz Crystal
XT	3.5 – 10 MHz Crystal
EC	0 – 80 MHz External Clock
HSPLL	10 – 40 MHz Crystal with PLL
XTPLL	3.5 – 10 MHz Crystal with PLL
ECPLL	0 – 80 MHz External Clock with PLL

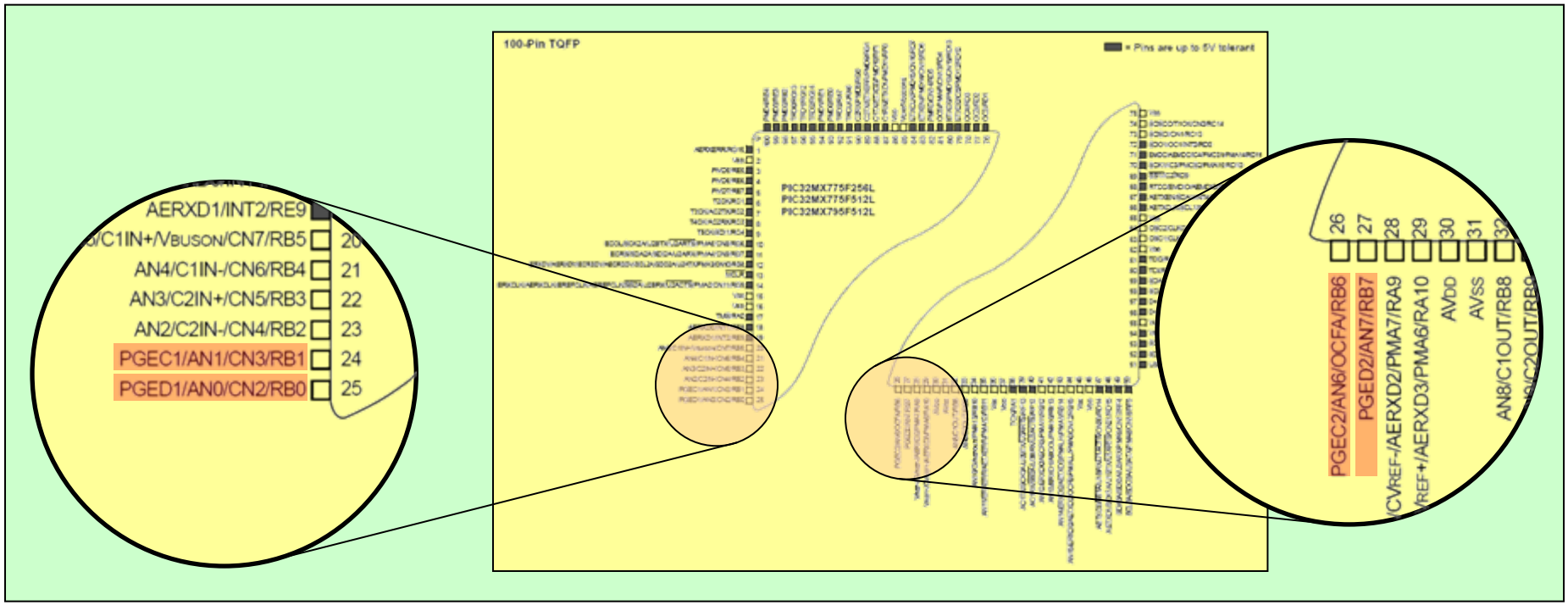
MICROCHIP
Regional Training Centers

Debug & Program Interface

- 開發工具與MCU之間透過PGED/PGEC兩支接腳來溝通,傳輸資料,燒錄程式或進行除錯。
- 每個系列的燒錄接腳組數不同，有些只有一組,有些系列則更多。這些接腳可以用來進行除錯跟燒錄。
- 運作在燒錄模式時, 連接任何一組都可以進行燒錄, 不需指定。
- 運作在除錯模式時, 則必須指定實際連接的組別, 才能進行除錯 (透過Configuration Bits設定, 後續章節會說明) 。

PIC32 ICD/ICSP 接腳位置圖

- 以PIC32MX795F512L-80I/PT為例(100 Pins,TQFP),接腳圖如下所示:
  第一組的ICD/ICSP位於24,25腳(PGEC1/PGED1);
  第二組的ICD/ICSP位於26,27腳(PGEC2/PGED2)。



MICROCHIP
Regional Training Centers

Configuration Bit

- Configuration Bit是用來設定MCU中一些重要的模式。其中包含如:CPU的時脈來源,除錯工具使用的接腳,Watchdog,程式碼保護,程式碼防寫等設定。
- 其中又以CPU的時脈來源,除錯工具使用的接腳設定最重要,時脈來源設定錯誤CPU無法正常運作,除錯工具使用的接腳設定錯誤,則無法使用硬體除錯工具。
- 參考Datasheet的Special Feature章節可以看到完整的說明。

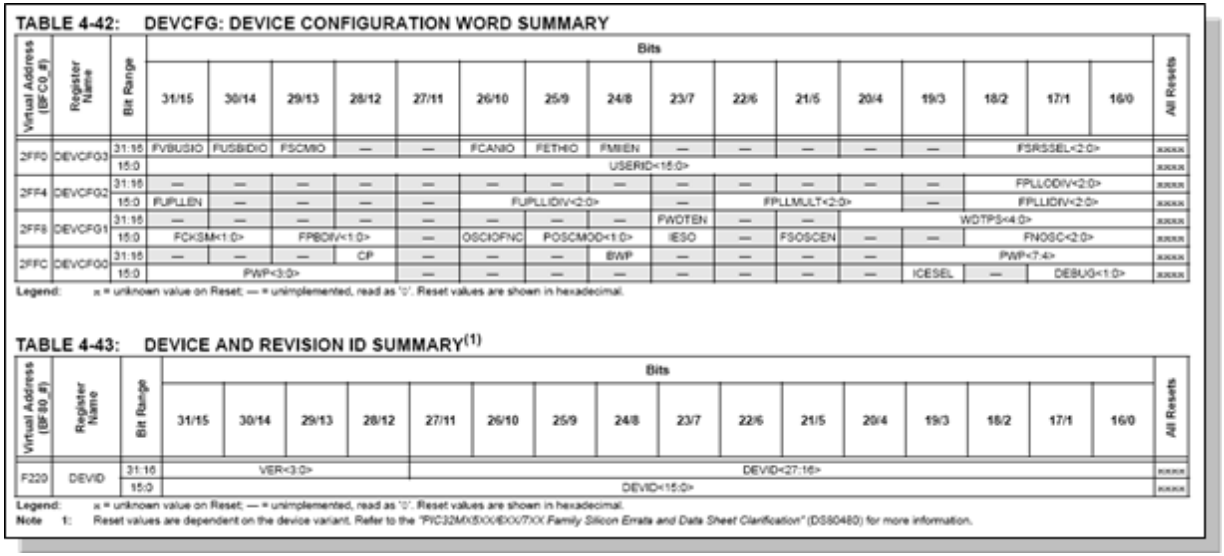


TABLE 4-42: DEVCFG: DEVICE CONFIGURATION WORD SUMMARY

Virtual Address (BF8C 0#)	Register Name	Bit Range	Bits 31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0	All Resets
2FF0	DEVCFG3	31:16	FVBUSIO	FUSBIDIO	FSCMIO	—	—	FCANIO	FETHIO	FMIIEN	—	—	—	—	—	FSRSSEL<2:0>			xxxx
		15:0	USERID<15:0>																xxxx
2FF4	DEVCFG2	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	FPLLODIV<2:0>			xxxx
		15:0	FUPLLEN	—	—	—	—	FUPLLIDIV<2:0>			—	FPLLMULT<2:0>			—	FPLLIDIV<2:0>			xxxx
2FF8	DEVCFG1	31:16	—	—	—	—	—	—	—	—	FWDTEN	—	—	WDTPS<4:0>					xxxx
		15:0	FCKSM<1:0>		FPBDIV<1:0>		—	OSCIOFNC	POSCMOD<1:0>		IESO	—	FSOSCEN	—	—	FNOSC<2:0>			xxxx
2FFC	DEVCFG0	31:16	—	—	—	CP	—	—	—	BWP	—	—	—	—	PWP<7:4>				xxxx
		15:0	PWP<3:0>				—	—	—	—	—	—	—	—	ICESEL	—	DEBUG<1:0>		xxxx

Legend: x = unknown value on Reset; — = unimplemented, read as '0'. Reset values are shown in hexadecimal.

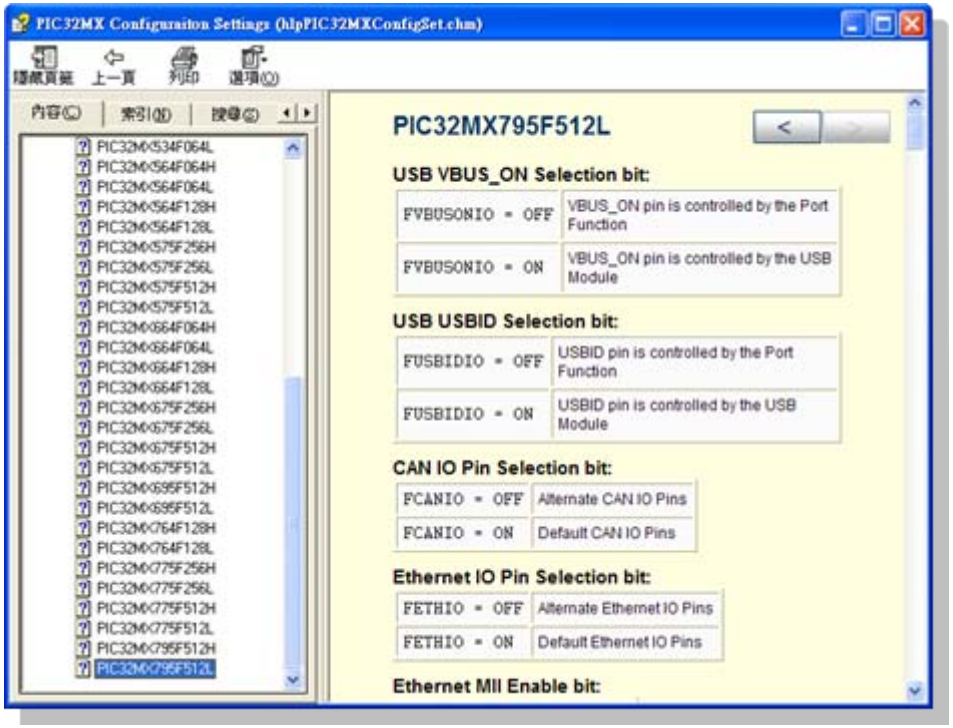
TABLE 4-43: DEVICE AND REVISION ID SUMMARY(1)

Virtual Address (BF80 8#)	Register Name	Bit Range	Bits 31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0	All Resets
F220	DEVID	31:16	VER<3:0>				DEVID<27:16>												xxxx
		15:0	DEVID<15:0>																xxxx

Legend: x = unknown value on Reset; — = unimplemented, read as '0'. Reset values are shown in hexadecimal.
Note 1: Reset values are dependent on the device variant. Refer to the "PIC32MX5XX/6XX/7XX Family Silicon Errata and Data Sheet Clarification" (DS80480) for more information.

C32 對 Configuration Bit 的支援

- MPLAB C32可以在程式碼中直接設定Configuration Bit。
  利用 *#pragma config* 在程式裡來設定,語法如下例所示:
  #pragma config UPLLIDIV = DIV_2
  #pragma config FPLLMUL = MUL_15 , FPLLIDIV = DIV_2
  ...
- *config* 的有效定義字與意義可參考說明文件 hlpPIC32MXConfigSet。
  預設路徑為
  C:\Program Files\Microchip\MPLAB C32\doc\hlpPIC32MXConfigSet.chm
- 也可透過 MPLAB IDE 的GUI直接修改所需的配置。

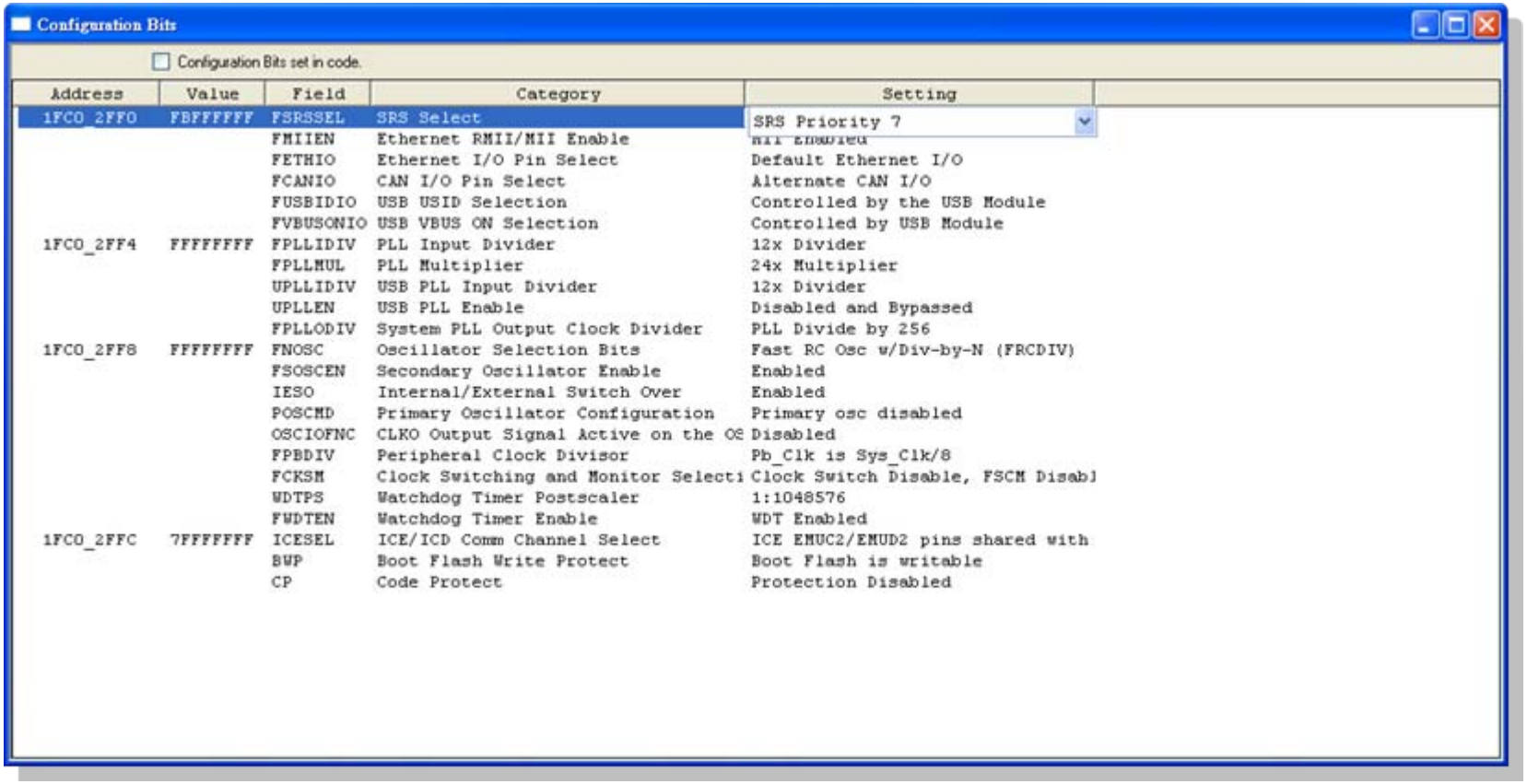


MICROCHIP
Regional Training Centers

C32 對 Configuration Bit 的支援

- MPLAB IDE GUI

  MPLAB IDE Configure\Configuration Bits



Address	Value	Field	Category	Setting
1FC0_2FF0	FBFFFFFF	FSRSSEL	SRS Select	SRS Priority 7
		FMIIEN	Ethernet RMII/MII Enable	MII Enabled
		FETHIO	Ethernet I/O Pin Select	Default Ethernet I/O
		FCANIO	CAN I/O Pin Select	Alternate CAN I/O
		FUSBIDIO	USB USID Selection	Controlled by the USB Module
		FVBUSONIO	USB VBUS ON Selection	Controlled by USB Module
1FC0_2FF4	FFFFFFFF	FPLLIDIV	PLL Input Divider	12x Divider
		FPLLMUL	PLL Multiplier	24x Multiplier
		UPLLIDIV	USB PLL Input Divider	12x Divider
		UPLLEN	USB PLL Enable	Disabled and Bypassed
		FPLLODIV	System PLL Output Clock Divider	PLL Divide by 256
1FC0_2FF8	FFFFFFFF	FNOSC	Oscillator Selection Bits	Fast RC Osc w/Div-by-N (FRCDIV)
		FSOSCEN	Secondary Oscillator Enable	Enabled
		IESO	Internal/External Switch Over	Enabled
		POSCMD	Primary Oscillator Configuration	Primary osc disabled
		OSCIOFNC	CLKO Output Signal Active on the OS	Disabled
		FPBDIV	Peripheral Clock Divisor	Pb_Clk is Sys_Clk/8
		FCKSM	Clock Switching and Monitor Selecti	Clock Switch Disable, FSCM Disabl
		WDTPS	Watchdog Timer Postscaler	1:1048576
		FWDTEN	Watchdog Timer Enable	WDT Enabled
1FC0_2FFC	7FFFFFFF	ICESEL	ICE/ICD Comm Channel Select	ICE EMUC2/EMUD2 pins shared with
		BWP	Boot Flash Write Protect	Boot Flash is writable
		CP	Code Protect	Protection Disabled

MICROCHIP
Regional Training Centers

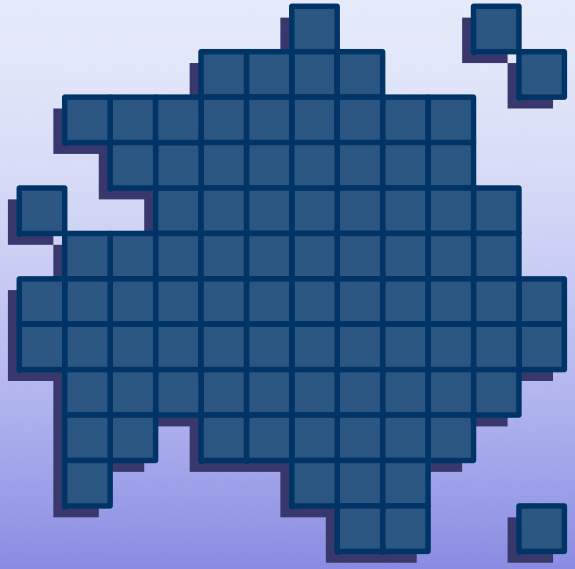
C32 對 Configuration Bit 的支援

- 建議將Configuration Bit的設定,寫在程式碼中。
  (*#pragma config* xxxx = xxxx)
- 因為使用MPLAB IDE的GUI設定時,Configuration Bit的設定是儲存在專案檔中(*.mcp) 。
- 將來如果想將提供程式給其他人時。此時如果只提供原始碼(*.c,*.h),沒有附上專案檔,或者對方沒有安裝MPLAB IDE則無法獲得正確的Configuration Bit的設定,會導致MCU動作不正常。
- 程式中的設定如果與GUI設定衝突時, 以程式中設定的為準。

C32 的基本程式框架

- Configuration Bit設定範本
  以APP1632, APP1632-2為例, 使用外部 8M Hz Crystal搭配PLL, 供給CPU 60MHz。

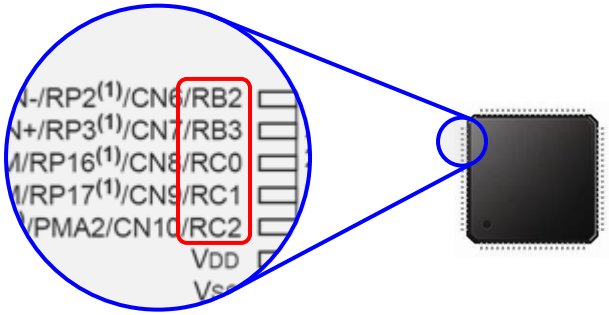
```
#include <p32xxxx.h>
#include <plib.h>
/* Configuration Settings */
// OSC Input = HS w/PLL
// CPU Clock = ( ( OSC Input / 2 ) * 15 ) / 1
// PB Clock = CPU Clock / 1
// Disabld WDT , ICSP/ICD = PGC1 , PGD1
#pragma config FNOSC = PRIPLL , POSCMOD = HS
#pragma config FPLLIDIV = DIV_2 , FPLLMUL = MUL_15 , FPLLODIV = DIV_1
#pragma config FPBDIV = DIV_1
#pragma config FWDTEN = OFF , ICESEL = ICS_PGx1

int main ( void )
{
    while( 1 );
}
```

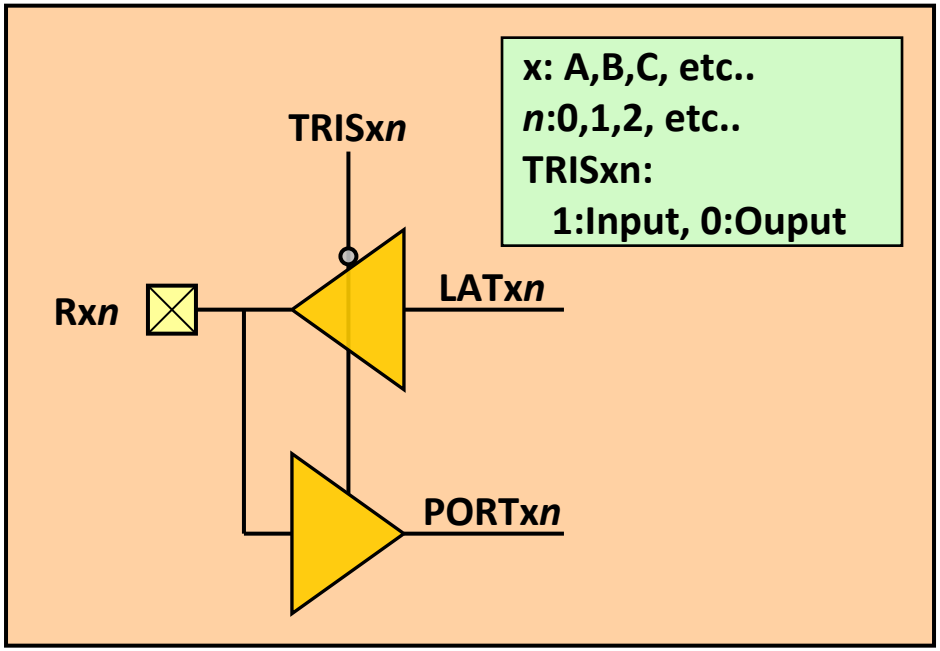


I/O Port Architecture

I/O Port Block Diagram

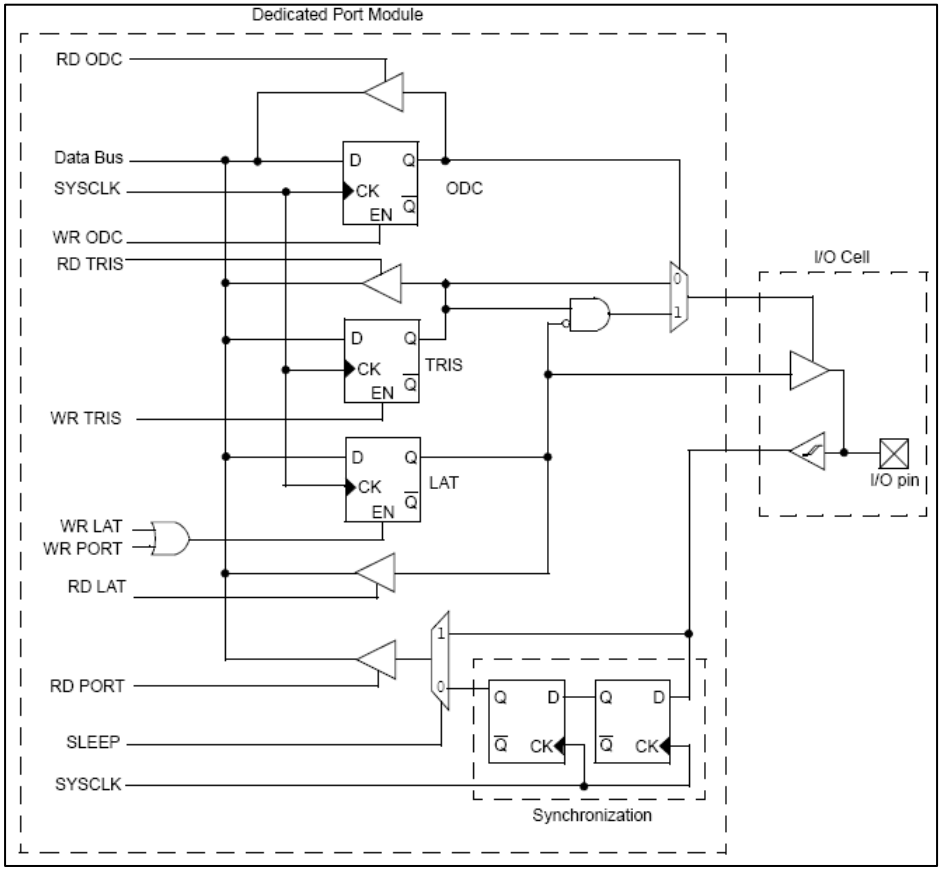
- 32-Bits MCU的IO Port示意圖, 如圖所示。
- 所有的IO Port都有TRIS, LAT跟PORT特殊功能暫存器。TRIS用來設定IO的方向,要輸出時(Ex:控制LED)TRISx*n*必須設為”0” 。要輸入(Ex:讀取按鍵狀態)時TRISx*n*必須設為”1”。
- 設為輸出時,要輸出的狀態填入LATx*n*,對應的接腳Rx*n*就會有輸出。
- 設為輸入時,可以自PORTx*n*取得外部的狀態。





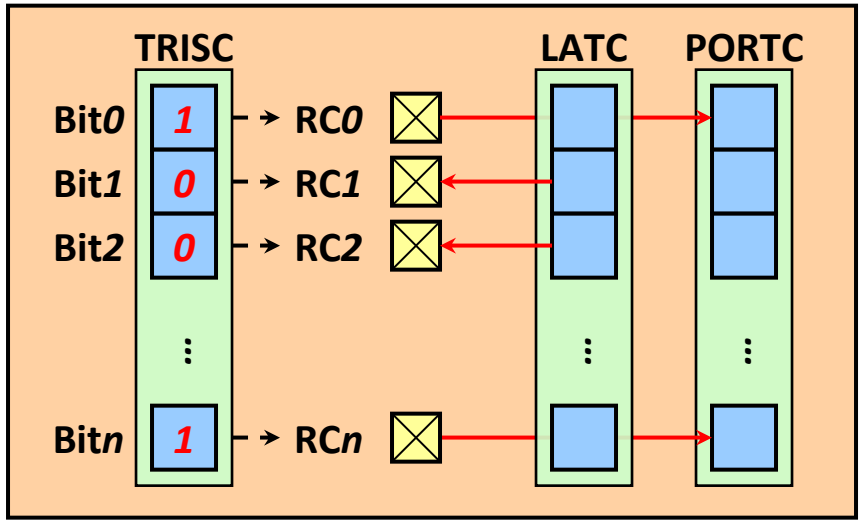
I/O Port Block Diagram

- 32-Bits MCU的IO Port架構方塊圖, 如圖所示。簡易示意圖是經過簡化後,便於了解,實際的架構為下圖。



I/O Port 的操作

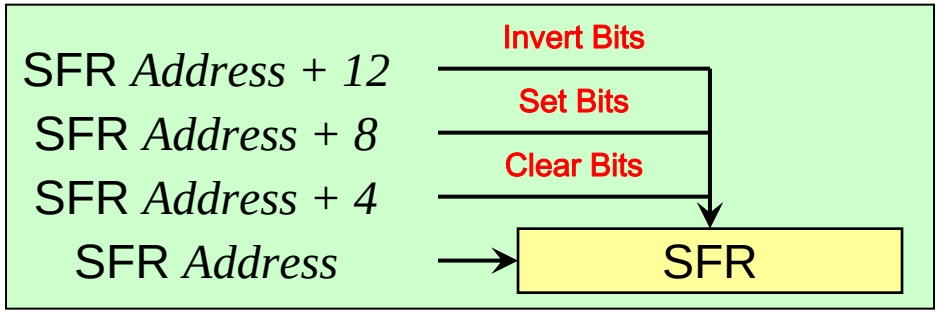
- TRIS, LAT跟PORT特殊功能暫存器都是一個32-Bits的暫存器,暫存器中每個Bit都控制著對應的IO接腳。
- 以IO Port C為例,TRISC, LATC, PORTC的Bit 0,控制RC0接腳。
- 要用RC0控制LED時,必須先把TRISC的Bit 0設為0(輸出),然後把要輸出的狀態,填入LATC的Bit 0。
- 要用RC0讀取按鍵狀態時,則必須把TRISC的Bit 0設為1(輸入),然後就可以從PORTC的Bit 0取得外部的狀態。



多位元的位元操作

Atomic Bits Manipulation

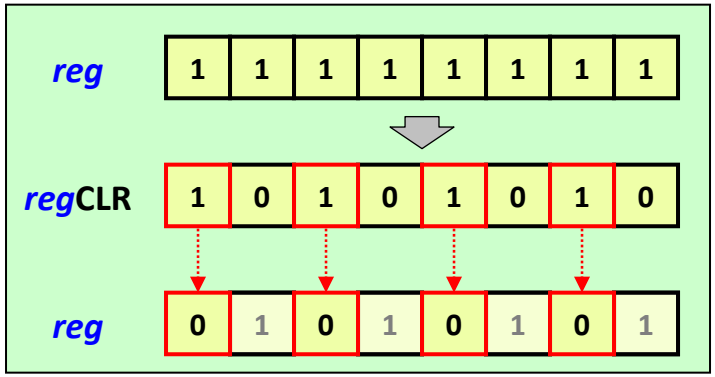
- PIC32提供新的暫存器操作方式- Atomic Bits Manipulation。主要用於快速,大量的暫存器位元變換操作。所有的特殊功能暫存器都有此特徵。但一般的Data memory不具有此功能。
- 每個特殊功能暫存器都搭配有額外的三個位元操作暫存器,"<reg>CLR","<reg>SET","<reg>INV",提供三種不同的操作功能。
- 位元操作暫存器的實際位址是以特殊功能暫存器為基底往上的佔用十二個位元組的空間。該暫存器僅供位元操作用,僅能寫入。讀取該暫存器所獲
- 得數值是不具任何意義的。



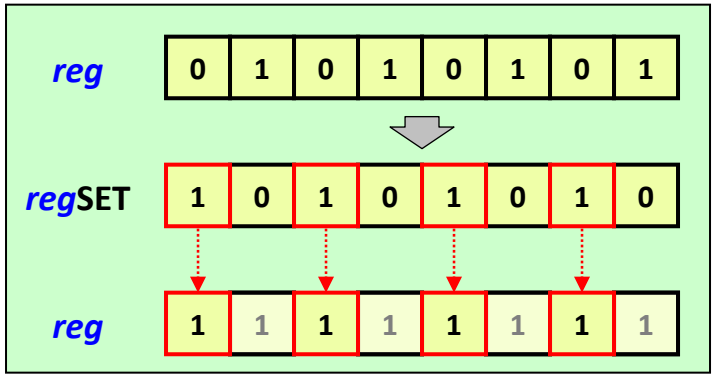
多位元的位元操作

Atomic Bits Manipulation

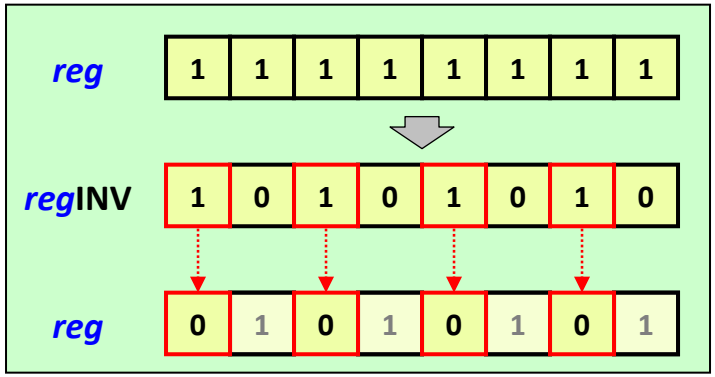
- 寫入特殊功能暫存器中有”1”的位元,則會執行對應的操作。



*reg*CLR:對應位元清除為”0”



*reg*SET:對應位元設定為”1”



*reg*INV:對應位元作反向

C32 對 I/O Port 的支援

- 常用的C32 I/O Port函數

```
PORTSetPinsDigitalIn( );          // 設定對應的I/O Port Bit為數位輸入
PORTSetPinsDigitalOut( );         // 設定對應的I/O Port Bit為數位輸出
PORTSetPinsAnalogIn( );           // 設定對應的I/O Port Bit為類比輸入
PORTSetPinsAnalogOut( );          // 設定對應的I/O Port Bit為類比輸出
PORTRead( );                      // 讀取I/O Port的資料
PORTReadBits( );                  // 讀取I/O Port Bit的資料
PORTWrite( );                     // 寫入I/O Port
PORTSetBits( );                   // 設定I/O Port Bit為”1”
PORTClearBits( );                 // 清除I/O Port Bit為”0”
PORTToggleBits( );                // 反向I/O Port Bit(“1”->”0”,”0”->”1”)
…
```

- 詳細說明請參閱 32-bit-Peripheral-Library-Guide.pdf,Sec.10 I/O Port Library。

C32 對 I/O Port 的支援

- 常用的C32 I/O Port巨集

```
mPORTxSetPinsDigitalIn( );      // 設定對應的I/O Portx Bit為數位輸入
mPORTxSetPinsDigitalOut( );     // 設定對應的I/O Portx Bit為數位輸出
mPORTxSetPinsAnalogIn( );       // 設定對應的I/O Portx Bit為類比輸入
mPORTxSetPinsAnalogOut( );      // 設定對應的I/O Portx Bit為類比輸出
mPORTxRead( );                  // 讀取I/O Portx的資料
mPORTxReadBits( );              // 讀取I/O Port Bitx的資料
mPORTxWrite( );                 // 寫入I/O Portx
mPORTAToggleBits( );            // 反向I/O Portx Bit(“1”->”0”,”0”->”1”)
…
```

- 詳細說明請參閱
  32-bit-Peripheral-Library-Guide.pdf,Sec.10 I/O Port Library。

C32 I/O Code Example

- 簡單的I/O設定為數位輸出,輸出操作範例

```
mPORTBSetPinsDigitalOut( BIT_2 | BIT_3 );
mPORTBSetBits( BIT_2 );
mPORTBClearBits( BIT_3 );
mPORTBToggleBits( BIT_2 | BIT_3 );
```

- 簡單的I/O設定為類比輸入範例

```
mPORTASetPinsAnalogIn( BIT_0 );
```

mPORTASetPinsDigitalOut
mPORTBSetPinsDigitalOut
mPORTCSetPinsDigitalOut
mPORTDSetPinsDigitalOut
mPORTESetPinsDigitalOut
mPORTFSetPinsDigitalOut
mPORTGSetPinsDigitalOut

Description: This macro configures the TRISx register bits as outputs.

Include: `plib.h`

Prototype:
```
void mPORTASetPinsDigitalOut(unsigned int _bits);
void mPORTBSetPinsDigitalOut(unsigned int _bits);
void mPORTCSetPinsDigitalOut(unsigned int _bits);
void mPORTDSetPinsDigitalOut(unsigned int _bits);
void mPORTESetPinsDigitalOut(unsigned int _bits);
void mPORTFSetPinsDigitalOut(unsigned int _bits);
void mPORTGSetPinsDigitalOut(unsigned int _bits);
```

Arguments: *_bits* This argument contains one or more bit masks bitwise OR'd together. Select one or more masks from the mask set defined below. Note: An absent mask symbol assumes corresponding bit(s) are disabled, or default value, and will be set = 0.

IO Pin Bit Masks
BIT_0
BIT_1
BIT_2
...
BIT_15

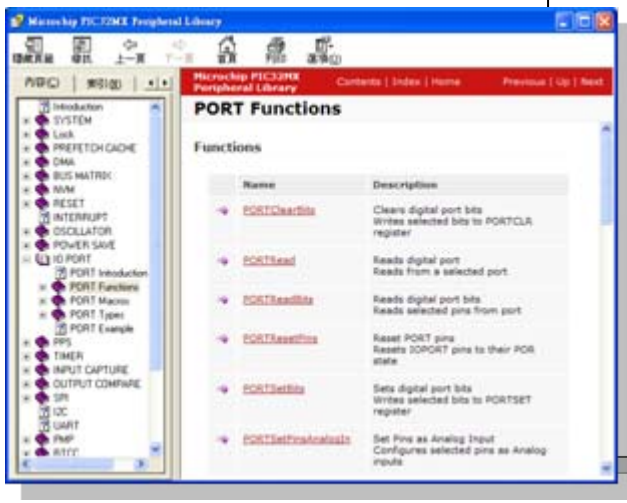
Return Value: None

Remarks: Argument is copied to the TRISCLRx register. If a bit is = '1', the corresponding IO pin becomes an output; if a bit = '0', the corresponding IO pin is not affected.
For those IO pins that share digital and analog functionality, the corresponding ADPCFG bits are set appropriately by the macro. See code example.

Same as mPORTxConfigOutput

Source File: None

Code Example:
```
/* make PORTE<7:6> = outputs */
mPORTESetPinsDigitalOut(BIT_7 | BIT_6);

/* PORTD<3> = output, all others not affected */
mPORTDSetPinsDigitalOut(0x0008);
```



MICROCHIP
Regional Training Centers

JTAG 與 PORTA

- PIC32除了可以使用ICD/ICSP介面做除錯與燒錄外,也可透過JTAG介面做除錯。
- JTAG介面的訊號腳與PORTA是共用的,如果有需要用到PORTA上的其他功能時,必須關閉JTAG,才不會造成PORTA不聽話的狀況。
- 建議直接關閉,除非有打算使用JTAG作為除錯介面,不然打開只是造成不必要的困惱。可使用以下敘述關閉JTAG功能:

```
// Disable the JTAG function
DDPCONbits.JTAGEN = 0;
```

MICROCHIP
Regional Training Centers

Lab1 – BasicIO (15 mins)

- 嘗試在Lab1的程式基礎上, 加入IO Port的控制程式。
- 利用C32所提供之Function控制PORTA, 讓PORTA的所有接腳可以不斷的轉態(Toggle, 1->0->1->...) 。 每次Toggle的時間間隔設為500 mS 。
- 軟體延遲時間,可利用MPLAB SIM提供的Stopwatch來計算。

- 觀察程式變化!

MICROCHIP
Regional Training Centers

Discuss

- 先將程式中SYSTEMConfig( )改為註解。
- 編譯後重新使用軟體模擬觀察軟體模擬的時間, 應該仍然是500mS。
- 接著再編譯,燒錄,執行狀況有何變化?

MICROCHIP
Regional Training Centers

Prefetch, Cache's Default Setting

- 原本是的變化時間與現在相差近10倍,為何如此?
  - Pre-fetch跟Cache功能沒有開啟,Wait State恢復到預設值。
    Flash 的 Wait State 設為最長時間 (7 Cycles)
    RAM 的 Wait State 設為最長時間 (1 Cycles)
- 使用MPLAB SIM的Stopwatch時要特別注意,MPLAB SIM無法模擬出實際的效能,因為 MPLAB SIM是以系統理想狀態(無快取失誤,理想效能曲線)並假設PB Clock = System Clock去計算執行時間。
- MPLAB C32提供數個系統函數來設定Pre-fetch,Cache,Wait State,設定的依據為系統的工作頻率(System Clock)。

MPLAB C32 System Function

- 常用的系統函數:

  SYSTEMConfig( );
  SYSTEMConfigPerformance( );
  SYSTEMConfigWaitStatesAndPB( );
  SYSTEMConfigPB( );

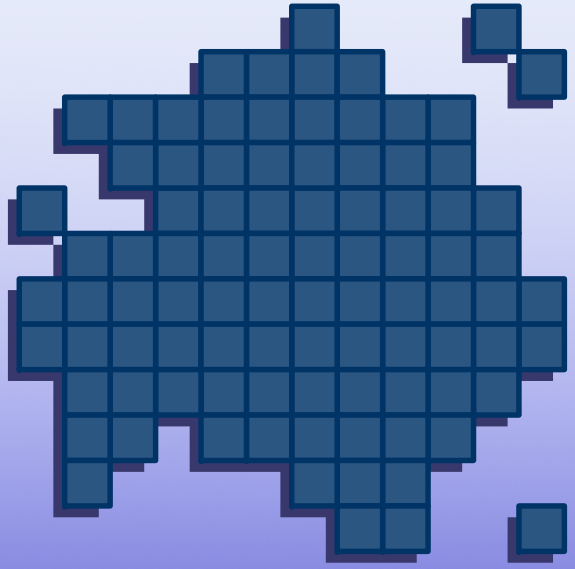
- Pre-fetch,Cache,Wait State的"全手動"設定模式範例:

  SYSTEMConfig(6000000L , SYS_CFG_WAIT_STATES | SYS_CFG_PCACHE);

  // SYSTEMConfig(unsigned int sys_clock, unsigned int flags);
  // sys_clock: 系統頻率(System Clock)
  // Flags: SYS_CFG_WAIT_STATES:依系統頻率設定Flash的Wait State。
  // SYS_CFG_PB_BUS:設定 PB Clock = System Clock。
  // SYS_CFG_PCACHE:啟用 Pre-fetch,Cache。
  // SYS_CFG_ALL:啟用以上三項設定。

- 詳細說明請參閱

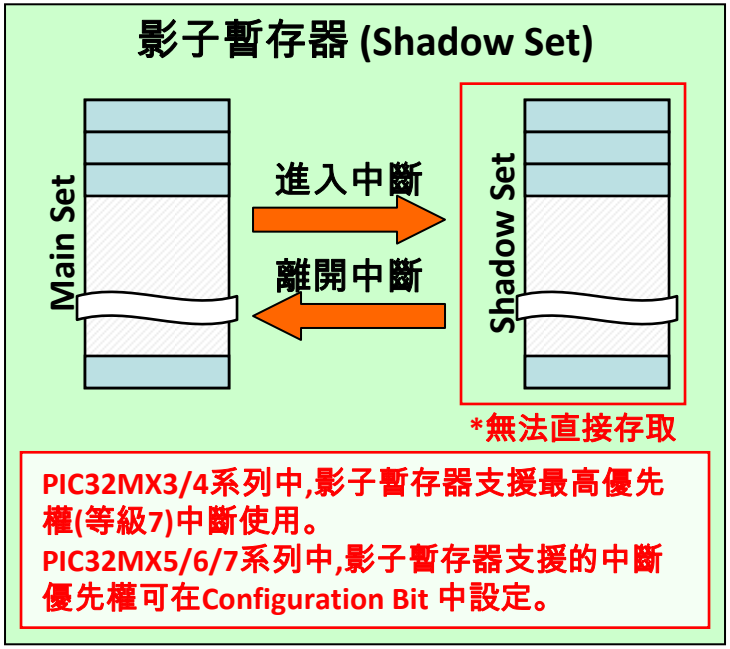
  32-bit-Peripheral-Library-Guide.pdf,Sec.2 System Functions或
  Microchip-PIC32MX-Peripheral-Library.chm。



Interrupt Architecture

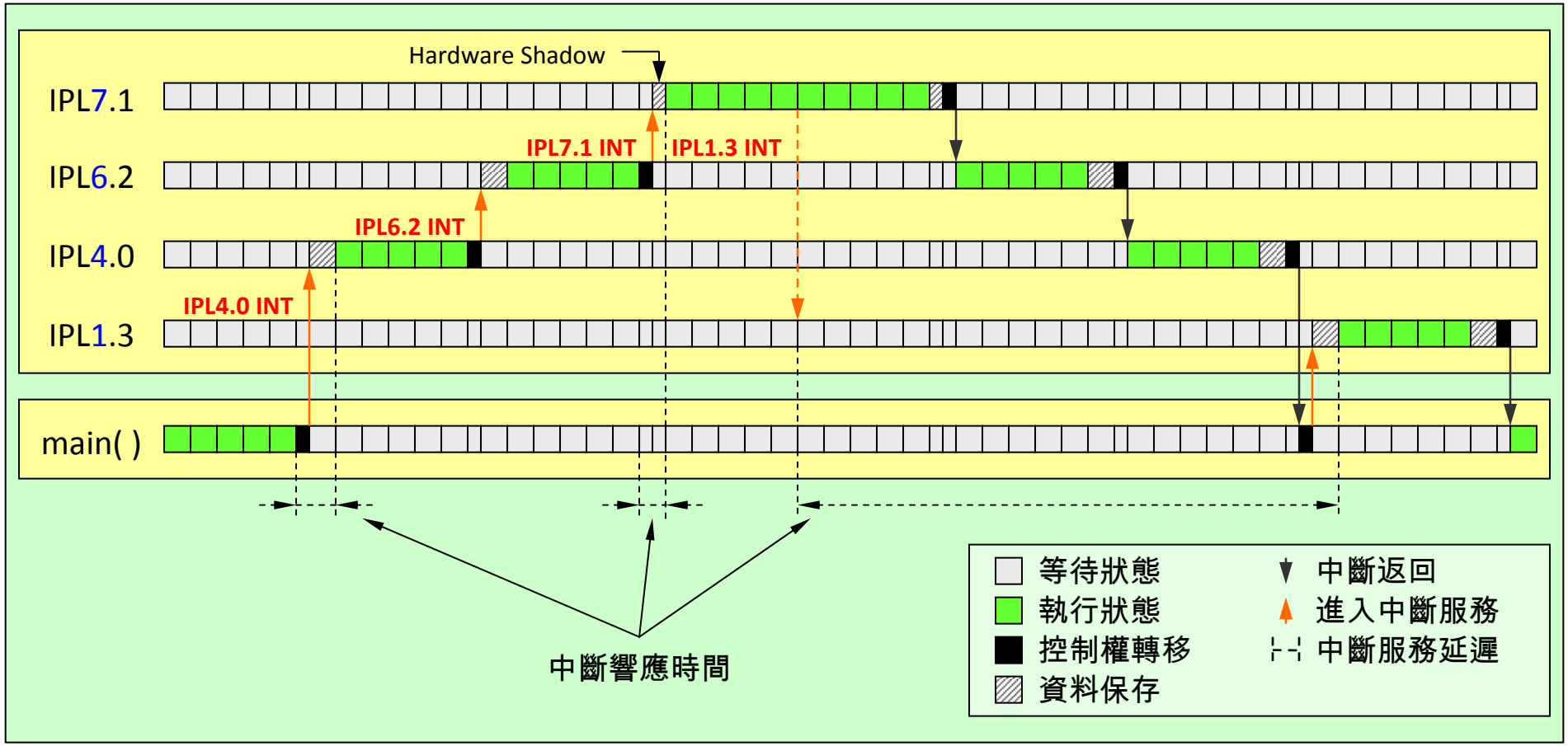
Interrupt Architecture

- PIC32具強健的中斷硬體架構,內建向量中斷控制器,可提升中斷效能。
- 支援兩種中斷模式:
  單一中斷向量模式(Single vector)
  (PIC32重置時預設值,通常使用在RTOS的架構上)
  多重中斷模式(Multi Vector)
- 提供96個中斷來源,64個中斷向量。
- 提供7個主要優先權(Priorities),4個次要優先權(Sub- Priorities),及自然優先權。
- 對於高優先權之中斷,提供硬體的影子暫存器集(Shadow Set),加快中斷響應時間。



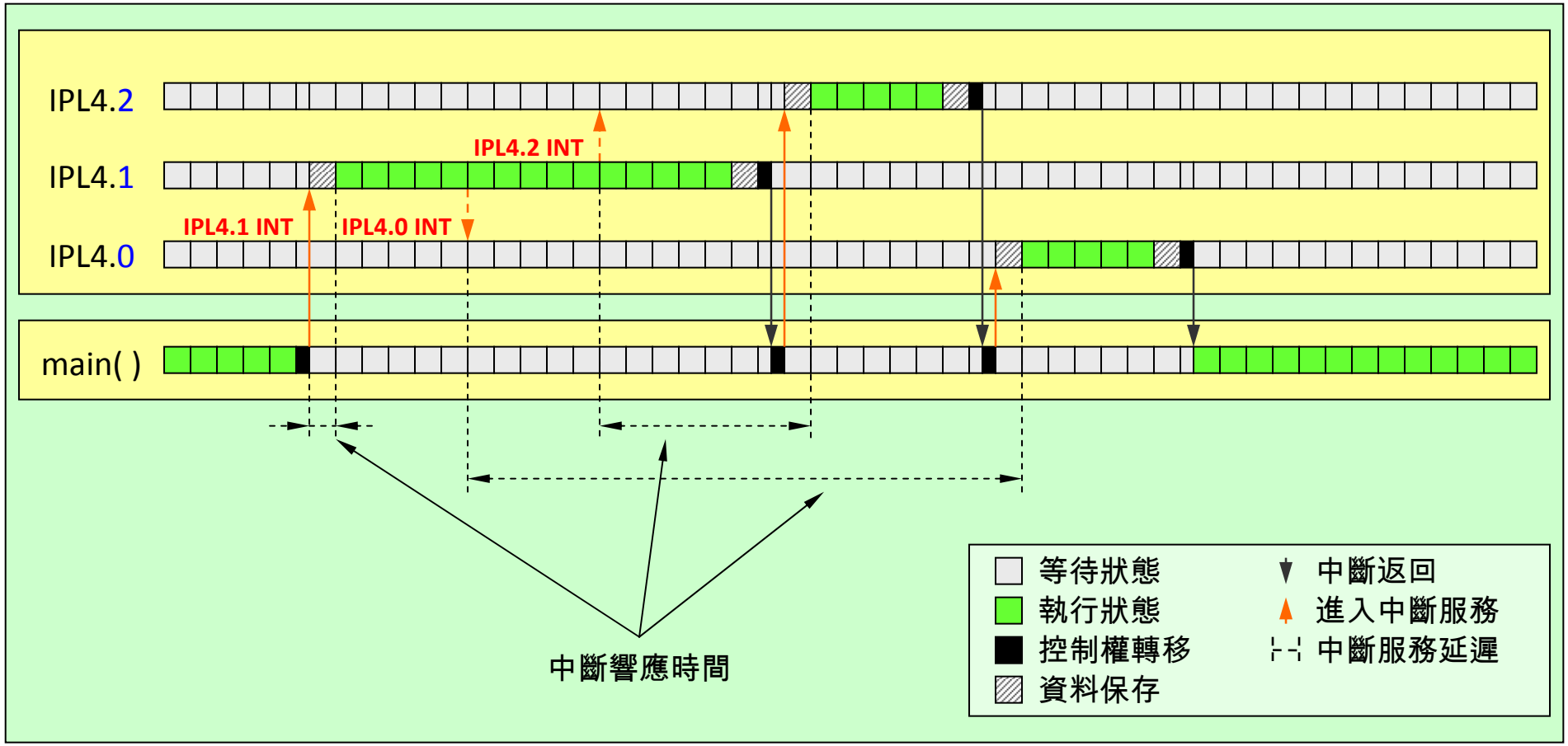
主要優先權的工作方式

主要優先權較高者可強佔(Preemptive)執行權



次要優先權的工作方式

中斷執行中不予理會，返回後再執行優先權最高者。



MICROCHIP
Regional Training Centers

MPLAB C32 的中斷支援

- 常用的C32中斷函數

  INTEnableSystemMultiVectoredInt( );
  INTEnableSystemSingleVectoredInt( );
  INTDisableInterrupts( );
  INTEnableInterrupts( );
  INTClearFlag( );
  INTSetFlag( );
  INTGetFlag( );
  INTSetPriority( );
  INTSetSubPriority( );

  ...

- 詳細說明請參閱

  32-bit-Peripheral-Library-Guide.pdf,Sec.7 Interrupt Functions。

MICROCHIP
Regional Training Centers

中斷設定流程

- 基本的中斷功能開啟必須具備以下動作(3+1):
  1.啟動多重中斷向量模式
    INTEnableSystemMultiVectoredInt( );
  2.定義中斷服務常式與原型。
    void __ISR(vector , ipl*n*) ISR_Function_Name(void);
  3.設定中斷主要與次要優先權並致能中斷。
    ConfigIntxxxx( );
  4.設定中斷來源的工作模式, 功能等。
    Openxxxx( );
- 詳細說明請參閱
  32-bit-Peripheral-Library-Guide.pdf。

MICROCHIP
Regional Training Centers

中斷服務常式

- C32 使用關鍵字 __ISR(,) 宣告為該函數為中斷服務常式。
  void __ISR(vector , ipl*n*) ISR_Function_Name(void)
  - vector:中斷向量編號,定義在對應的MCU標頭檔中。
    C:\Program Files\Microchip\MPLAB C32\pic32mx\include\proc\xxxx.h
  - ipl*n*:中斷優先權,(ipl1~ ipl7,ipl0為關閉中斷) 。
    *ISR所設定之ipl*n*必須與中斷本身的Priority相同(參考範例) 。
  - ISR_Function_Name:為中斷函數名稱。
- __ISR的詳細說明可參考
  C:\Program Files\Microchip\MPLABC32\vx.x\doc\
  MPLAB C32 User Guide.pdf

MICROCHIP
Regional Training Centers

中斷向量定義範例

- PICMX795F512H.h

```
/* Vector Numbers */
                    /* Label */                         /* Number */
#define             _CORE_TIMER_VECTOR                  0
#define             _CORE_SOFTWARE_0_VECTOR             1
#define             _CORE_SOFTWARE_1_VECTOR             2
#define             _EXTERNAL_0_VECTOR                  3
#define             _TIMER_1_VECTOR                     4
#define             _INPUT_CAPTURE_1_VECTOR             5
#define             _OUTPUT_COMPARE_1_VECTOR            6
#define             _EXTERNAL_1_VECTOR                  7
...
```

MICROCHIP
Regional Training Centers

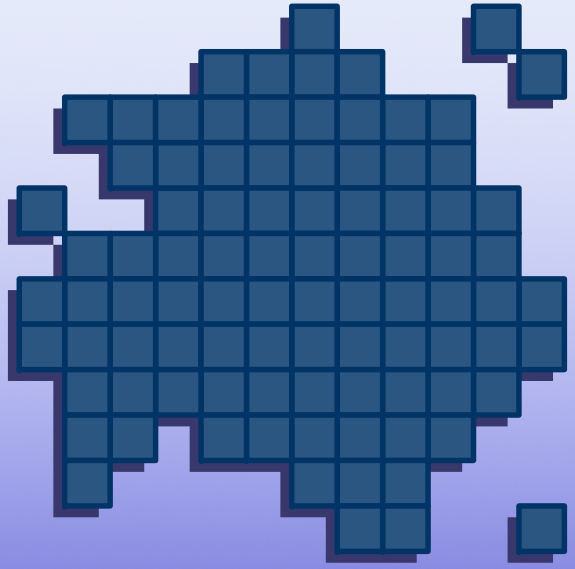
設定周邊中斷及優先權

- MPLAB C32 透過以下方式來設定周邊中斷的開啟/關閉及主副優先權。
  - ConfigIntxxxx (INT ON/OFF | INT_Priority | INT_Sub_Priority);
    參數一:開啟 / 關閉該周邊的中斷
    參數二:設定主要中斷優先權等級
    參數三:設定次要中斷優先權等級
  - INTSetVectorPriority(INT_UART_3A_VECTOR , INT_PRIORITY_LEVEL_5);
    // 設定主要中斷優先權等級。
    INTSetVectorSubPriority(INT_UART_3A_VECTOR ,
    INT_SUB_PRIORITY_LEVEL_0); // 設定次要中斷優先權等級。
    INTEnable(INT_U3ARX , INT_ENABLED); // 開啟周邊中斷。
- 詳細說明請參閱
  32-bit-Peripheral-Library-Guide.pdf,Sec.2 System Functions或
  Microchip-PIC32MX-Peripheral-Library.chm。

清除中斷旗標

- 進入ISR後,必須清除中斷旗標,MPLAB C32可透過以下方式清除中斷旗標:
  - 直接寫入暫存器方式:IFS0bits.T1IF = 0;
  - 使用C32的清除巨集:mxxxxClearIntFlag( );
    完整函數名稱請參考:32-bit-Peripheral-Library-Guide.pdf,Sec.2 System Functions或Microchip-PIC32MX-Peripheral-Library.chm。
  - 使用C32的清除函數:INTClearFlag(yyyy);
    yyyy為C32為該週邊所定義之列舉型別(INT_SOURCE)之成員。
    其定義可在32-bit-Peripheral-Library-Guide.pdf取得。
    (TABLE 8-1: INTERRUPT ENUMERATIONS)
    或pic32-lib-helpINT-PLIB-Help.chm
    中搜尋INT_SOURCE Enumeration)。

列舉名稱	周邊名稱
INT_INT0	External Interrupt 0
INT_T1	Timer 1 Interrupt
INT_IC1	Input Capture 1 Interrupt
INT_OC1	Output Compare 1 Interrupt
...	

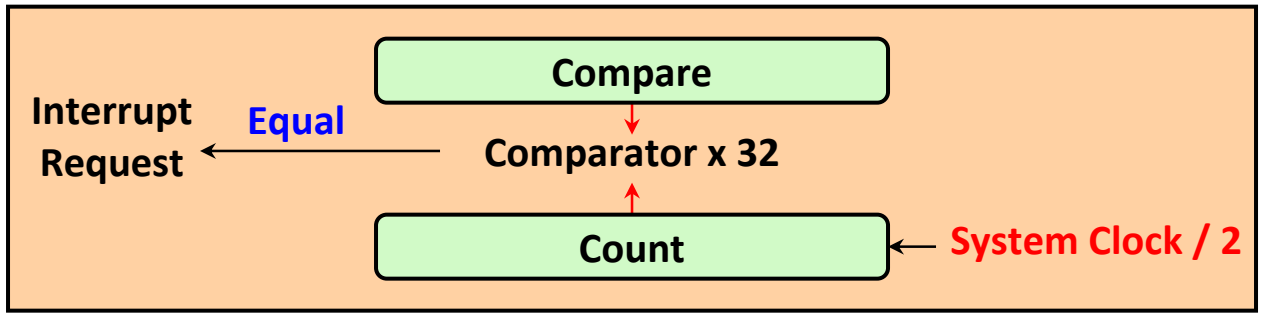


Core Timer and General Purpose Timer

MICROCHIP
Regional Training Centers

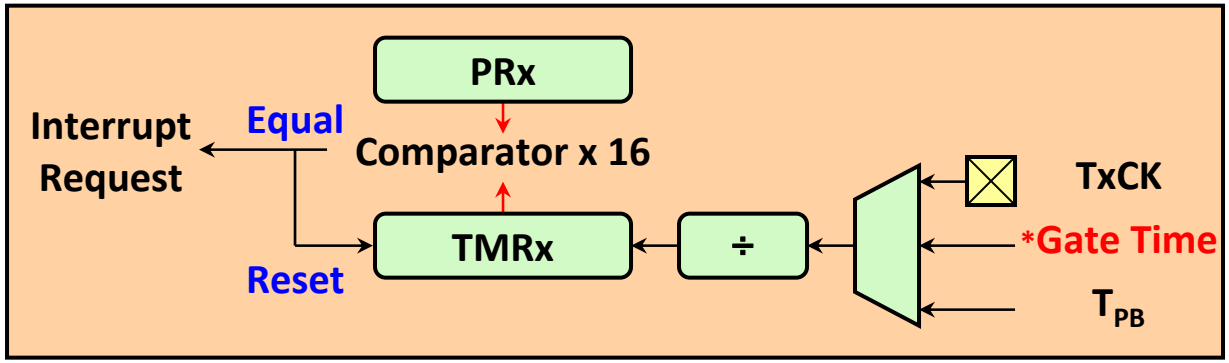
Core Timers

- Core Timer是PIC32中,由CP0直接控制的32-Bit Timer。使用System Clock為時脈來源,與其他Timer則是使用PB Clock不同。
- Core Timer的架構非常簡單,擁有兩個功能暫存器Count與Compare。Count固定每兩個System Clock就加1,沒有任何預除器/後除器。Count會累加到與Compare暫存器match時,發出中斷。
- 要注意的是Core Timer並無自動歸零功能,因此match後,必須以軟體的方式處理Count與Compare之間的關係。



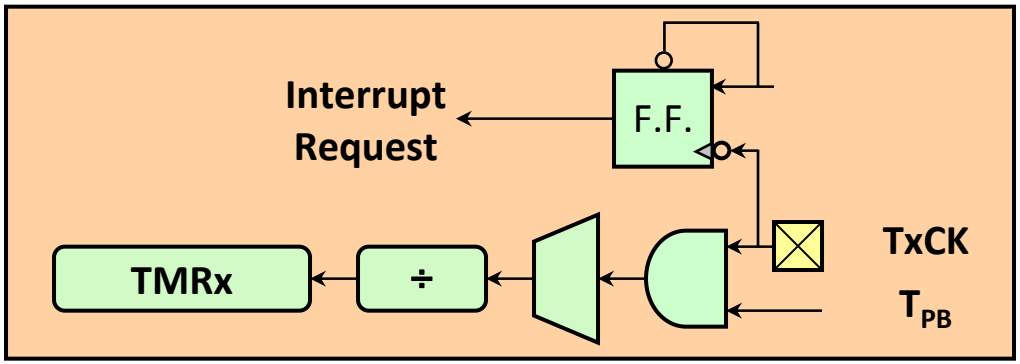
General Purpose Timers

- PIC32的General Purpose Timers結構與16-Bits MCU相同。
- Clock可以選擇內部,或者由外部接腳輸入。經過預除器後,送入TMRx。TMRx從”0”開始遞增,Compare會不斷比較PRx與TMRx的值,相同時發出中斷需求,並且將TMRx的值歸零。
- PIC32具有五個General Purpose Timers。Timer1支援非同步功能,可在Idle模式下,繼續運作。Timer2~5 強制與系統時脈同步,在Idle模式,因System Clock停止, 所以無法運作。

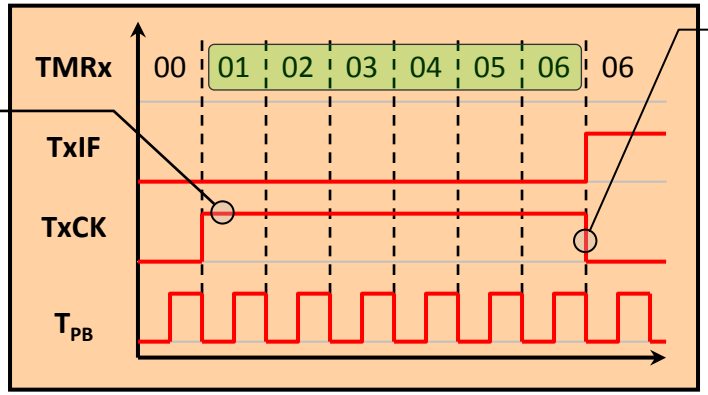


Gate Time Function

- PIC32的General Purpose Timers另外支援閘控計時(Gate Time)的特殊功能。可以用來計數外部輸入訊號的寬度。
- TxCK的輸入為 High時,Tcy可以被Timer計數,一直持續到TxCK的輸入由High變Low時停止。同時設定中斷旗標,發出中斷需求。



TxCK為High,AND Gate動作,Tcy可進入TMRx。

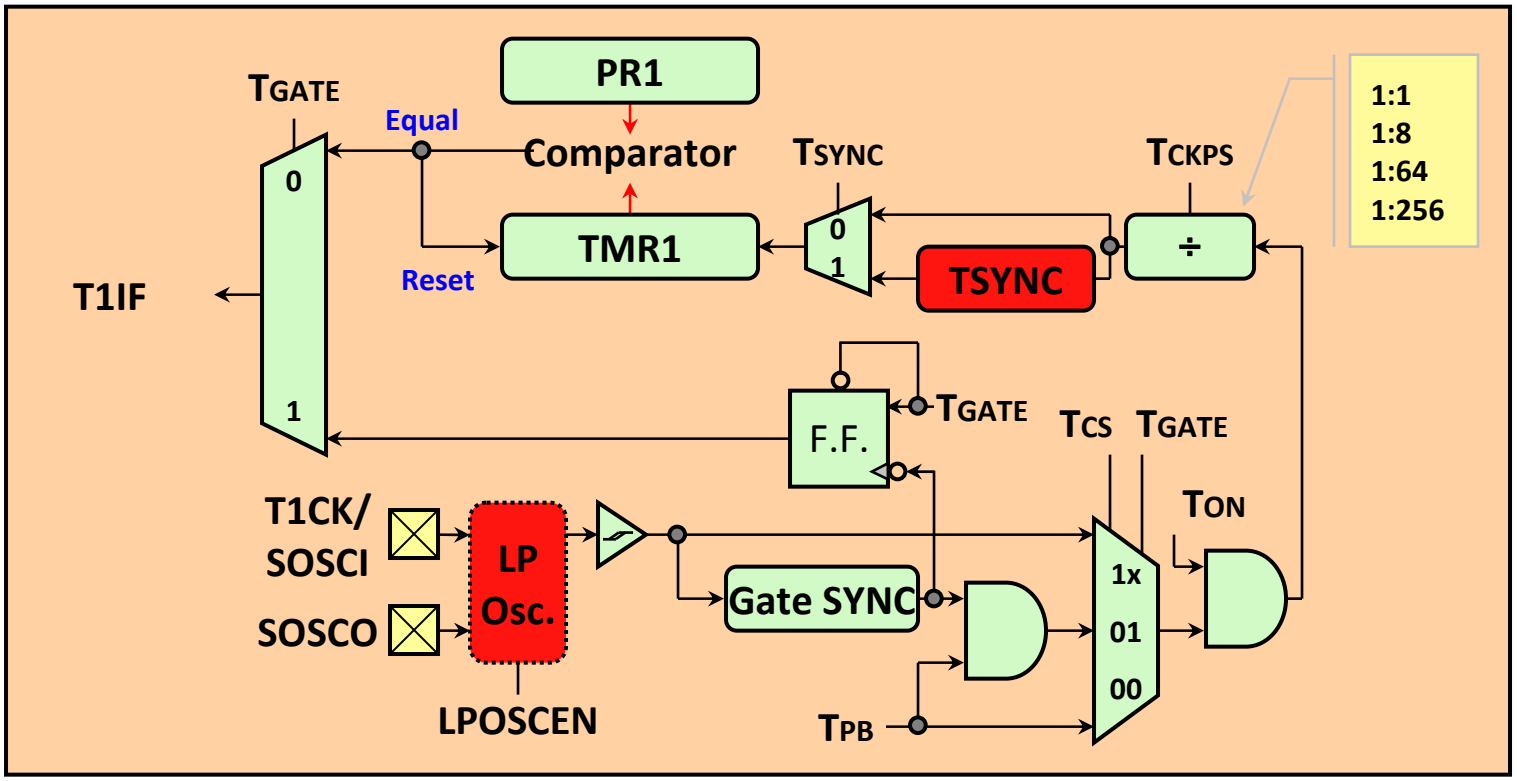


負緣觸發正反器,設定中斷旗標,發出中斷需求。

MICROCHIP
Regional Training Centers

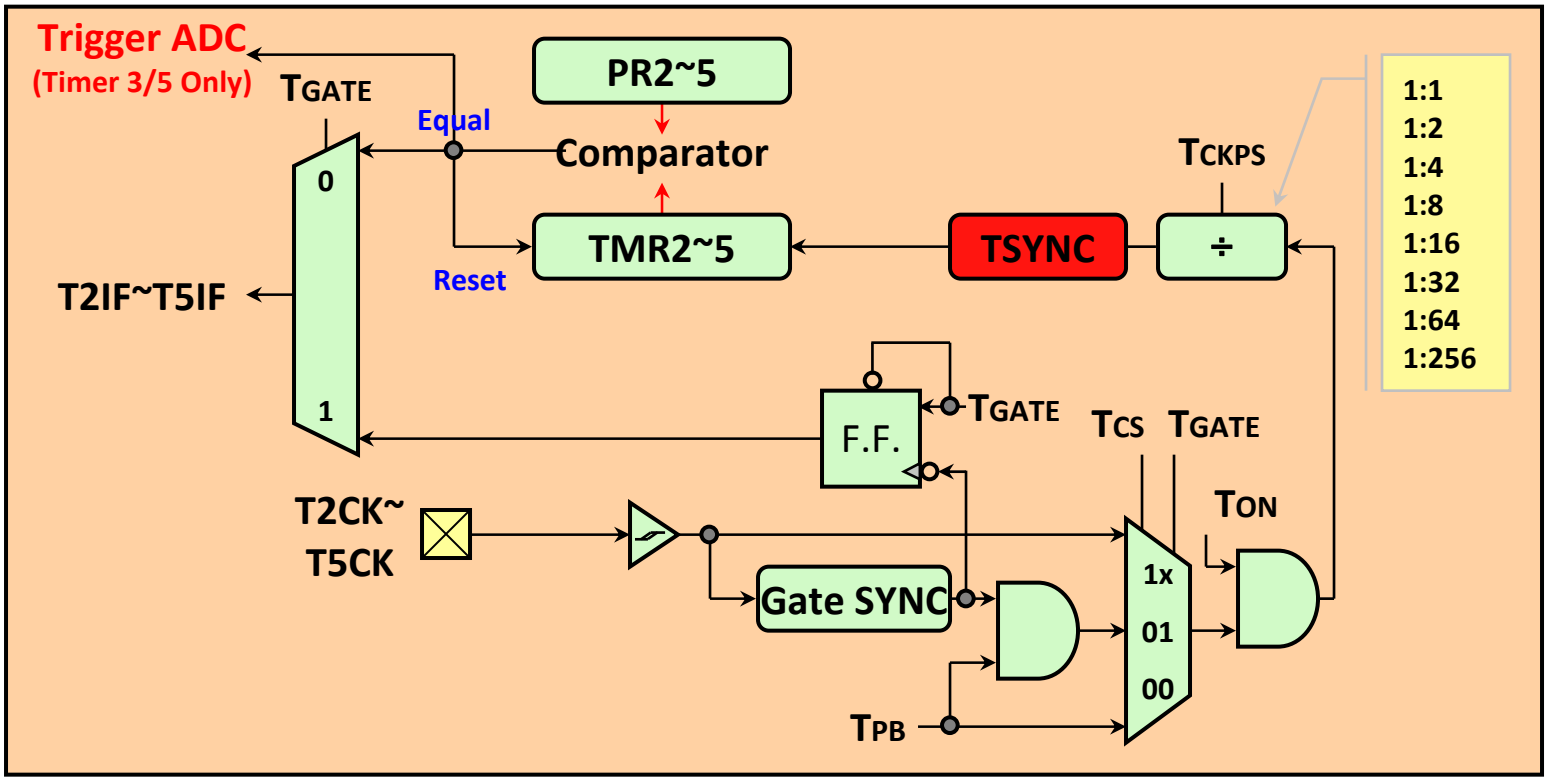
Type A Timer (Timer1)

- Timer1方塊圖如下。Timer1最大的特點,是可選擇同步與否,與內建晶體振盪電路,可以直接連接外部的Low Power Crystal, 如:32.768K Hz,做RTCC。



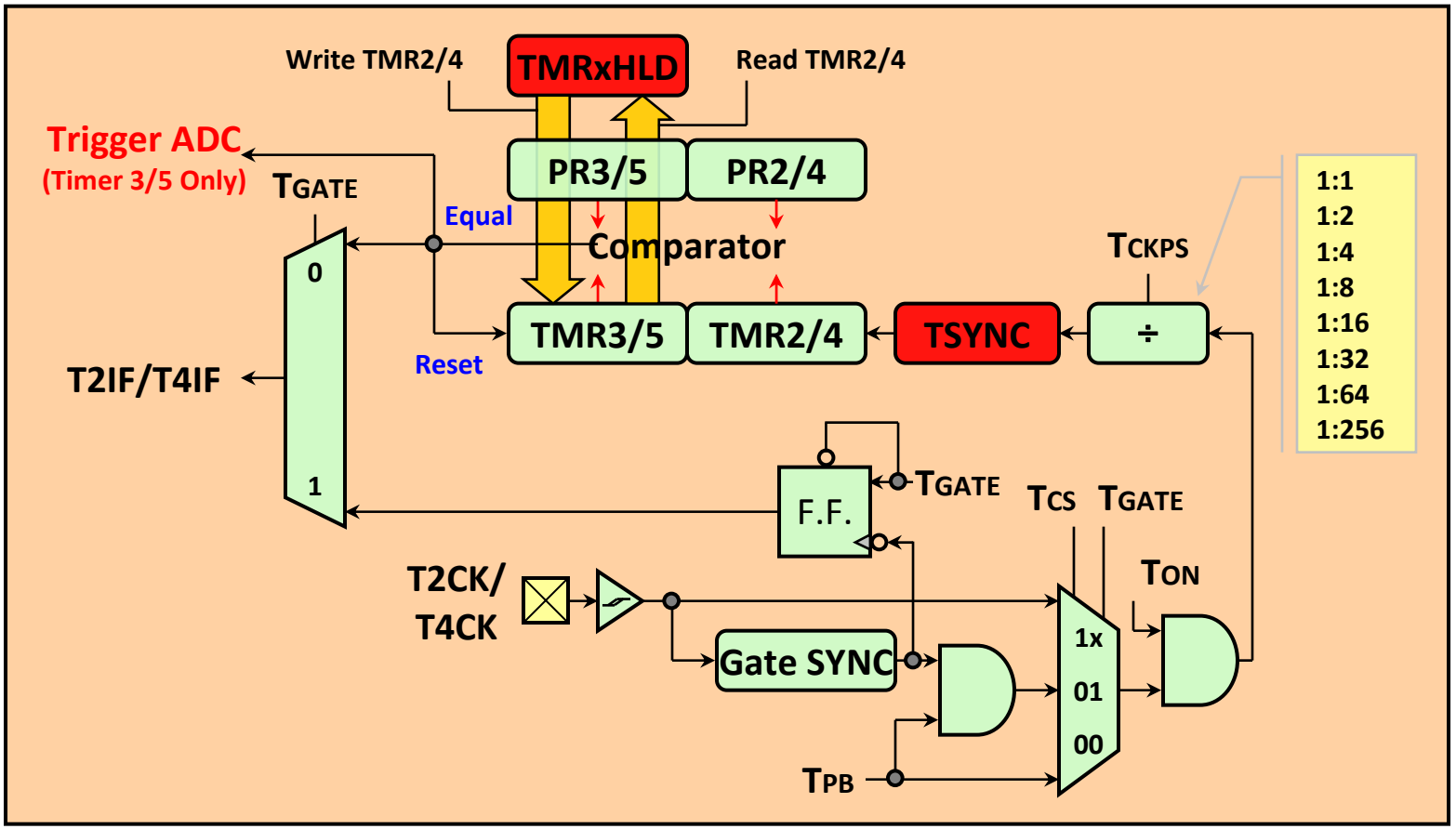
Type B Timer (Timer2~5)

- Timer2~5方塊圖如下。Timer2~5強制與系統時脈同步, 在Idle模式下無法工作。如果要計數外部訊號時,必須連接有明確正負緣狀態的方波,不可以直接連接Crystal。



32-Bits Timer

- Timer23, Timer45可以組合成32-Bits的Timer。



MICROCHIP
Regional Training Centers

C32 對 Core Timer 的支援

- 常用的C32 Core Timer函數

```
OpenCoreTimer( );                 // 初始化,載入新的Period 到 Compare。
UpdateCoreTimer( );               // Compare = 目前的計數值 + Period值。
mConfigIntCoreTimer( );           // 設定Core Timer中斷開/關與優先權。
mEnableIntCoreTimer( );           // 開啟Core Timer。
mDisableIntCoreTimer( );          // 關閉Core Timer。
mSetPriorityIntCoreTimer( );      // 設定Core Timer中斷優先權。
ReadCoreTimer( );                 // 讀取 Count。
WriteCoreTimer( );                // 寫入 Count。
```

- 詳細說明請參閱
  32-bit-Peripheral-Library-Guide.pdf,Sec.11 Timer Functions或
  Microchip-PIC32MX-Peripheral-Library.chm。

MICROCHIP
Regional Training Centers

C32 對 G.P. Timers 的支援

- 常用的General Purpose Timers函數

```
OpenTimerx( );          // 啟用Timerx,設定Timerx工作模式。
ReadTimerx( );          // 讀取TMRx。
WriteTimerx( );         // 寫入TMRx。
CloseTimerx( );         // 關閉Timerx。
ConfigIntTimerx( );     // 致能Timerx的中斷,並設定中斷優先權。
```

- 32-Bits Timer模式的Function

```
OpenTimerxy( );         // 啟用32-Bits Timerxy,設定Timer工作模式。
ReadTimerxy( );         // 讀取32-Bits Timerxy的TMRx。
WriteTimerxy( );        // 寫入32-Bits Timerxy的TMRx。
CloseTimerxy( );        // 關閉32-Bits Timerxy。
```

- 詳細說明請參閱
  32-bit-Peripheral-Library-Guide.pdf,Sec.11 Timer Functions或
  Microchip-PIC32MX-Peripheral-Library.chm。

MICROCHIP
Regional Training Centers

C32 Core Timer Code Example

```
// 建立中斷服務常式,服務常式優先權必須與中斷向量的優先權相同。
void __ISR( _CORE_TIMER_VECTOR , ipl3 ) ISR_CoreTimer( void )
{
    // 清除Core Timer中斷旗標。
    mCTClearIntFlag();
    // 更新Core Timer之Period。
    UpdateCoreTimer( 1000 );
}
int main ( void )
{
    // 使用C32 Interrupt Function啟用多重中斷向量模式。
    INTEnableSystemMultiVectoredInt( );
    // 使用C32 Timer Function啟用Core Timer，並指定其中斷優先權。
    mConfigIntCoreTimer ( CT_INT_ON | CT_INT_PRIOR_3 );
    // 使用C32 Timer Function進行設定。
    OpenCoreTimer(  1000 );
    while( 1 );
}
```

C32 G.P. Timers Code Example

```
// 建立中斷服務常式,服務常式優先權必須與中斷向量的優先權相同。
void __ISR (_TIMER_1_VECTOR, ipl4 ) ISR_Timer1( void )
{
    // 清除Timer中斷旗標。
    mT1ClearIntFlag( );
}
int main ( void )
{
    // 使用C32 Interrupt Function啟用多重中斷向量模式。
    INTEnableSystemMultiVectoredInt( );
    // 使用C32 Timer Function啟用Timer，並指定其中斷優先權。
    ConfigIntTimer1( T1_INT_ON | T1_INT_PRIOR_4 );
    // 使用C32 Timer Function進行設定。
    OpenTimer1( T1_ON | T1_IDLE_CON | T1_GATE_OFF | T1_PS_1_256 |
    T1_SOURCE_INT , 1000 );
    while( 1 );
}
```

Timer Period 計算

- 自己算太累了,請Compiler幫我們算。

```
#define SystemFrequency 60000000L
#define CoreTick ( (  SystemFrequency  / 2 ) CoreTogglesPerSec ) )
#define CoreTogglesPerSec 10
#define PBFrequency SystemFrequency / 1
#define Timer1Tick ( ( PBFrequency / 256 ) / Timer1TogglesPerSec )
#define Timer1TogglesPerSec 10
```

- SystemFrequency:系統頻率。
- SystemFrequency / 2:取得一秒鐘進入Core Timer的Clock個數。
  (SystemFrequency / 2) / n:一秒鐘有這麼多個Clock,那0.5秒就是/2, 0.1秒就是/10。
- PBFrequency / x:取得一秒鐘進入Timer1的Clock個數。(x:預除器的設定)
  (PBFrequency / x) / n:一秒鐘有這麼多個Clock,那0.5秒就是/2, 0.1秒就是/10。

MICROCHIP
Regional Training Centers

Lab2 – Timer (20 mins)

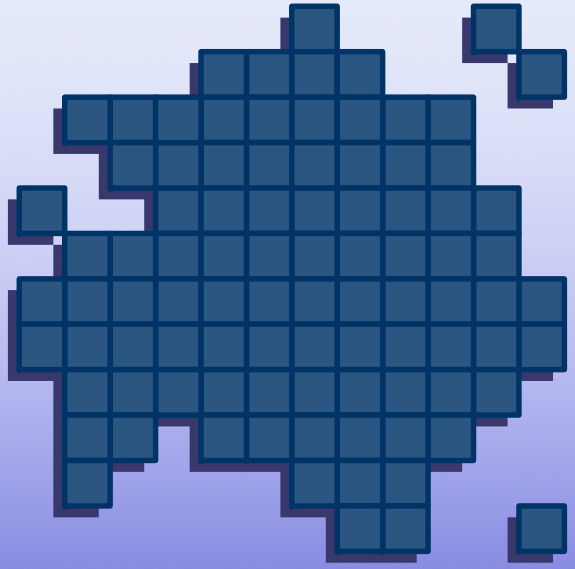
- 嘗試在Lab2的程式基礎上, 加入Core Timer跟Timer1的控制程式。
- 使用Core Timer跟Timer1搭配中斷機制, 取代Lab1中的軟體Delay功能。
- 利用C32所提供之IO Function控制RA0跟RA1, 讓RA0,跟RA1接腳可以不斷的轉態(Toggle, 1->0->1->…) 。 RA0 Toggle間隔為500 mS , RA1 Toggle間隔為1S 。
- Timer1時脈來源設定為內部,除頻器設定為1:256,Gate Time功能關閉, Idle Mode下繼續執行。
- 注意Core Timer match時, 並不會自動清除Count, 因此必須透過軟體的方式自行處置Count 。

MICROCHIP
Regional Training Centers

Discuss

- 比照Lab1, 先將程式中SYSTEMConfig( )改為註解。
- 編譯後重新使用軟體模擬觀察軟體模擬的時間, 應該仍然是500mS, 1S。
- 接著再編譯,燒錄,執行狀況有何變化?

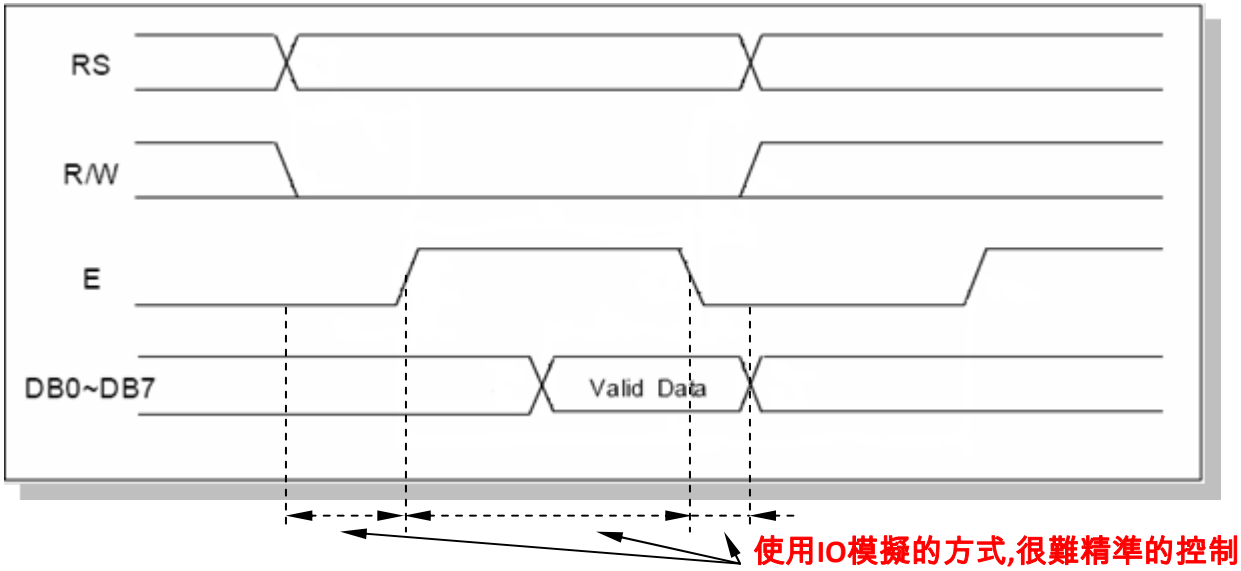
- 原本是的變化時間與現在無差別,為何如此?
- Pre-fetch跟Cache功能沒有開啟,Wait State恢復到預設值，影響的是CPU執行與抓取資料,指令的時間,並不會影響時脈。因此依時脈來運作的周邊,不受影響。



Parallel Master Port

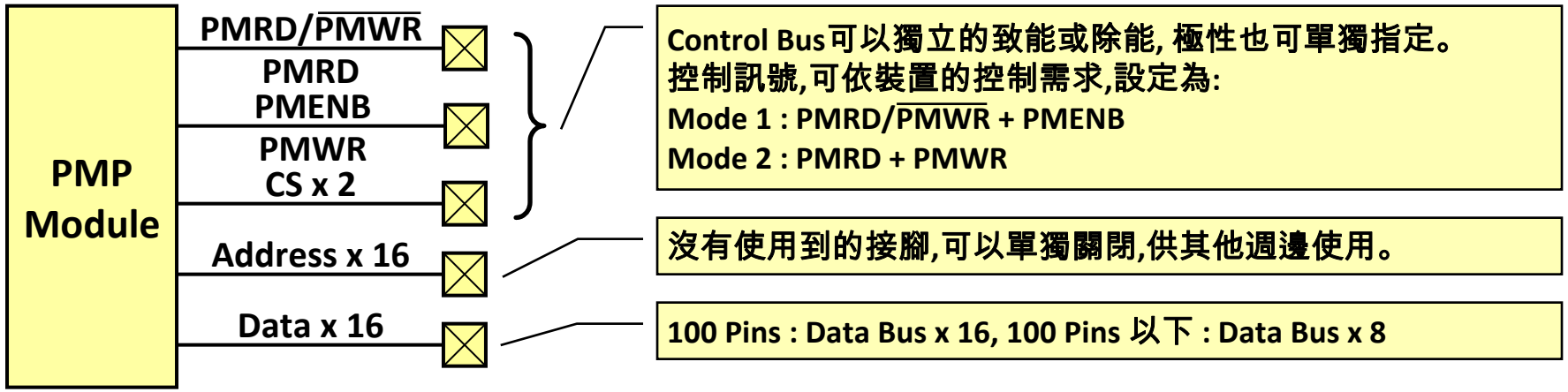
What's PMP ?

- Parallel Master Port (PMP), 並列周邊存取模組。
- PMP可以使用來存取外部的並列周邊,例如:並列介面的記憶體,記憶卡或LCD Module等裝置。
- 古老的方式,都是使用IO來存取並列裝置,執行效率低落,對於時序的控制也很難處理,一旦改變系統時脈,程式必須大翻修。
- 透過PMP, 可以自動的控制匯流排動作, 時序的需求也只需要在初始化時指定即可。可以大幅減低CPU的負擔。



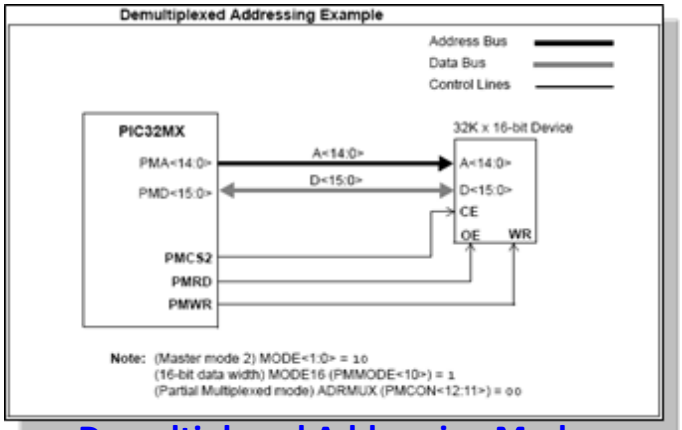
PMP Introduction

- PIC32的PMP結構與16-Bits MCU幾乎相同。提供Control Bus, Address Bus(2^{16}), Data Bus(2^{16}) 。
- Control Bus可以獨立的致能或除能, 極性也可單獨指定。提供最多兩個Chip Select Pin(與Address Bus A_{14}, A_{15}共用)。
- Address Bus共有16條,最高定址範圍為2^{16} 。沒有使用到的接腳,可以單獨關閉,供其他週邊使用。提供最多16條的Data Bus(100 Pins Only, 100 Pins以下8條)。

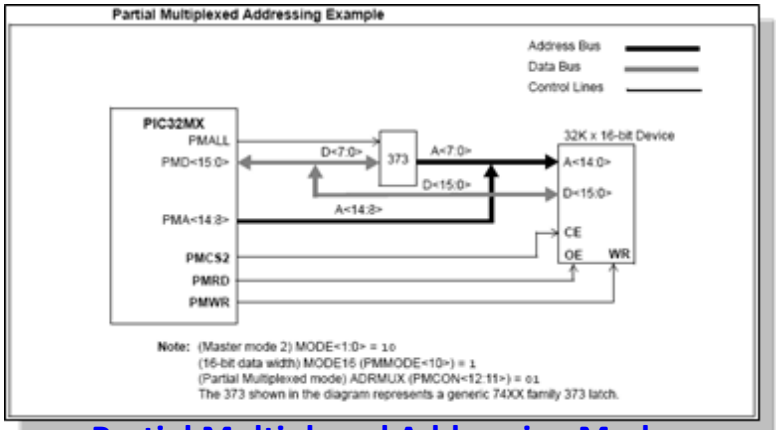


PMP Address Multiplex

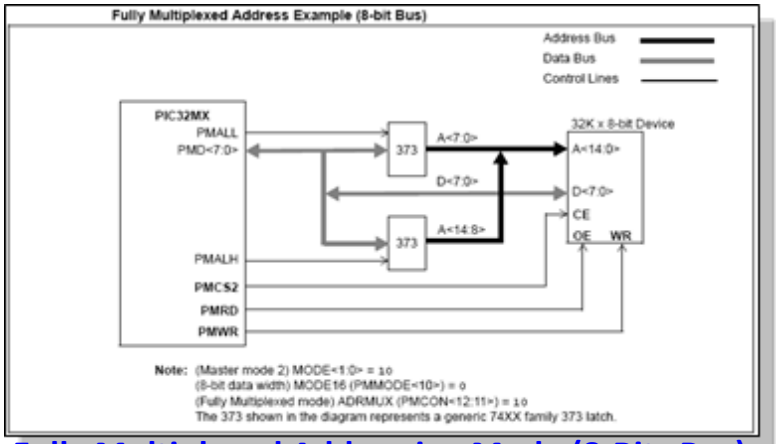
- PIC32支援多種Address/Data多工模式。



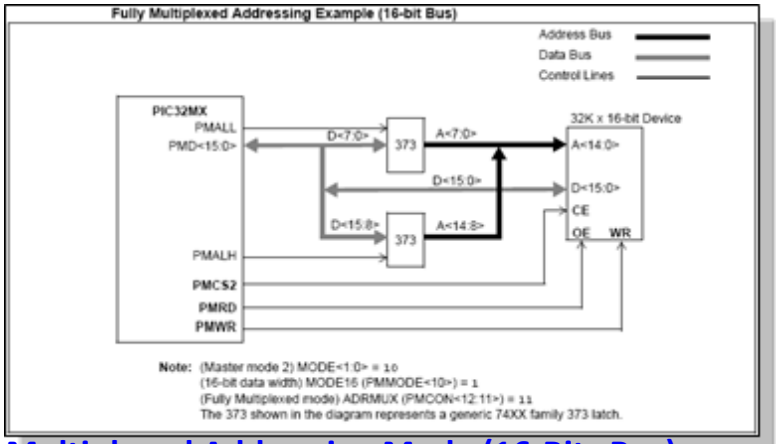
Demultiplexed Addressing Mode



Partial Multiplexed Addressing Mode



Fully Multiplexed Addressing Mode (8-Bits Bus)

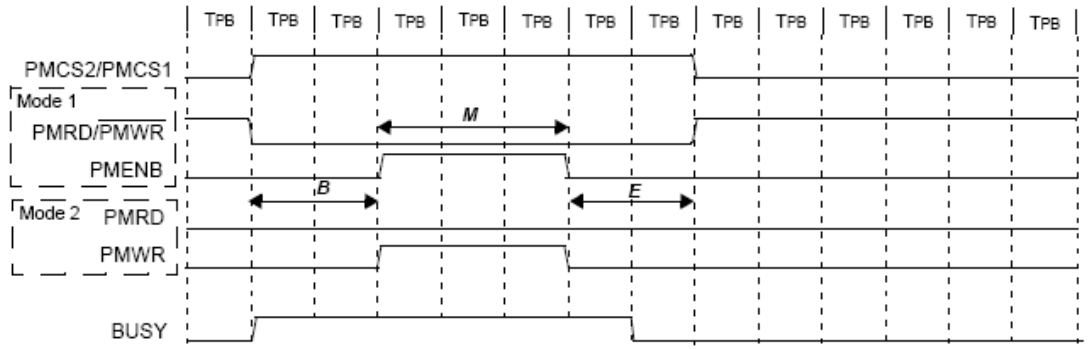


Fully Multiplexed Addressing Mode (16-Bits Bus) 100 Pins Only

Wait State

- 針對低速周邊,可以在時序中加入Wait State,調整存取時序。
- 時序的前,中及後均可調整插入Wait State的數量。前及後最多可加入4個Wait State,中間最多15個。
- 1 Wait State = 1 PB Clock。

Figure 13-16: 8-bit, 16-bit Write Operations, ADRMUX = 00, Wait States Enabled



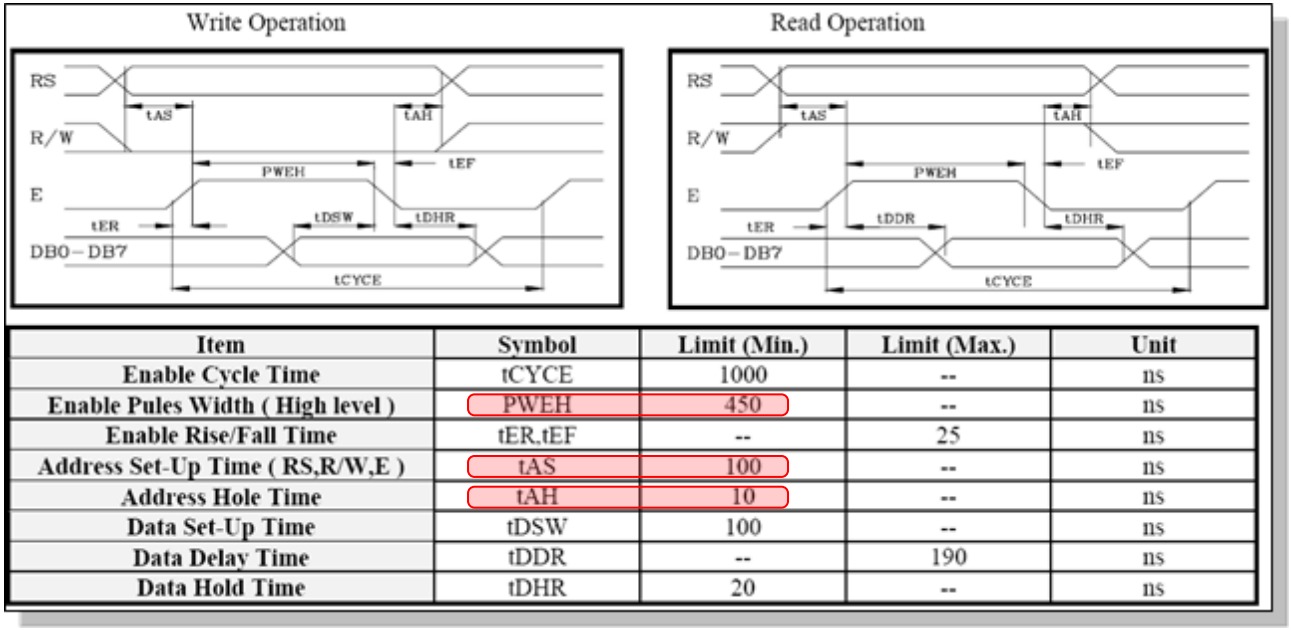
Legend:

B = WAITB<1:0> = 01 (2 Wait states)
M = WAITM<3:0> = 0010 (3 Wait states)
E = WAITE<1:0> = 01 (2 Wait states)

Note: If WAITM<3:0> = 0000, M is forced to 1 TPBCLK, WAITB is ignored (B forced to 1 TPBCLK), and WAITE is ignored (E forced to 1 TPBCLK).

Wait State for LCD Module

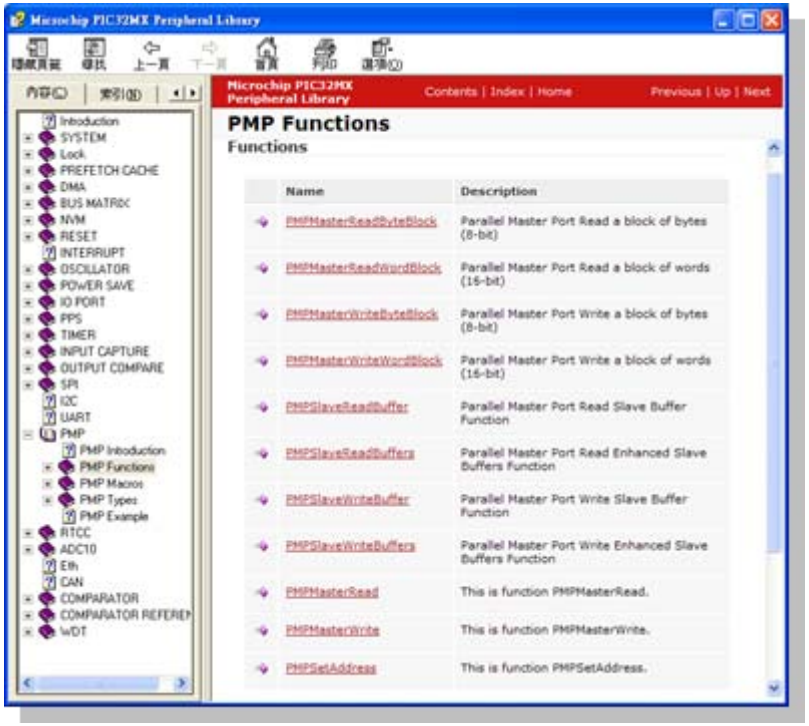
- 以LCD Module為例,在60MHz的PB Clock下,則前中及後的Wait State至少必須設定為2,8,1才能滿足規範(不同LCD規範不同,看Datasheet最準)。
- B : 0.1 ns / (1 / 60Mz) = 1.6667 -> 2
  M : 0.45 nS / (1 / 60Mz) = 7.5 -> 8
  E : 0.01 ns / (1 / 60Mz) = 0.1667 -> 1



Item	Symbol	Limit (Min.)	Limit (Max.)	Unit
Enable Cycle Time	tCYCE	1000	--	ns
Enable Pules Width (High level)	PWEH	450	--	ns
Enable Rise/Fall Time	tER,tEF	--	25	ns
Address Set-Up Time (RS,R/W,E)	tAS	100	--	ns
Address Hole Time	tAH	10	--	ns
Data Set-Up Time	tDSW	100	--	ns
Data Delay Time	tDDR	--	190	ns
Data Hold Time	tDHR	20	--	ns

C32 PMP Code Example

- PMP初始化範例
  mPMPOpen(PMP_ON | PMP_MUX_OFF | PMP_READ_WRITE_EN |PMP_CS2_POL_LO |PMP_WRITE_POL_HI | PMP_READ_POL_HI | … , PMP_MODE_MASTER1 |… , PMP_WAIT_BEG_4 | PMP_WAIT_MID_15 | PMP_WAIT_END_4 , PMP_PEN_0 , PMP_INT_OFF | PMP_INT_PRI_0);
- PMP Read範例
  PMPSetAddress(*Addr*);
  Data = **mPMPMasterReadByte**( );
- PMP Write範例
  PMPSetAddress(*Addr*);
  PMPMasterWrite(*Data*);
- 詳細說明請參閱
  32-bit-Peripheral-Library-Guide.pdf, Sec.17 PMP Functions或 Microchip-PIC32MX-Peripheral-Library.chm。



PMP Read Operation

- 進行PMP Read操作時, 必須注意先進行一次Dummy Read, 以啟始Bus的活動, 等待PMP完成讀取後, 才能取得正確的資料。
  (參考RFMSect. 13 Parallel Master Port, 13.3.3說明)

13.3.3 Read Operation

To perform a read on the parallel bus, the user application reads the PMDIN register. The effect of reading the PMDIN register retrieves the current value and causes the PMP to activate the Chip Select lines and the address bus. The read line PMRD is strobed in Master mode 2, PMRD/PMWR and PMENB lines in Master mode 1, and the new data is latched into the PMDIN register making it available the next time the PMDIN register is read.

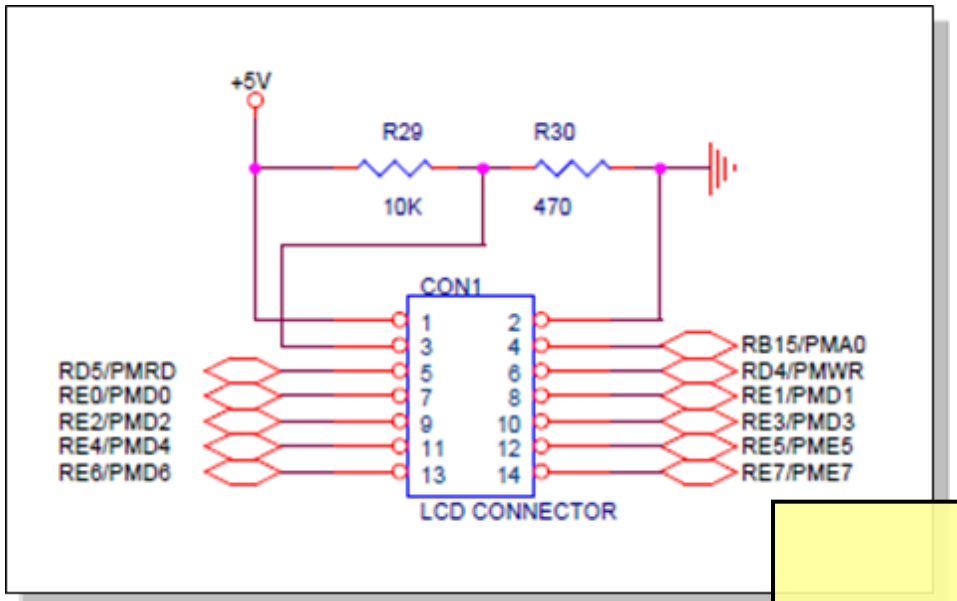
Note that the read data obtained from the PMDIN register is actually the read value from the previous read operation. Therefore, the first user application read will be a dummy read to initiate the first bus read and fill the read register. See Figure 13-11, which illustrates this sequence. Also, the requested read value will not be ready until after the BUSY bit is observed low. Thus, in a back-to-back read operation, the data read from the register will be the same for both reads. The next read of the register will yield the new value.

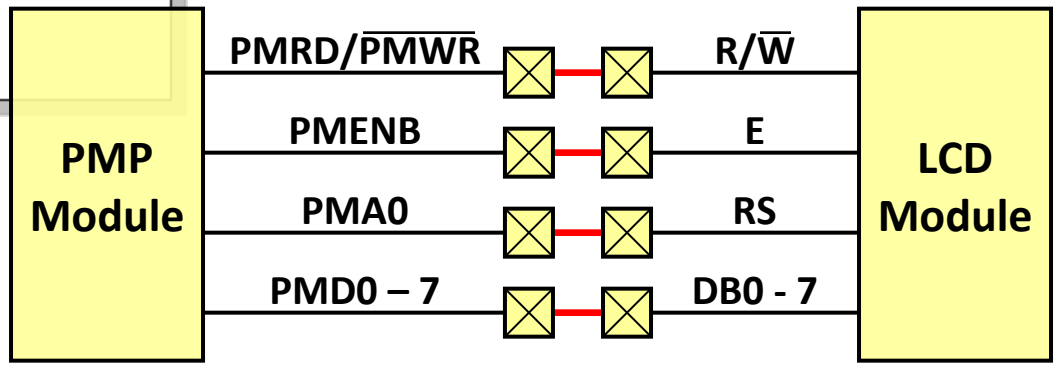
In 16-bit Data mode, PMMODE<MODE16> = 1, the read from the PMDIN register causes the data bus PMD<15:0> to be read into PMDIN<15:0>. In 8-bit mode, PMMODE<MODE16> = 0, the read from the PMDIN register causes the data bus PMD<7:0> to be read into PMDIN<7:0>. The upper 8 bits, PMD<15:8>, are ignored.

```
unsigned char Temporary;
...
  // Set Address
  PMPSetAddress( Addr );
 // Dummy Read for trigger PMP active
  Data = mPMPMasterReadByte( );
 // Check & Clear The PMPIF
  while( !INTGetFlag( INT_PMP ) );
  INTClearFlag( INT_PMP );
// Read Data from device
  Data = mPMPMasterReadByte( );
...
```

APP1632 LCD Module的連接

- APP1632上已經將LCD Module的信號接至PMP正確腳位,連接方式可以參考下圖。





MICROCHIP
Regional Training Centers

Lab3 – PMP (Demo)

- 嘗試在Lab3的程式基礎上,完成PMP的初始化設定,讓PMP Module可以正確的控制LCD Module。
- 正確設定完成後,可以LCD Module上看到預先設計好的文字。
- PMP存取模式設定為Mode1:**PMRD/PMWR** + **PMENB**。Read High Write Low, **PMENB** Active High。
- Wait State設定為B:4,M15,E:4(不確定裝置速度時,先開到最大,比較保險) 。
- Address Bus只有用到A_0, A_1~A_{14}可以Disable 。

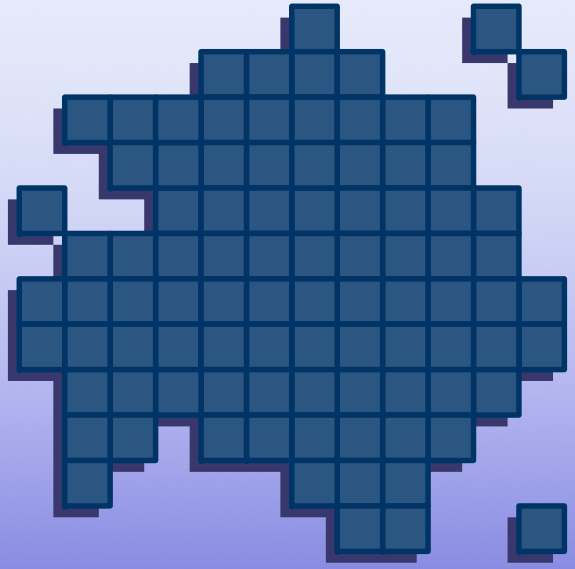
MICROCHIP
Regional Training Centers

Discuss

- LCD Module的Function已經替各位完成了,可以直接使用(APP1632_LCM.c)。
- 提供以下幾個Function:

```
void LCM_Init( )                                  // 初始化LCD Module。
int LCM_IsBusy( )                                 // 讀取LCD  Module的 BUSY Flag並傳回狀態。
                                                  // 1:Busy,0:Not Busy。
void LCM_PutASCII( unsigned char )                // 輸出字元。
Void LCM_SetCursor( char X , char Y )             // 設定遊標位置。
Void LCM_PutROMString( const unsigned char *String )// 輸出const 字串。
Void LCM_PutRAMString( unsigned char *String )    // 輸出字串。
Void LCM_PutHex( unsigned char Hex )              // 將變數轉為Hex輸出。
void LCM_PutNumber( unsigned int Number , unsigned char Digit );// 輸出整數。
```

- 如果需要其它的功能,可以自行修改程式碼。



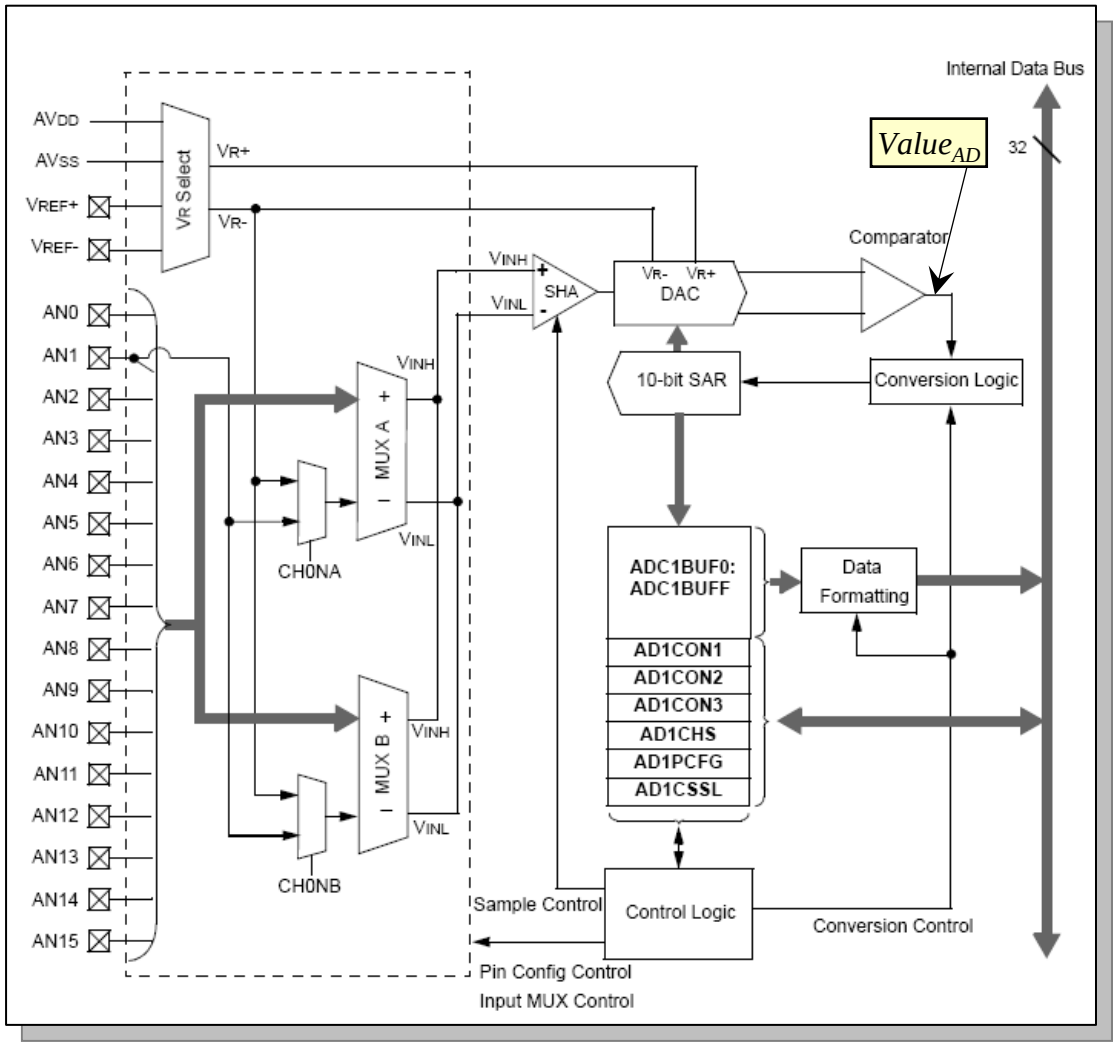
10-Bits ADC

PIC32's ADC Architecture

- PIC32MX具有一組採用SAR(連續近似法)的10-Bits ADC。搭配16對1之類比多工器,達成多通道轉換功能。
- 具兩組多工器,可交替使用,多工器A支援通道掃描轉換功能。
- 類比信號轉換結果為

$$Value_{AD} = \frac{V_{AD} - V_{R-}}{V_{R+} - V_{R-}} \times 2^n$$

- V_{R+}, V_{R-}可使用V_{REF+}, V_{REF-}或AV_{DD}, AV_{SS}。



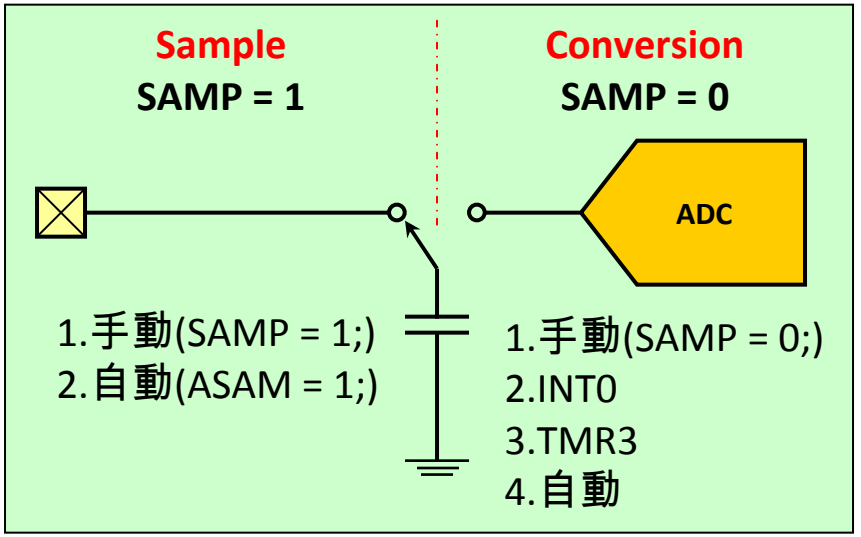
MICROCHIP
Regional Training Centers

PIC32's ADC Architecture

- ADC輸出格式共有八種不同格式(有/無號,浮點數/整數,32/16 Bits) 。
- ADC內建十六個32-Bit的Buffer(ADCxBUF0 ~ ADCxBUFF),用以儲存轉換所得之結果。ADC每次轉換後會將結果存入ADC的Buffer。存入後,Buffer的Index會自動累加,每次發出中斷需求後,Index會歸零。
- ADC可選擇經過多少次轉換後,發出中斷需求。
- ADC具有兩種觸發"取樣"的來源,可透過自動,手動。
- ADC具有多種觸發"轉換"的來源,可透過手動,自動,外部中斷或Timer3 Match觸發等。

ADC Sample, Conv. Trigger Source

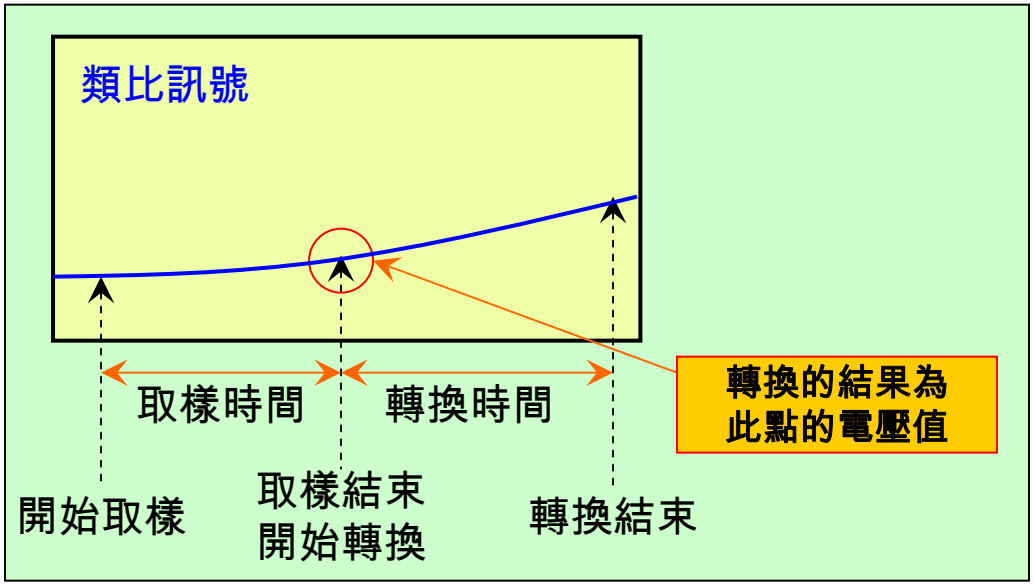
- ADC的轉換其實一點都不複雜,所有動作其實都圍繞在SAMP位元(A/D Sample Enable bit), SAMP=1時進行取樣, SAMP=0則進行轉換。
- 手動取樣:SAMP完全由軟體控制,硬體不干涉。



- 自動取樣:SAMP可由硬體控制,在每次轉換完成後,自動將SAMP設為1。(透過將ASAM設為1可達成)
- 要進入轉換狀態,將SAMP清除為0即可。
- PIC32MX提供手動(軟體清除),自動,外部中斷或Timer3 Match等方式,清除SAMP 。

ADC 取樣與轉換的概念

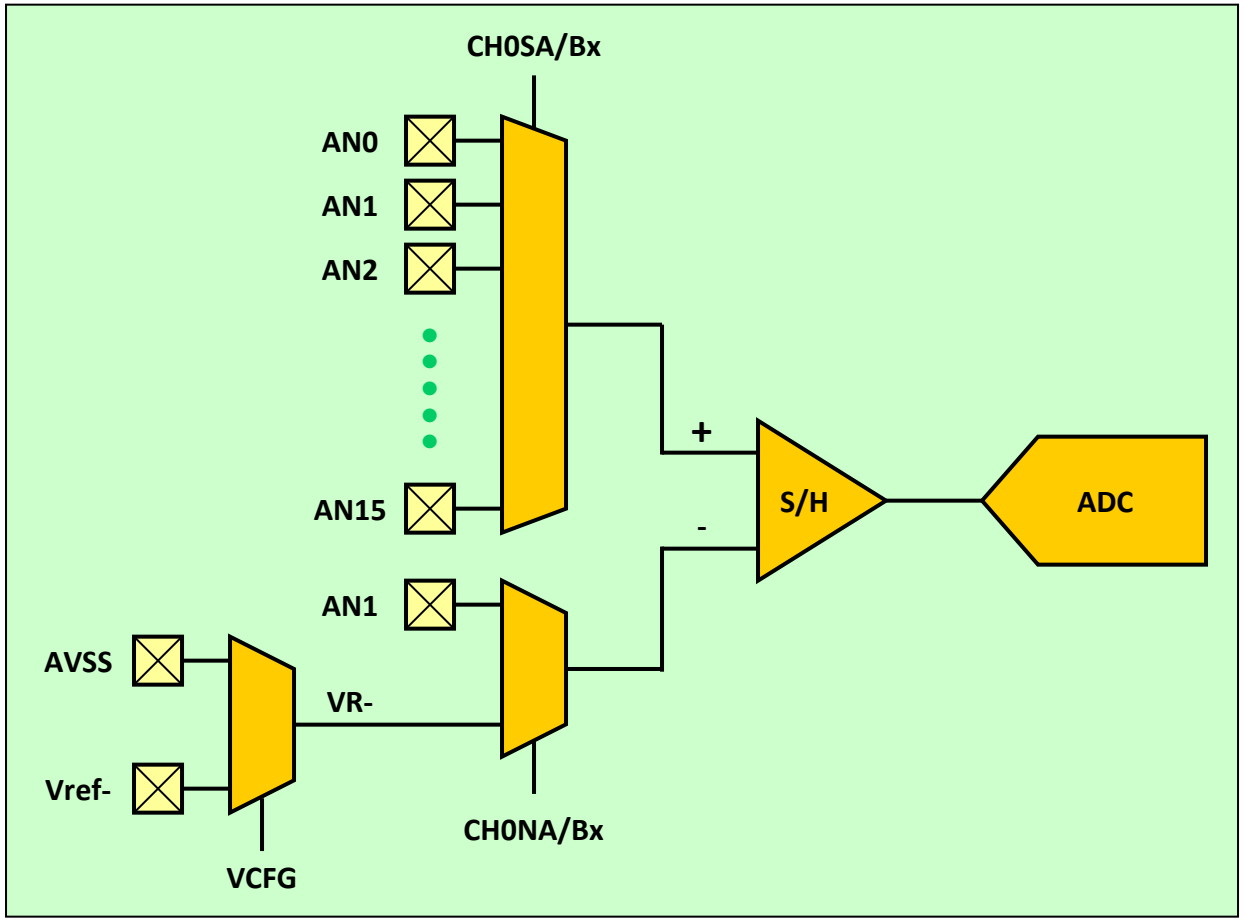
- SAR ADC作動分為"取樣"及"轉換"兩步驟,
  取樣時利用外部訊號對ADC內部的電容器充電,取得外部電壓值。
  轉換時依據取得的電壓值換算出結果。



- ADC的取樣時間與轉換時間都有最短需求時間的規範,設計時必須滿足才能確保轉換結果正確。時間規範可查詢Datasheet電氣特性章節。
- PIC32MX系列的取樣時間必須大於$1T_{AD}$(詳細規範請參考Datasheet),轉換時間通常需大於$12T_{AD}$($1\ T_{AD}$ =65nS) 。

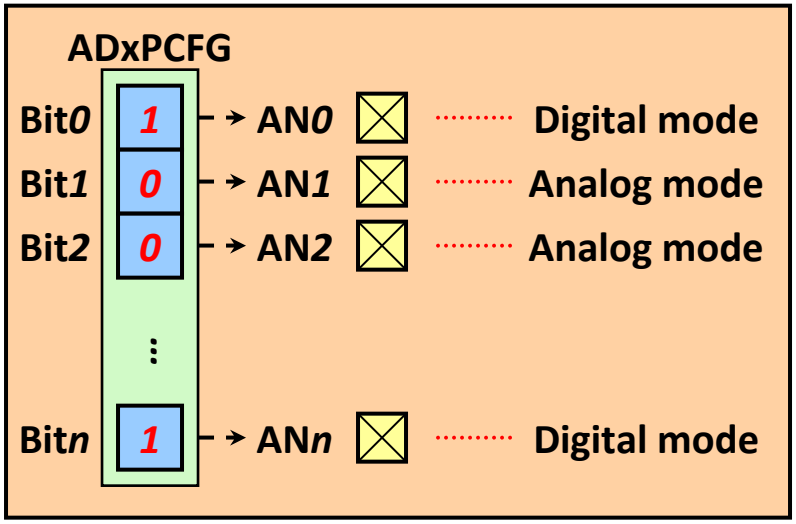
ADC Channel Select

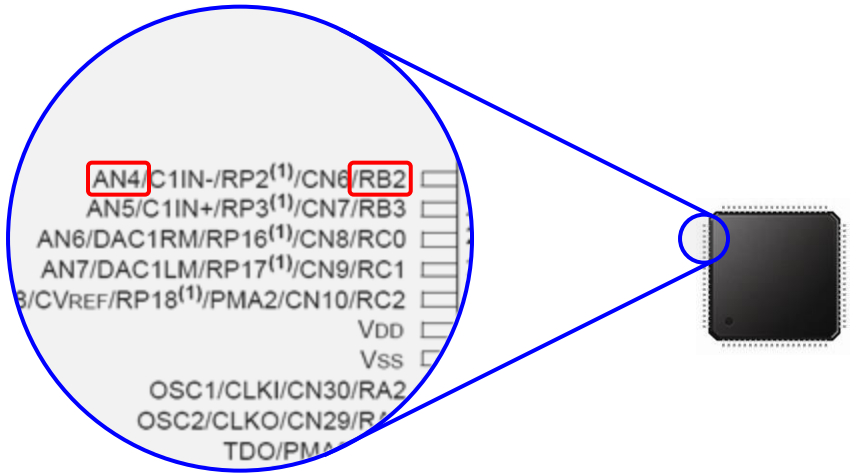
- ADC的正端輸入可以選擇AN0~AN15。負端輸入則可選擇連接VR-(V_{REF-}或AV_{SS})。
- 負端輸入也可選擇連接到AN1,讓兩訊號相減後,再送入ADC(單端差動模式)。
- 單端差動,類似”差動”的概念,但兩者有相同的地(GND)。



Analog or Digital Mode

- 有些 IO接腳(RX*n*)的功能跟類比輸入(AN*n*)共用的。但MCU在Reset/Power On後的預設值為Analog mode,無法做為數位輸出入使用。所以如果碰到這類接腳,在使用時必須先改為Digital Mode。
- 在PIC32MX中,透過ADxPCFG來設定。設1為Digital mode,0為Analog mode。





C32 對 ADC 的支援

- 常用的ADC函數

```
OpenADC10( );                 // 啟用ADC,設定ADC工作模式。
SetChanADC10( );              //手動切換ADCx的取樣通道。
ConvertADC10( );              //手動觸發轉換( SAMP -> 0 ) 。
BusyADC10( );                 // 測試ADC是否忙碌中?
ReadADC10( );                 // 自AD Buffer讀取AD轉換後的結果。
ConfigIntADC10( );            // 設定AD的中斷優先權,中斷啟用。
…
```

- 詳細說明請參閱

32-bit-Peripheral-Library-Guide.pdf,Sec.18 ADC10 Functions。

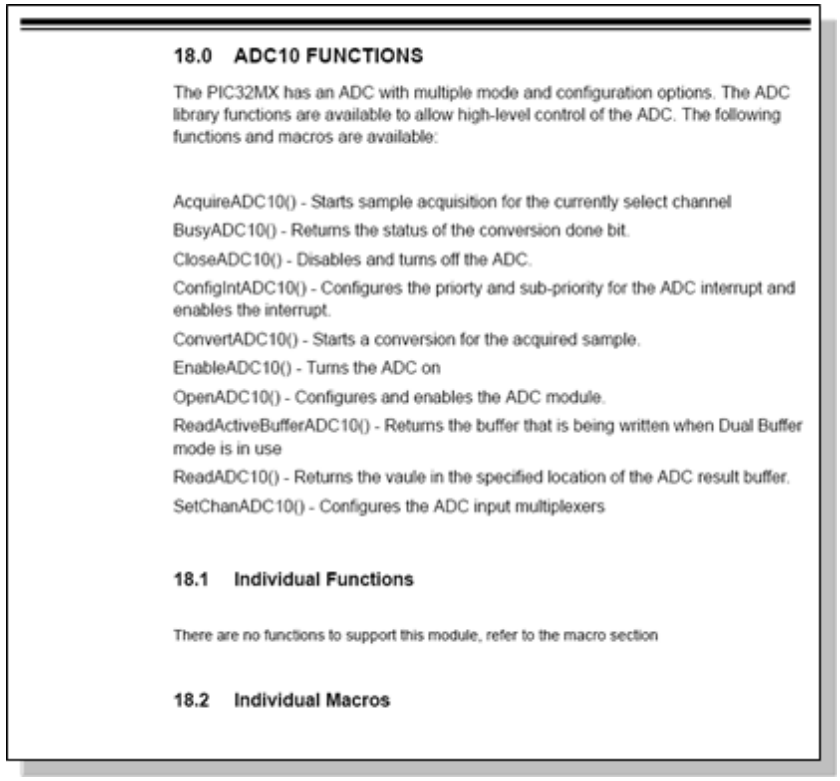
C32 ADC Code Example

- 簡單的ADC初始化範例

  OpenADC10(ADC_MODULE_ON & ADC_FORMAT_INTG & … ,);
  SetChanADC10(ADC_CH0_POS_SAMPLEA_AN5 & …);
  mPORTAConfig(…);
- 簡單的ADC讀值範例

  while(BusyADC10( ));
  ADValue = ReadADC10( );
- 簡單的ADC中斷開啟範例

  ConfigIntADC10
  (ADC_INT_ENABLE & ADC_INT_PRI_4);



18.0 ADC10 FUNCTIONS

The PIC32MX has an ADC with multiple mode and configuration options. The ADC library functions are available to allow high-level control of the ADC. The following functions and macros are available:

AcquireADC10() - Starts sample acquisition for the currently select channel
BusyADC10() - Returns the status of the conversion done bit
CloseADC10() - Disables and turns off the ADC
ConfigIntADC10() - Configures the priorty and sub-priority for the ADC interrupt and enables the interrupt.
ConvertADC10() - Starts a conversion for the acquired sample.
EnableADC10() - Turns the ADC on
OpenADC10() - Configures and enables the ADC module.
ReadActiveBufferADC10() - Returns the buffer that is being written when Dual Buffer mode is in use
ReadADC10() - Returns the vaule in the specified location of the ADC result buffer.
SetChanADC10() - Configures the ADC input multiplexers

18.1 Individual Functions

There are no functions to support this module, refer to the macro section

18.2 Individual Macros

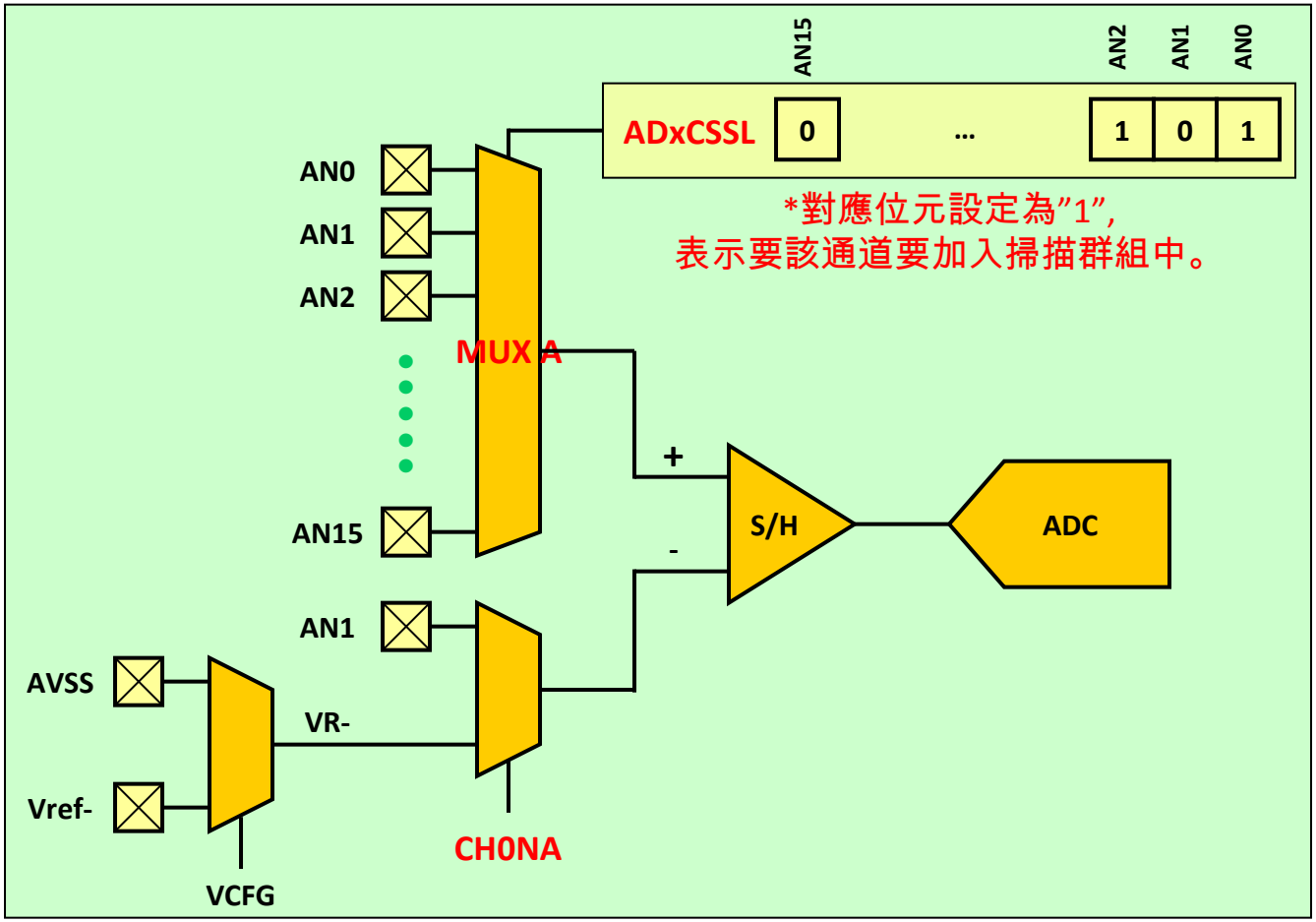
MICROCHIP
Regional Training Centers

Lab4 ADC Single (20 mins)

- 嘗試在Lab4的程式基礎上,加入ADC的相關功能。
- 設定為自動取樣與手動轉換模式,並手動指定類比通道為AN2。關閉通道掃描等功能。
- 利用Timer1每100mS,手動觸發ADC轉換。
- 正確設定完成後,可以LCD Module上看到VR1的值(0~1023)。
- 注意!APP1632的VR1是接到PICtail Plus中AN5/RB5的腳位。但APP1632-2的PIC32 PIM Module已經依照美國實驗板的跳接將CPU的AN2/RB2跳接至PICtail Plus介面的AN5/RB5,所以 VR1 的正確接線為接至 PIC32MX795 的 AN2/RB2。

ADC Auto Channel Scan

- 多工器A支援自動通道掃描,開啟通道掃描(CSCNA=1),就不需要手動設定通道。
- ADC會自動的切換通道,取得類比電壓,進行掃描。
- 搭配轉換次數(SMPI<3:0>)的設定可以規劃轉換資料存在Buffer中的方式。



MICROCHIP
Regional Training Centers

Lab5 ADC Scan (10 mins)

- 嘗試在Lab5的程式基礎上,修改ADC的工作模式。
- 與Lab4相同,設定為自動取樣與手動轉換模式, SMPI設定為2 Conversions/Interrupt。啟動ADC通道掃瞄功能,指定掃瞄AN2(VR1),AN4(VR2) 。
- 利用Timer1每50mS,手動觸發ADC轉換。
- 由於SMPI設定為2,第一次轉換的結果會存在ADCxBUF0,第二次會存在ADCxBUF1。兩次轉換後發出中斷需求。Buffer Index歸零, 從ADCxBUF0填起。
- 正確設定完成後,可以LCD Module上看到VR1,VR2的值(0~1023)。

MICROCHIP
Regional Training Centers

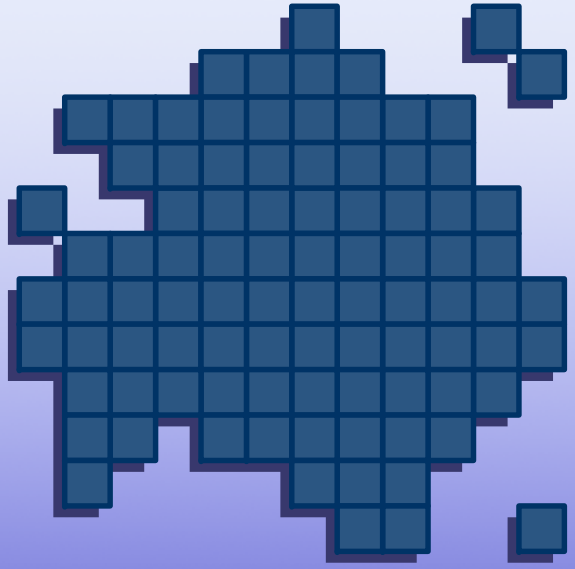
Ping-Pong Buffer Mode

- ADC的Buffer在操作時,有可能會發生ADC與CPU同時存取的狀況,造成資料不一致或被破壞的情形。
- 通常的情形是,讀取Buffer的速度太慢,CPU正準備讀取Buffer時,ADC也同時在將新的轉換結果存入Buffer中。
- ADC Buffer可以切割成兩塊各8的32-Bit的Buffer,作為Ping-Pong Buffer使用 (ADCxBUF0~7,ADCxBUF8~F),避免上述問題 。
- ADC在每次中斷時,會交替的存取兩塊區域,並透過BUFS來標示目前存取的區域。CPU必須先檢查BUFS,得知ADC的存取區塊,然後自另一塊空間,取得資料。
- BUFS=0時,就從ADCxBUF8~F取得資料。BUFS=1時,就改由ADCxBUF0~7取得資料。

MICROCHIP
Regional Training Centers

Lab6 Timer Trigger ADC (10 mins)

- 嘗試在Lab6的程式基礎上, 修改ADC的工作模式。
- 設定為自動取樣與TIMER3轉換模式,並手動指定類比通道為AN2。開啟Ping-Pong Buffer Mode,關閉通道掃描等功能。
- 設定Timer3每100mS, 觸發ADC轉換。
- 開啟ADC中斷並建立AD中斷服務常式。並於AD中斷服務常式中取得AD轉換數值。
- 由於開啟Ping-Pong Buffer Mode,在讀取Buffer時,為了避免跟AD同時存取一個區塊,必須透過BUFS Bits來確認ADC目前存取的區塊。
- 正確設定完成後,可以LCD Module上看到VR1的值(0~1023)。

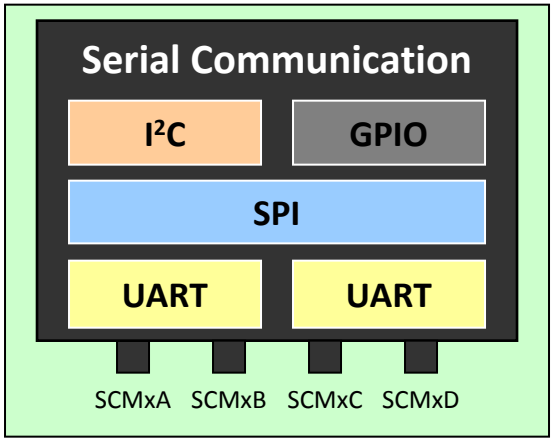


Serial Communication Modules

Only for PIC32MX5/6/7

Serial Communications Module

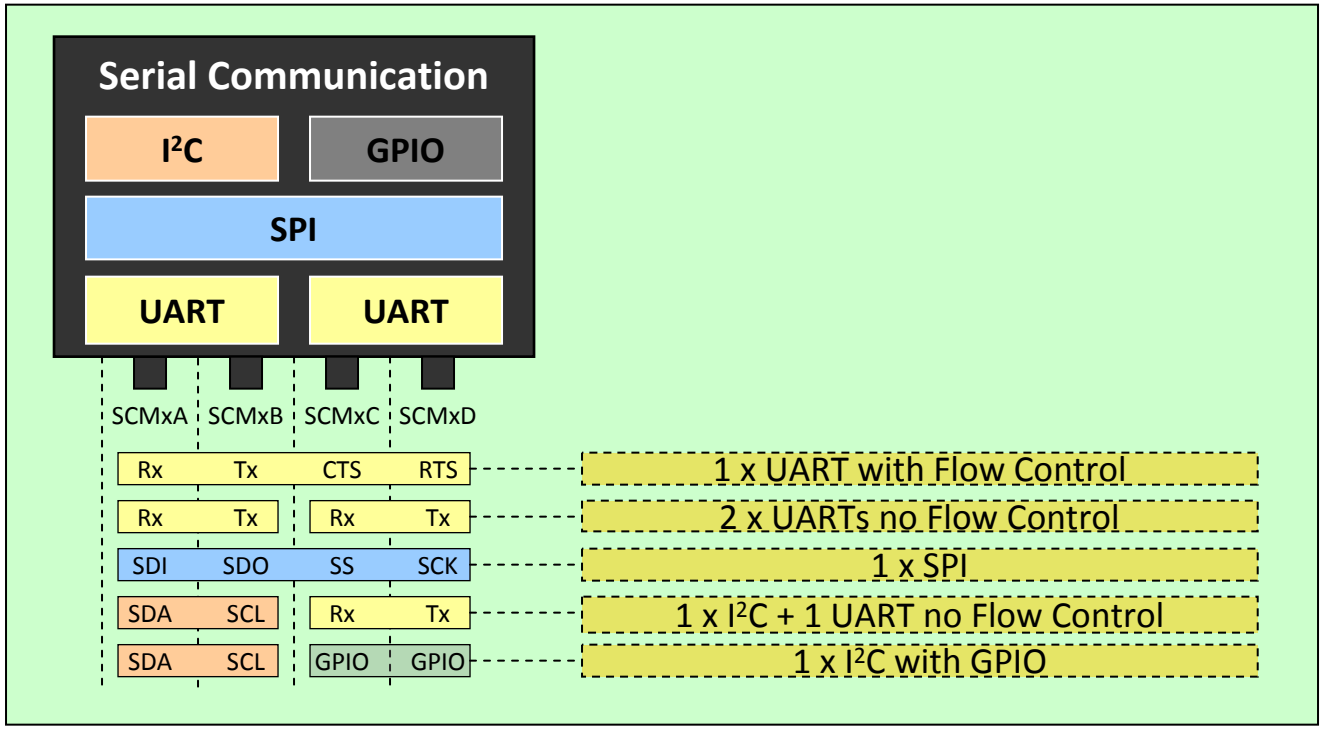
- PIC32MX5/6/7系列的串列通訊介面做了一個新的變革,採用Serial Communications Module取代了原本各自獨立的UART,SPI及I²C Module。
- Serial Communications Module,整合了UART,SPI及I²C Module於一體,透過軟體可以設定所要扮演的介面。每個通訊介面具有兩組UART,一組SPI及一組I²C。
- Serial Communications Module可以提高接腳運用的彈性,減少接腳功能定義上的衝突。



MICROCHIP
Regional Training Centers

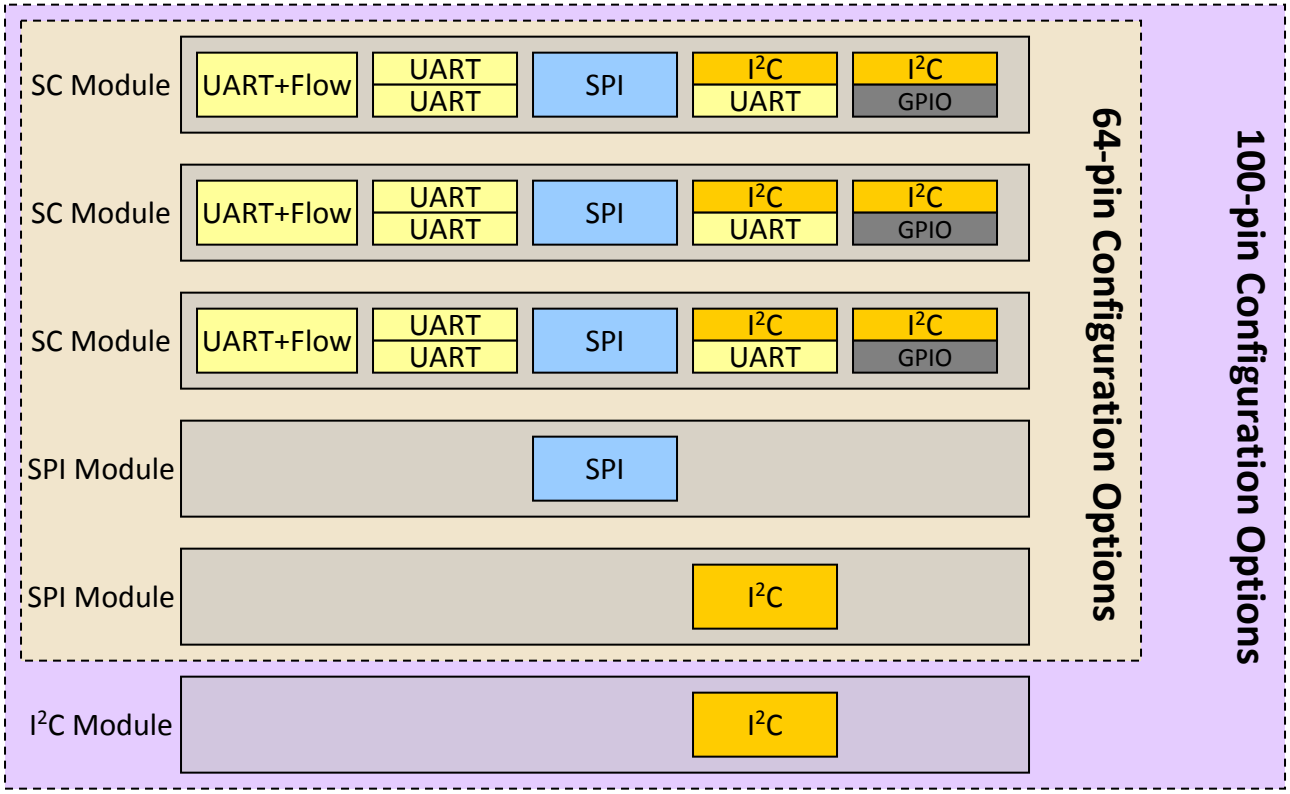
Module Options

- Serial Communications Module透過軟體可以設定成以下幾種的配置方式。



Module Options

- 所有的PIC32MX5/6/7系列,都有三組Serial Communications Module。64 Pins的元件中,額外有一組獨立的SPI及I²C,100 Pins的元件則有一組獨立的SPI及兩組I²C。



MICROCHIP
Regional Training Centers

C32 對 SC Module 的支援

- 雖然底層是透過Serial Communications Module來架構串列通訊的模組,不過在特殊功能暫存器上,還是承襲以往命名方式,使用如:UxMODE,SPIxCON,I2CxCON等暫存器來控制。
- MPLAB C32也將各個串列通訊模組的控制函數分開,所以使用上可以把Serial Communications Module當作透明般,單獨操作各個串列通訊模組。
- PIC32MX5/6/7系列的UART, I^2C及SPI控制上較為複雜。MPLAB C32中的相關函數也弄得很複雜,需要多花點時間才能理解使用。
- Lab 7, Lab 8,Lab 9提供完整的UART, I^2C, SPI Example Code。供學員參考,可利用課後時間好好的研究。

MICROCHIP
Regional Training Centers

Lab7,8,9
UART, I²C, SPI Example Code

- **Lab7 UART Rx Interrupt**
  UART收送的Demo Code。可使用超級終端機接收資料,在超級終端機上會顯示利用Timer1製作的電子鐘。在超級終端機輸入的資料會顯示在LCD Module上。
- **Lab 8 I2C**
  I²C存取的Demo Code 。存取APP1632 EVM Board上的I²C EEPROM。自動依序的存取EEPROM的各的位址,並將存入與讀出的值顯示於LCD Module上。
- **Lab 9 SPI**
  SPI存取的Demo Code 。存取APP1632 EVM Board上的SPI EEPROM。自動依序的存取EEPROM的各的位址,並將存入與讀出的值顯示於LCD Module上。

MICROCHIP
Regional Training Centers

C32 UART Code Example

- UART初始化範例

```
UARTConfigure( UART2 , UART_ENABLE_PINS_TX_RX_ONLY );
UARTSetFifoMode( UART2 , UART_INTERRUPT_ON_TX_NOT_FULL |
UART_INTERRUPT_ON_RX_NOT_EMPTY );
UARTSetLineControl( UART2 , UART_DATA_SIZE_8_BITS |
UART_PARITY_NONE | UART_STOP_BITS_1 );
UARTSetDataRate( UART2 , PBFrequency , 9600 );
UARTEnable( UART2 , UART_ENABLE | UART_PERIPHERAL | UART_TX |
UART_RX );
```

- UART中斷開啟範例

```
INTSetVectorPriority( INT_UART_2_VECTOR , INT_PRIORITY_LEVEL_5 );
INTSetVectorSubPriority( INT_UART_2_VECTOR ,
INT_SUB_PRIORITY_LEVEL_0 );
INTEnable( INT_U2RX , INT_ENABLED );
```

MICROCHIP
Regional Training Centers

C32 SPI Code Example

- SPI初始化範例

```
// SPI Chip Select
mPORTDSetPinsDigitalOut( BIT_12 );
mPORTDSetBits( BIT_12 );
SpiChnConfigure( SPI_CHANNEL2 , SPI_OPEN_MSTEN | SPI_OPEN_CKE_REV |
SPI_OPEN_MODE8 | SPI_OPEN_SIDL );
SpiChnSetBitRate( SPI_CHANNEL2 , PBFrequency , 100000 );
SpiChnEnable( SPI_CHANNEL2 , 1 );
```

- SPI Write/Read範例

```
while( SpiChnTxBuffFull( SPI_CHANNEL2 ) );
SpiChnWriteC( SPI_CHANNEL2 , data );
while( !SpiChnDataRdy( SPI_CHANNEL2 ) );
SpiChnReadC( SPI_CHANNEL2 );
```

MICROCHIP
Regional Training Centers

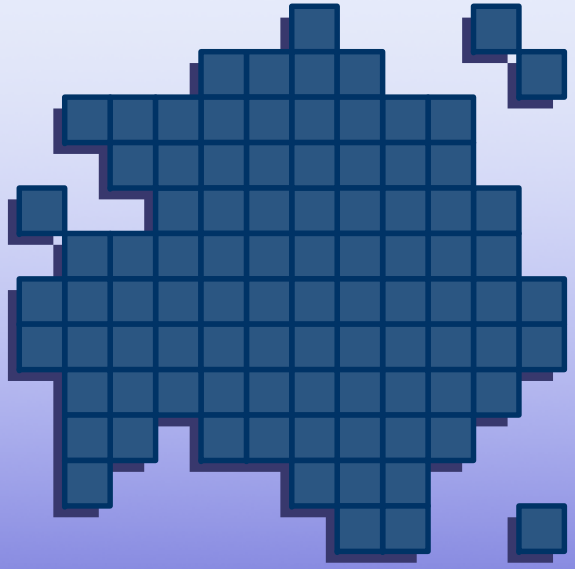
C32 I²C Code Example

- I²C初始化範例

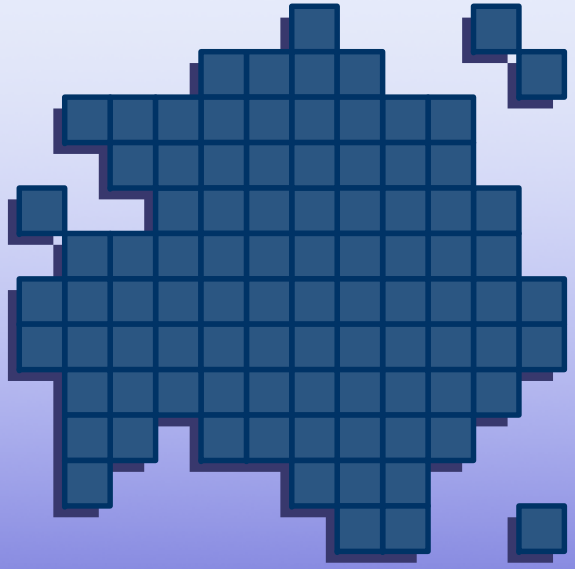
```
I2CConfigure( I2C1 , 0 );
I2CSetFrequency( I2C1 , PBFrequency , 100000 );
I2CSetSlaveAddress( I2C1, 0x00 , 0x00 , I2C_USE_7BIT_ADDRESS );
I2CEnable( I2C1, 1 );
```

- I²C Write/Read範例

```
while ( !( I2CGetStatus( I2C1 ) & I2C_START ) );
while( !I2CTransmitterIsReady( I2C1 ) );
I2CSendByte( I2C1 , SalveAddr & 0xfffe );
while( !I2CTransmissionHasCompleted( I2C1 ) );
while( !I2CByteWasAcknowledged( I2C1 ) );
```

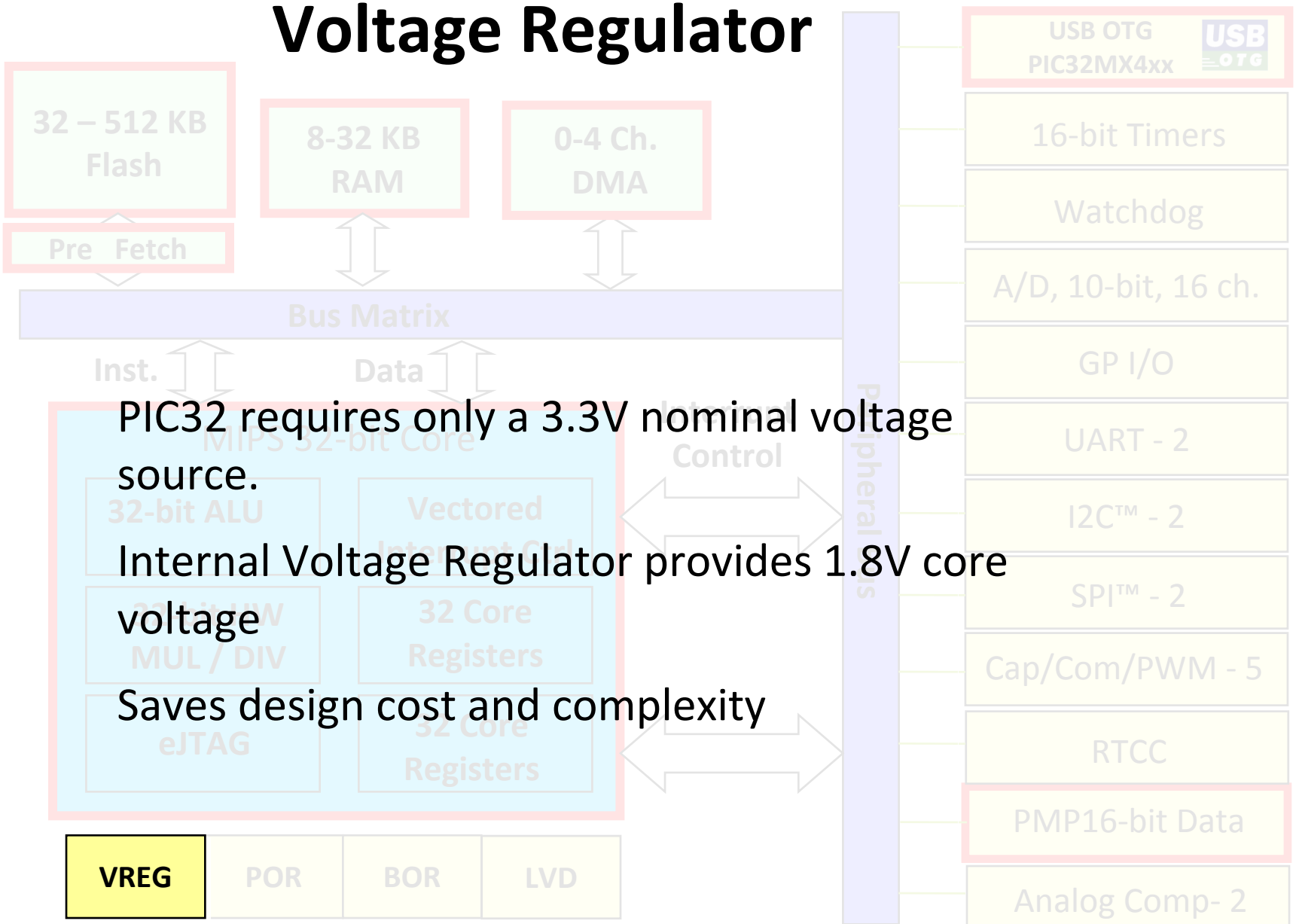


The End, Thank you !!



Reference Data

Voltage Regulator



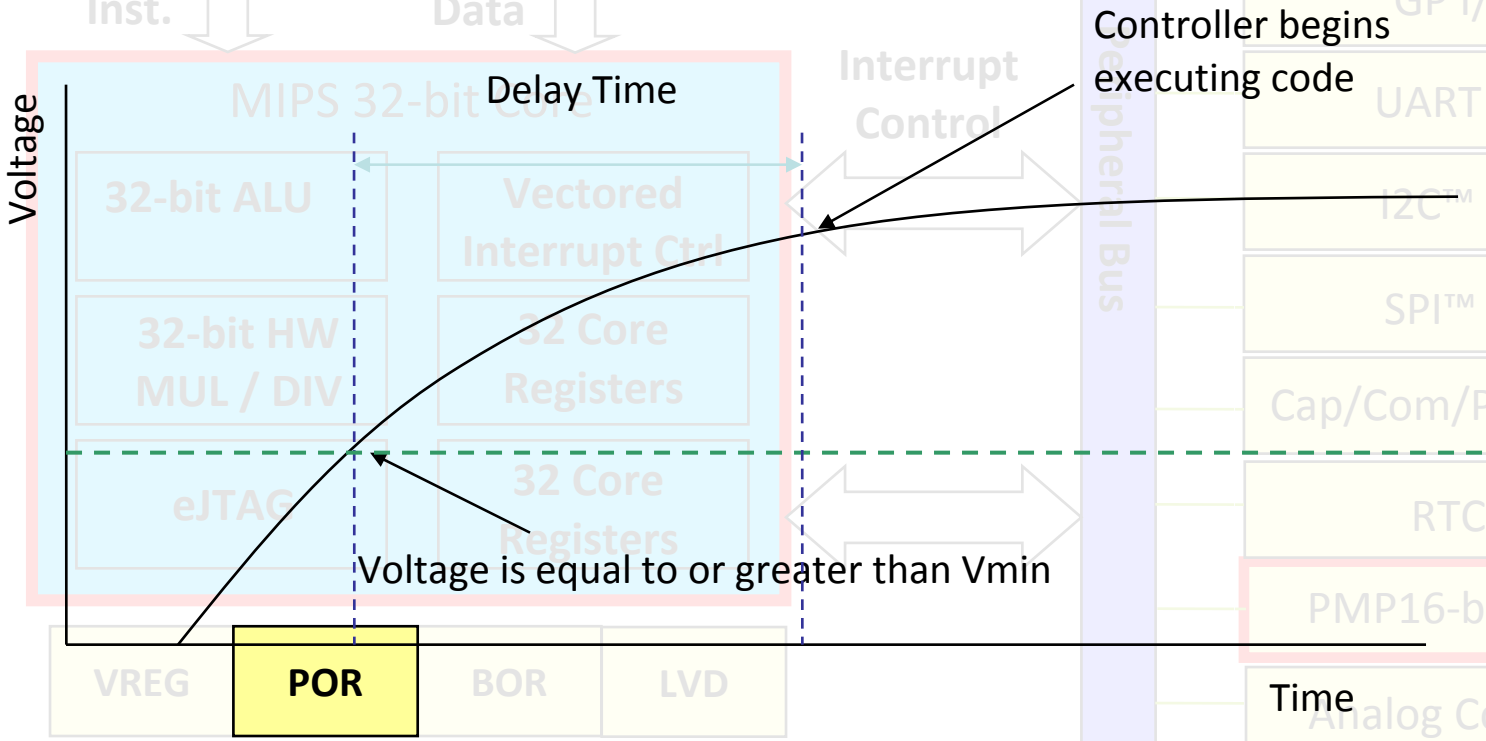
PIC32 requires only a 3.3V nominal voltage source.

Internal Voltage Regulator provides 1.8V core voltage

Saves design cost and complexity

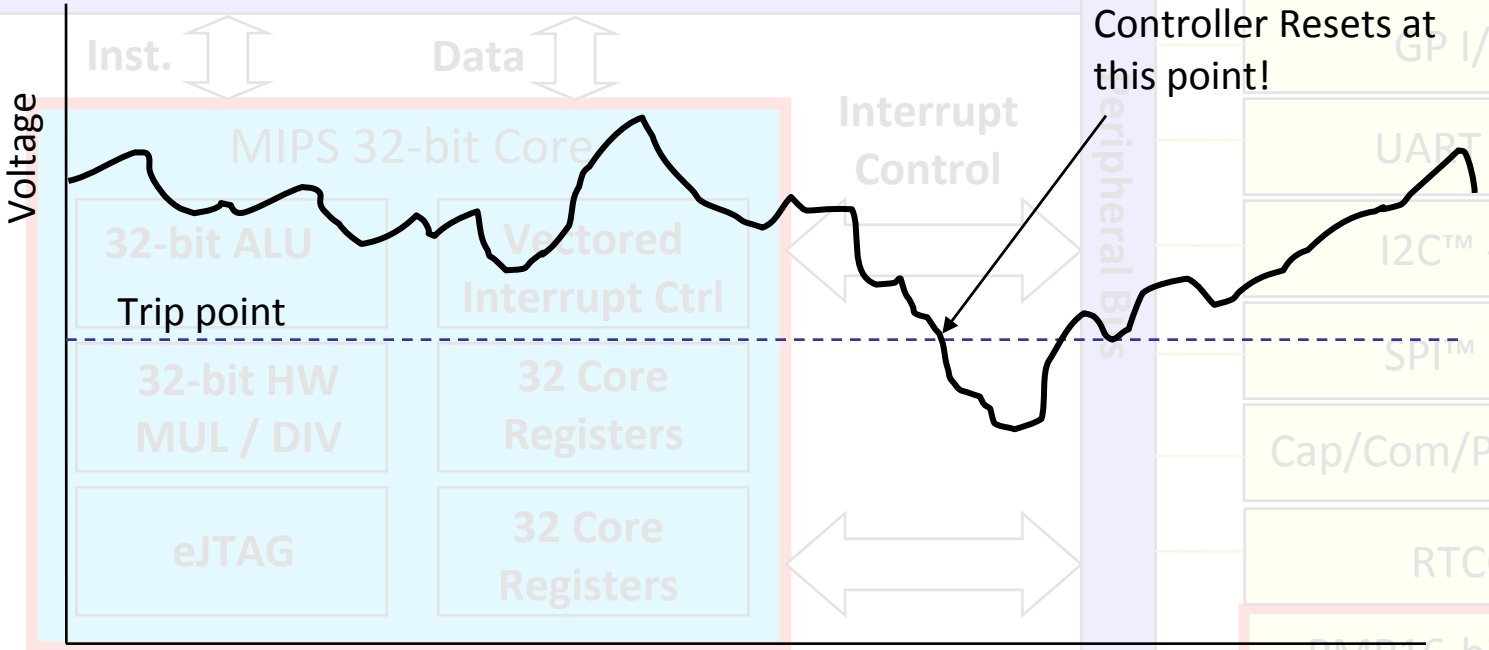
Power On Reset

- When Supply voltage is first applied, this peripheral holds the device in Reset for a short time, allowing the supply to become stable before allowing the processor to start executing code.



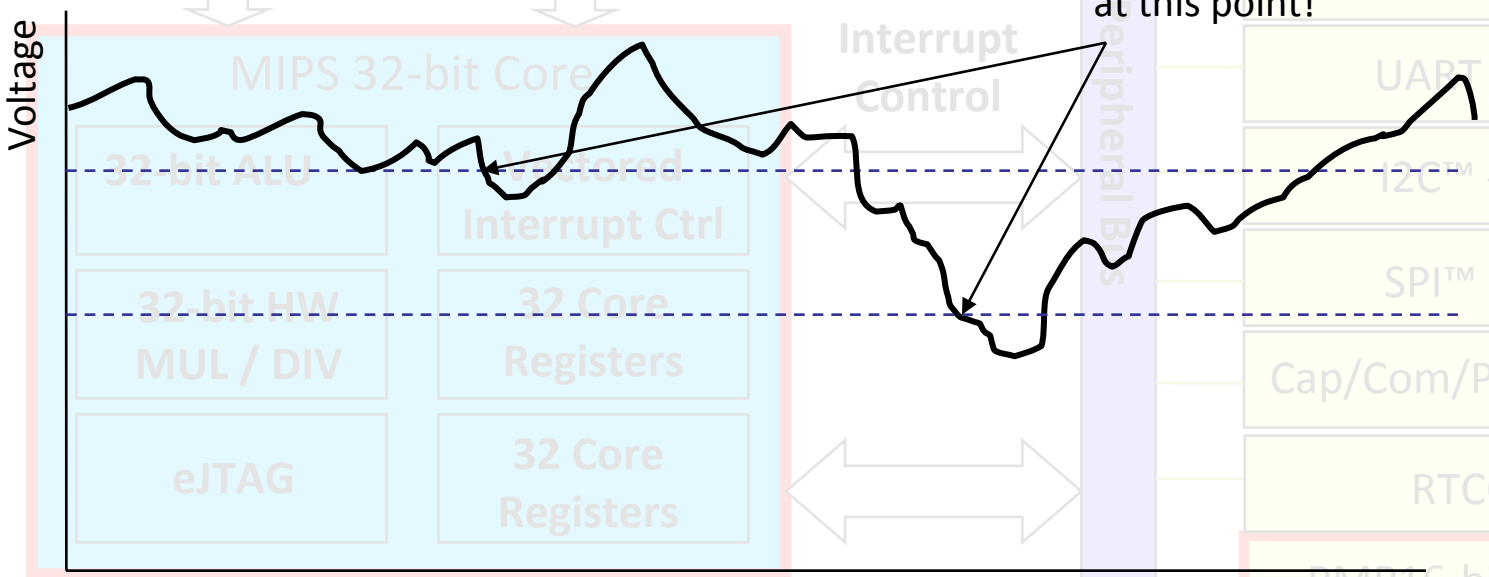
Brown Out Reset

If enabled, this peripheral will reset the microcontroller if the Source Voltage drops below a certain level, or Trip point. We want this to happen, because the program could behave erratically if voltage is bad.

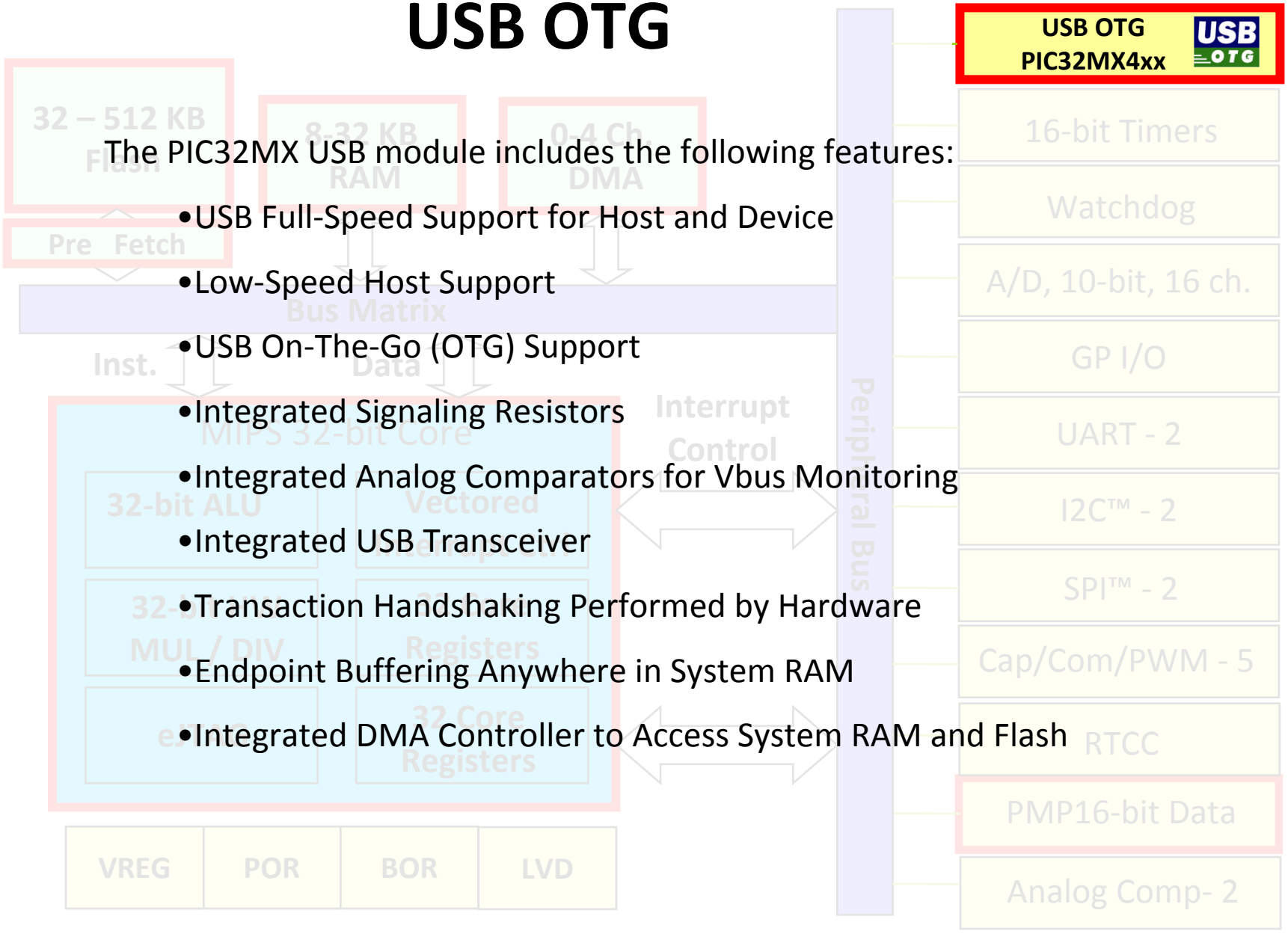


Low Voltage Detect

- Low Voltage Detect and High/Low Voltage Detect
- Monitor Supply Voltage and trigger an Interrupt if voltage crosses the detect value



USB OTG



The PIC32MX USB module includes the following features:

- USB Full-Speed Support for Host and Device
- Low-Speed Host Support
- USB On-The-Go (OTG) Support
- Integrated Signaling Resistors
- Integrated Analog Comparators for Vbus Monitoring
- Integrated USB Transceiver
- Transaction Handshaking Performed by Hardware
- Endpoint Buffering Anywhere in System RAM
- Integrated DMA Controller to Access System RAM and Flash

16 bit Timers

All PIC32 Timers are 16 bit. There are two types, A and B.

All timers have the following features:

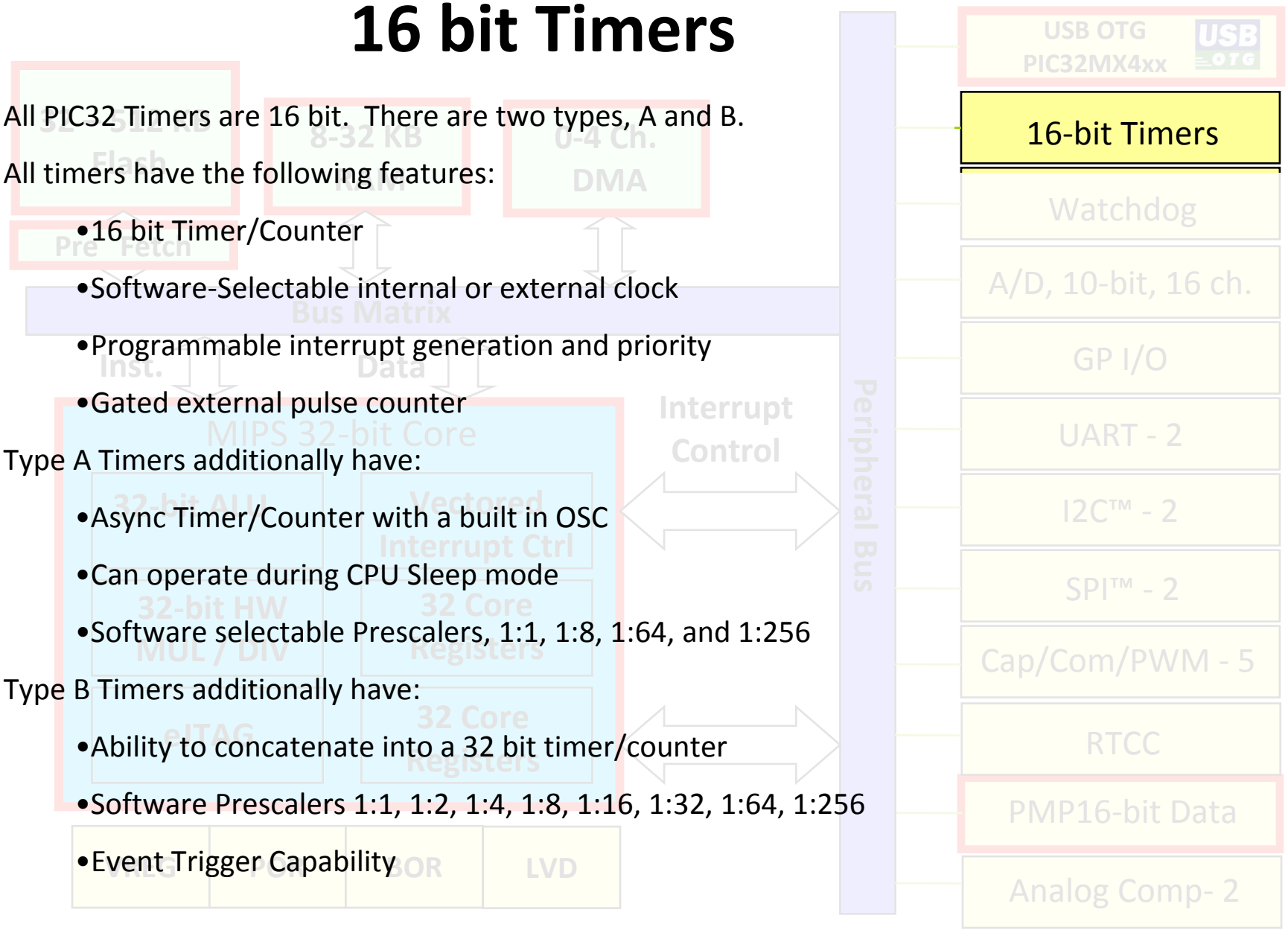
- 16 bit Timer/Counter
- Software-Selectable internal or external clock
- Programmable interrupt generation and priority
- Gated external pulse counter

Type A Timers additionally have:

- Async Timer/Counter with a built in OSC
- Can operate during CPU Sleep mode
- Software selectable Prescalers, 1:1, 1:8, 1:64, and 1:256

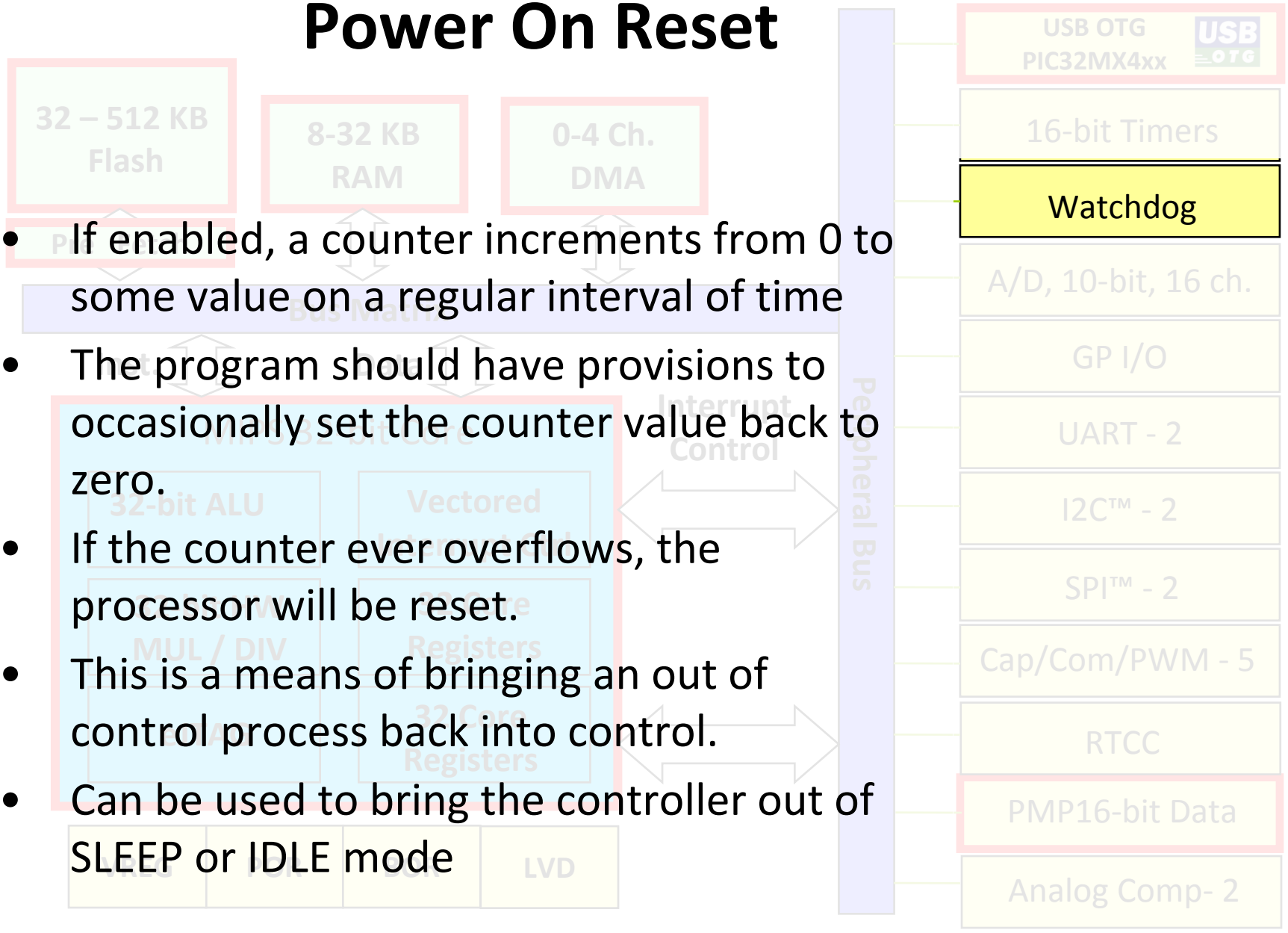
Type B Timers additionally have:

- Ability to concatenate into a 32 bit timer/counter
- Software Prescalers 1:1, 1:2, 1:4, 1:8, 1:16, 1:32, 1:64, 1:256
- Event Trigger Capability



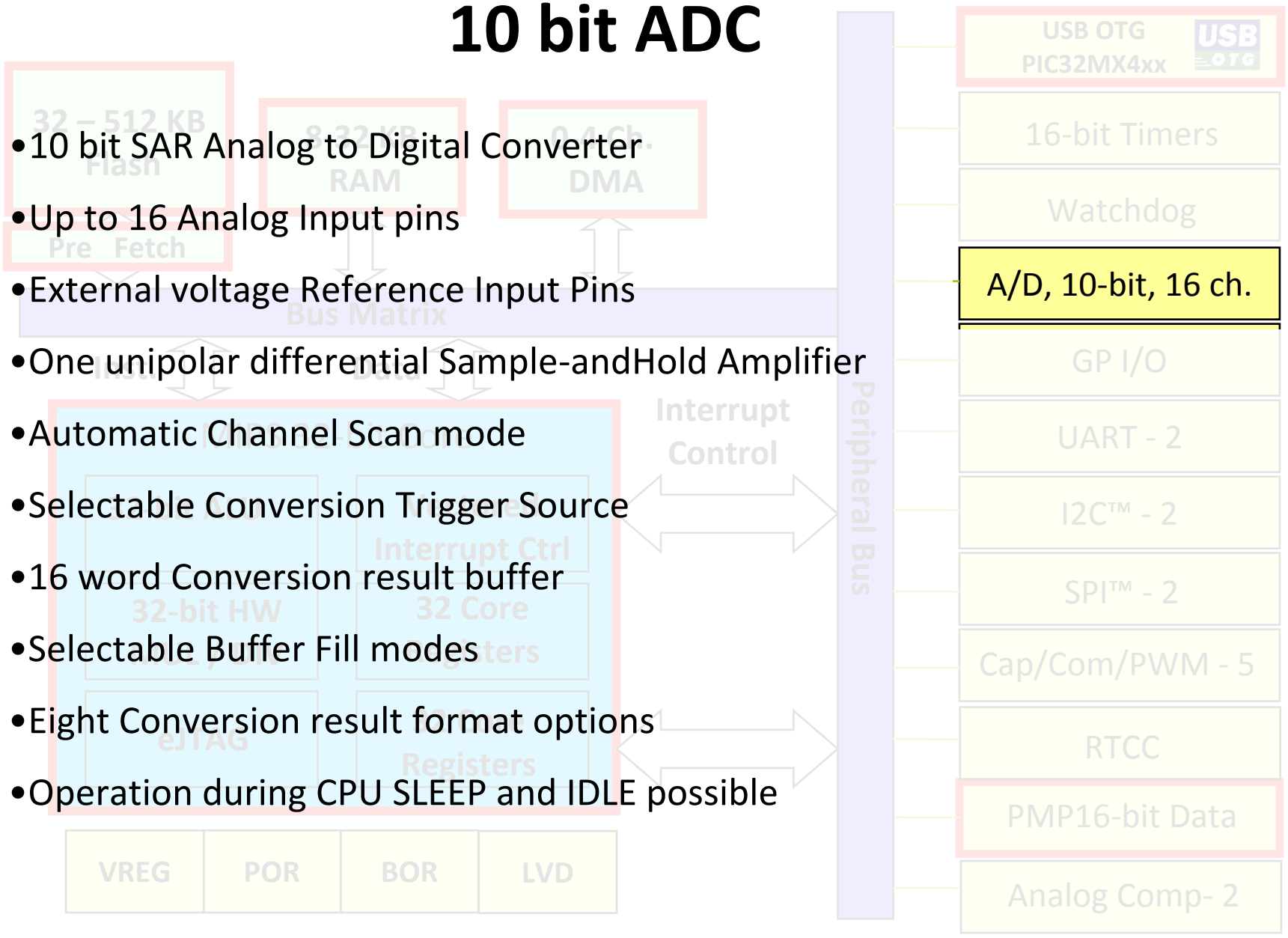
Power On Reset

- If enabled, a counter increments from 0 to some value on a regular interval of time
- The program should have provisions to occasionally set the counter value back to zero.
- If the counter ever overflows, the processor will be reset.
- This is a means of bringing an out of control process back into control.
- Can be used to bring the controller out of SLEEP or IDLE mode



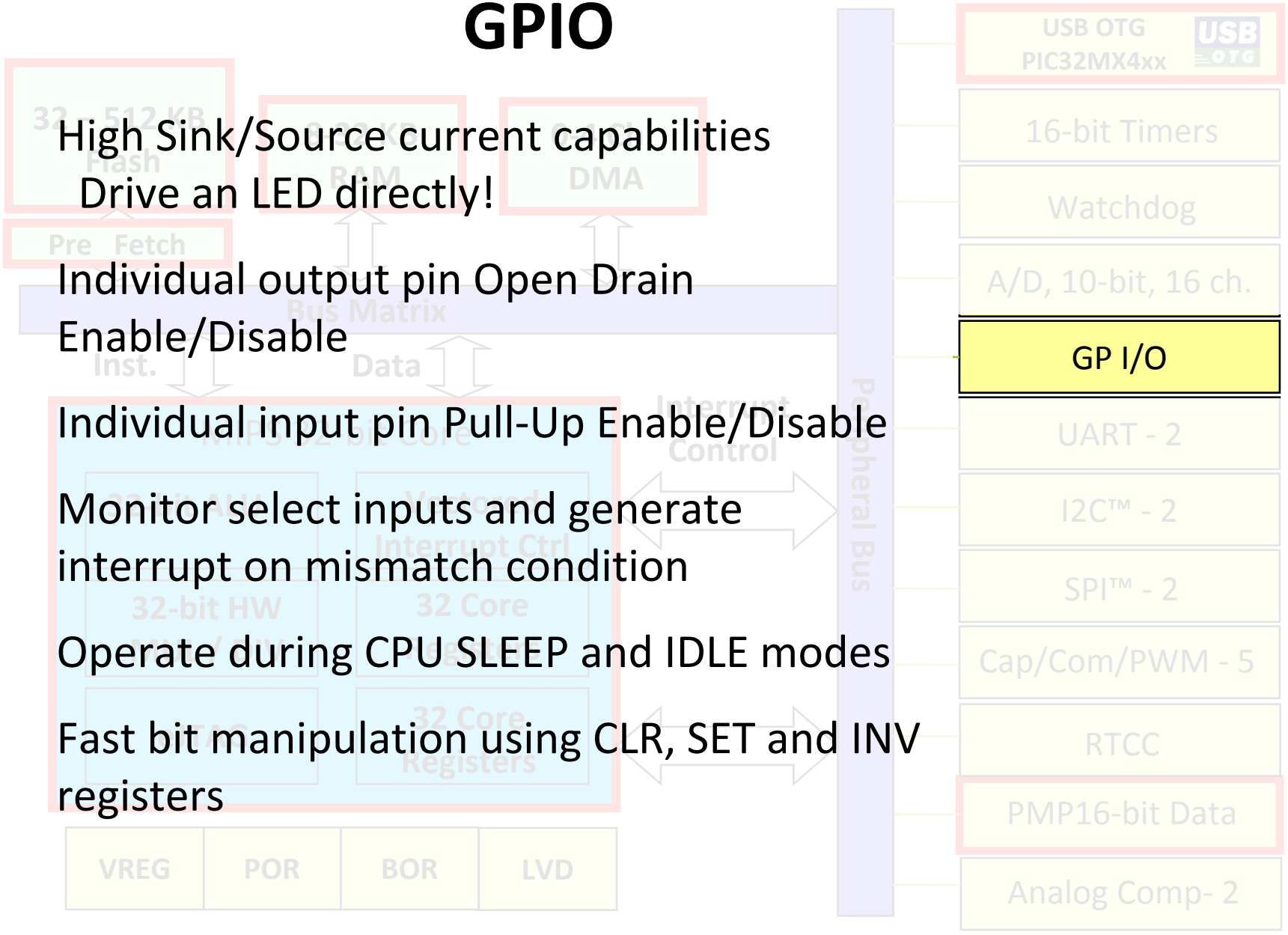
10 bit ADC

- 10 bit SAR Analog to Digital Converter
- Up to 16 Analog Input pins
- External voltage Reference Input Pins
- One unipolar differential Sample-andHold Amplifier
- Automatic Channel Scan mode
- Selectable Conversion Trigger Source
- 16 word Conversion result buffer
- Selectable Buffer Fill modes
- Eight Conversion result format options
- Operation during CPU SLEEP and IDLE possible



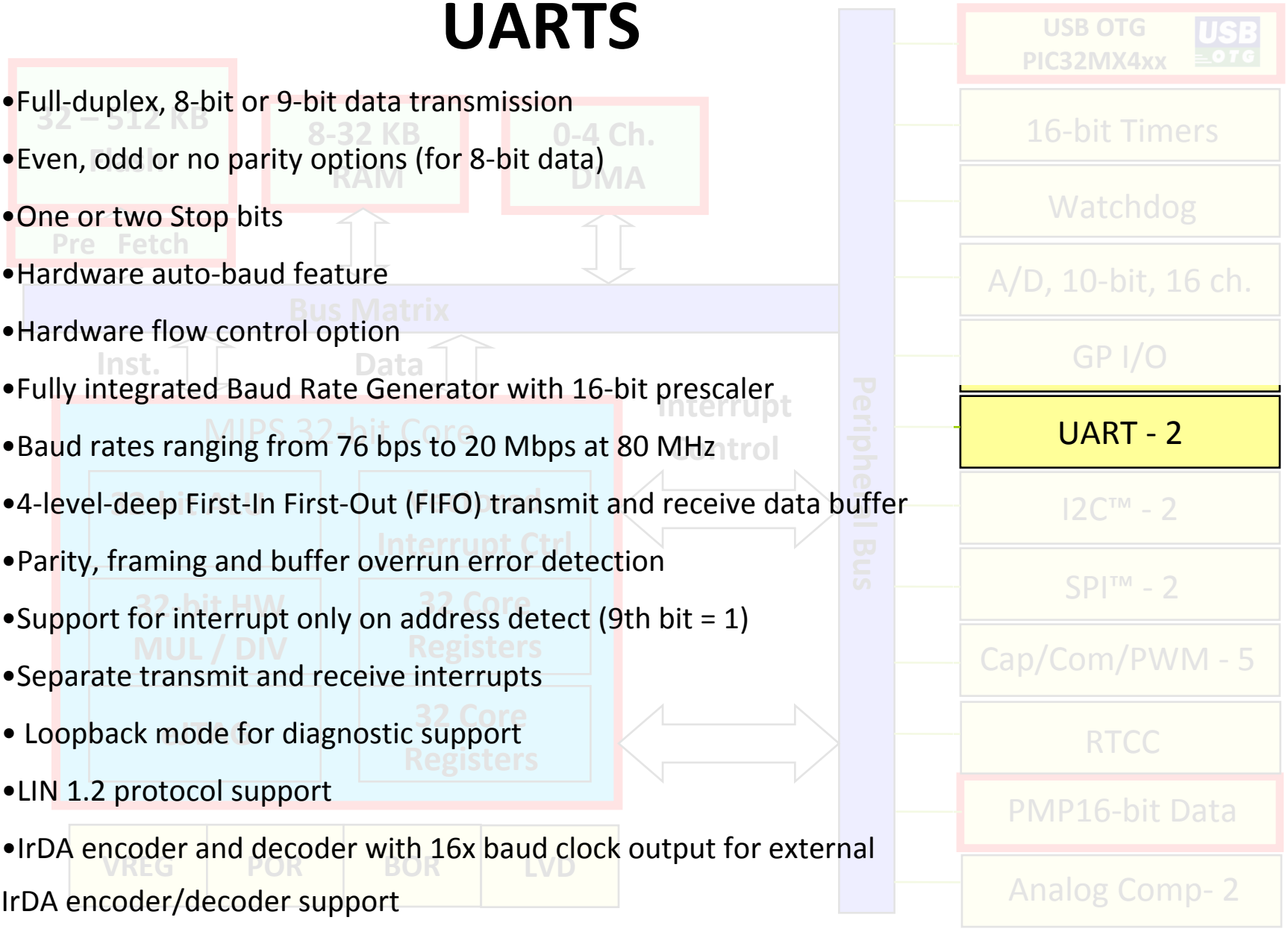
GPIO

- High Sink/Source current capabilities
  - Drive an LED directly!
- Individual output pin Open Drain Enable/Disable
- Individual input pin Pull-Up Enable/Disable
- Monitor select inputs and generate interrupt on mismatch condition
- Operate during CPU SLEEP and IDLE modes
- Fast bit manipulation using CLR, SET and INV registers



UARTS

- Full-duplex, 8-bit or 9-bit data transmission
- Even, odd or no parity options (for 8-bit data)
- One or two Stop bits
- Hardware auto-baud feature
- Hardware flow control option
- Fully integrated Baud Rate Generator with 16-bit prescaler
- Baud rates ranging from 76 bps to 20 Mbps at 80 MHz
- 4-level-deep First-In First-Out (FIFO) transmit and receive data buffer
- Parity, framing and buffer overrun error detection
- Support for interrupt only on address detect (9th bit = 1)
- Separate transmit and receive interrupts
- Loopback mode for diagnostic support
- LIN 1.2 protocol support
- IrDA encoder and decoder with 16x baud clock output for external IrDA encoder/decoder support

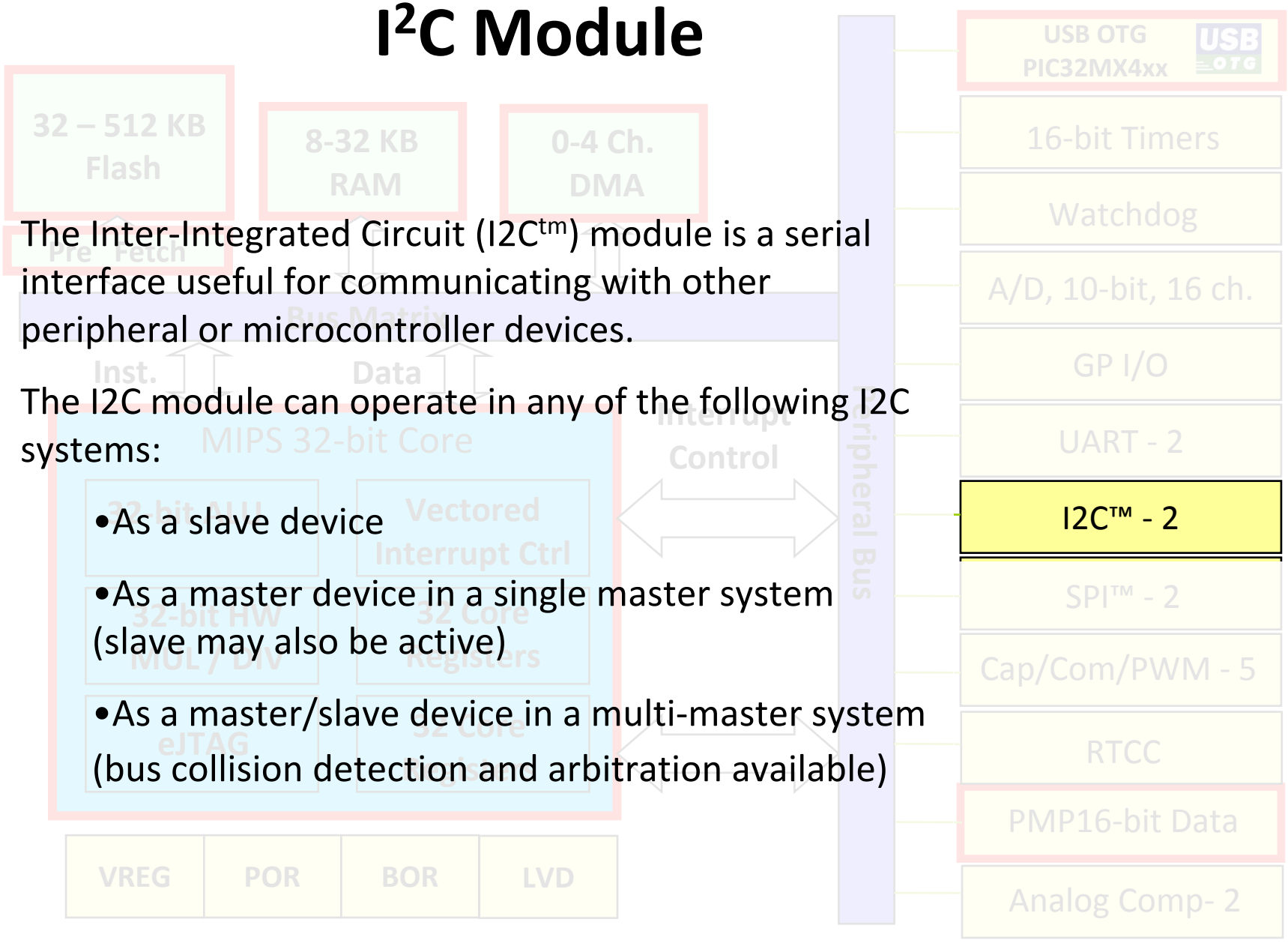


I²C Module

The Inter-Integrated Circuit (I2C™) module is a serial interface useful for communicating with other peripheral or microcontroller devices.

The I2C module can operate in any of the following I2C systems:

- As a slave device
- As a master device in a single master system (slave may also be active)
- As a master/slave device in a multi-master system (bus collision detection and arbitration available)

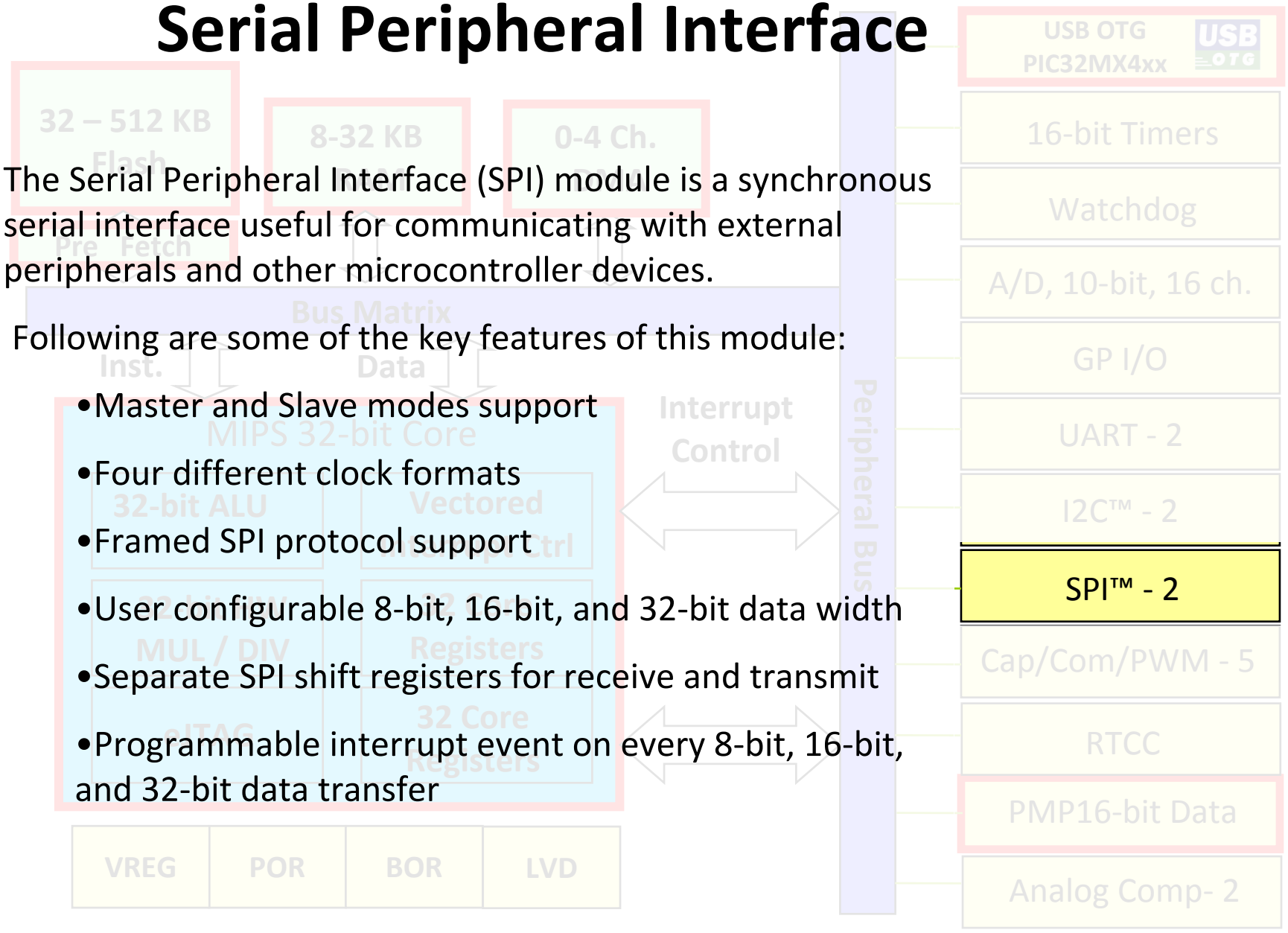


Serial Peripheral Interface

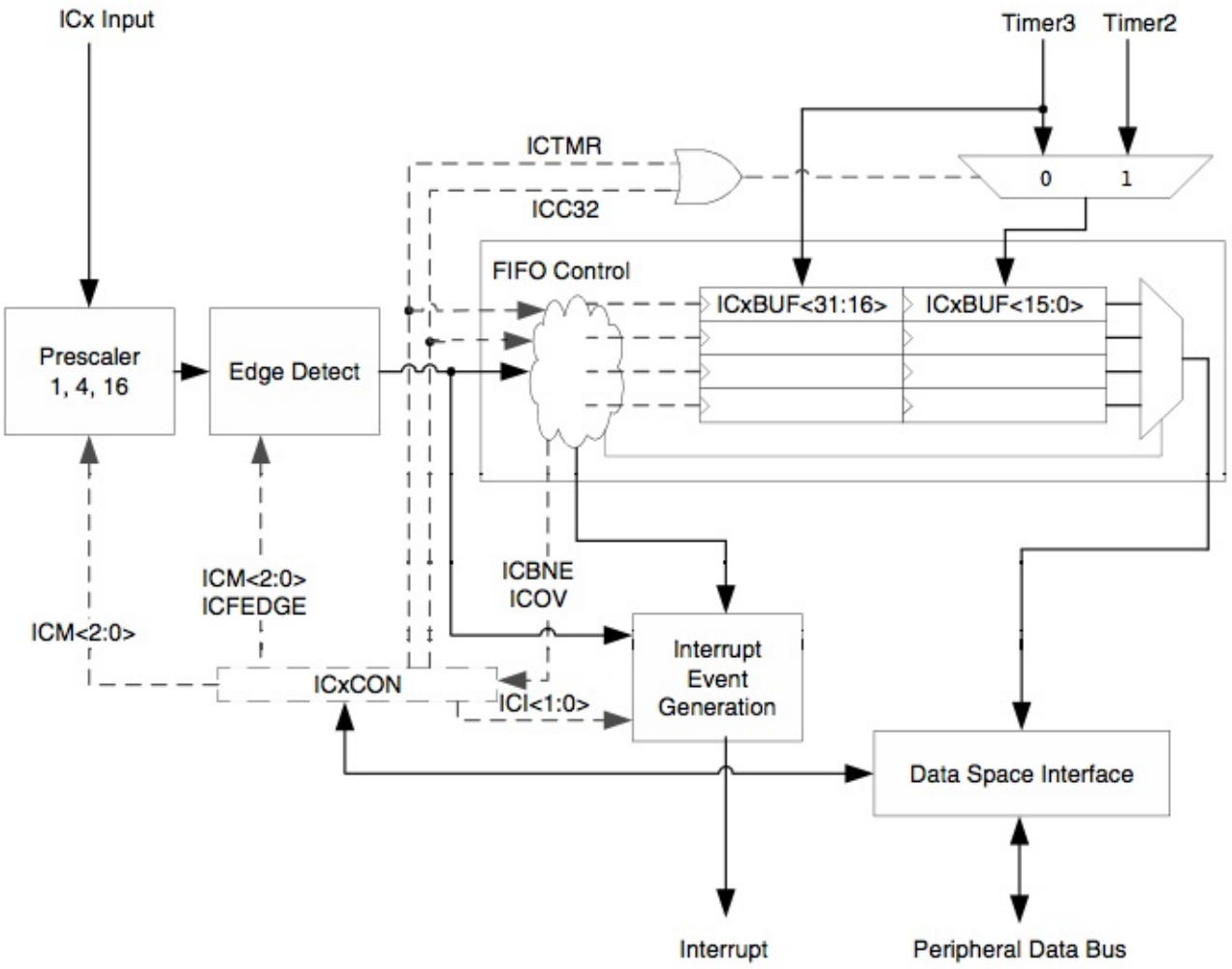
The Serial Peripheral Interface (SPI) module is a synchronous serial interface useful for communicating with external peripherals and other microcontroller devices.

Following are some of the key features of this module:

- Master and Slave modes support
- Four different clock formats
- Framed SPI protocol support
- User configurable 8-bit, 16-bit, and 32-bit data width
- Separate SPI shift registers for receive and transmit
- Programmable interrupt event on every 8-bit, 16-bit, and 32-bit data transfer



Input Capture



Can capture every rising or falling event, and can count every 4th or 16th event

USB OTG PIC32MX4xx

16-bit Timers

Watchdog

A/D, 10-bit, 16 ch.

GP I/O

CAN - 2

I2C™ - 2

SPI™ - 2

Cap/Com/PWM - 5

RTCC

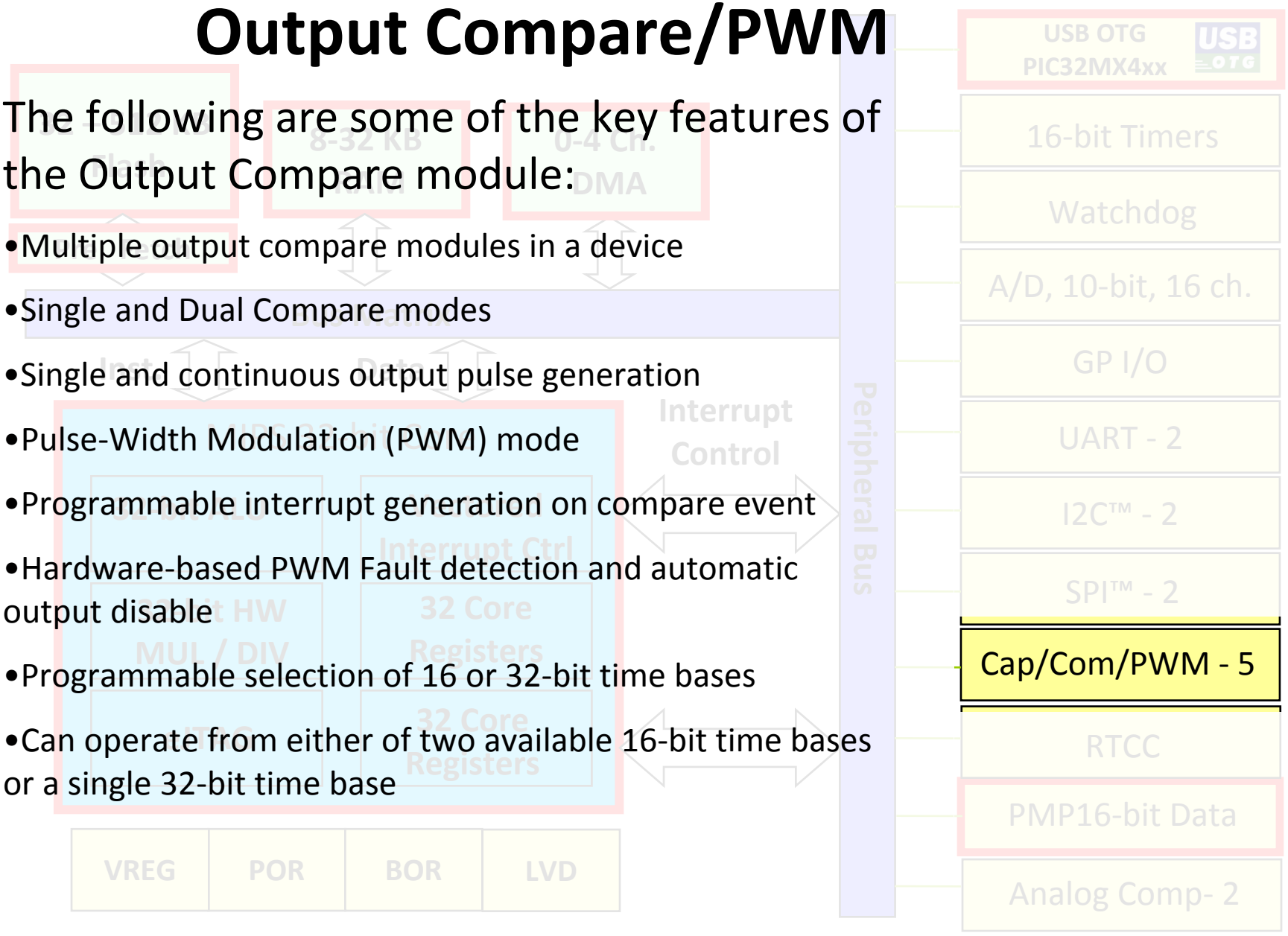
PMP16-bit Data

Analog Comp- 2

Output Compare/PWM

The following are some of the key features of the Output Compare module:

- Multiple output compare modules in a device
- Single and Dual Compare modes
- Single and continuous output pulse generation
- Pulse-Width Modulation (PWM) mode
- Programmable interrupt generation on compare event
- Hardware-based PWM Fault detection and automatic output disable
- Programmable selection of 16 or 32-bit time bases
- Can operate from either of two available 16-bit time bases or a single 32-bit time base



Real Time Clock/Calander

Intended for applications where time must be maintained with minimum intervention from the CPU.

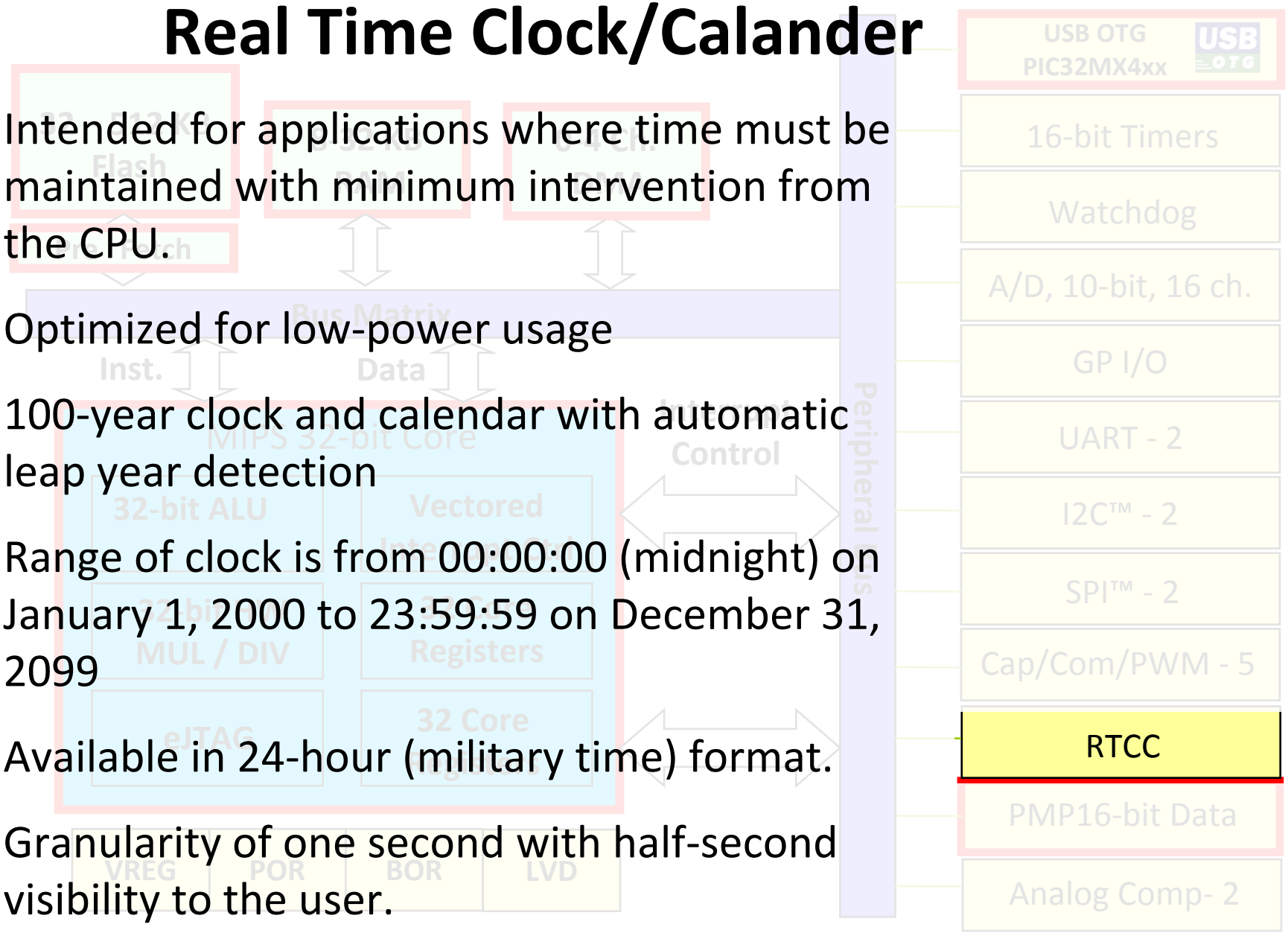
Optimized for low-power usage

100-year clock and calendar with automatic leap year detection

Range of clock is from 00:00:00 (midnight) on January 1, 2000 to 23:59:59 on December 31, 2099

Available in 24-hour (military time) format.

Granularity of one second with half-second visibility to the user.



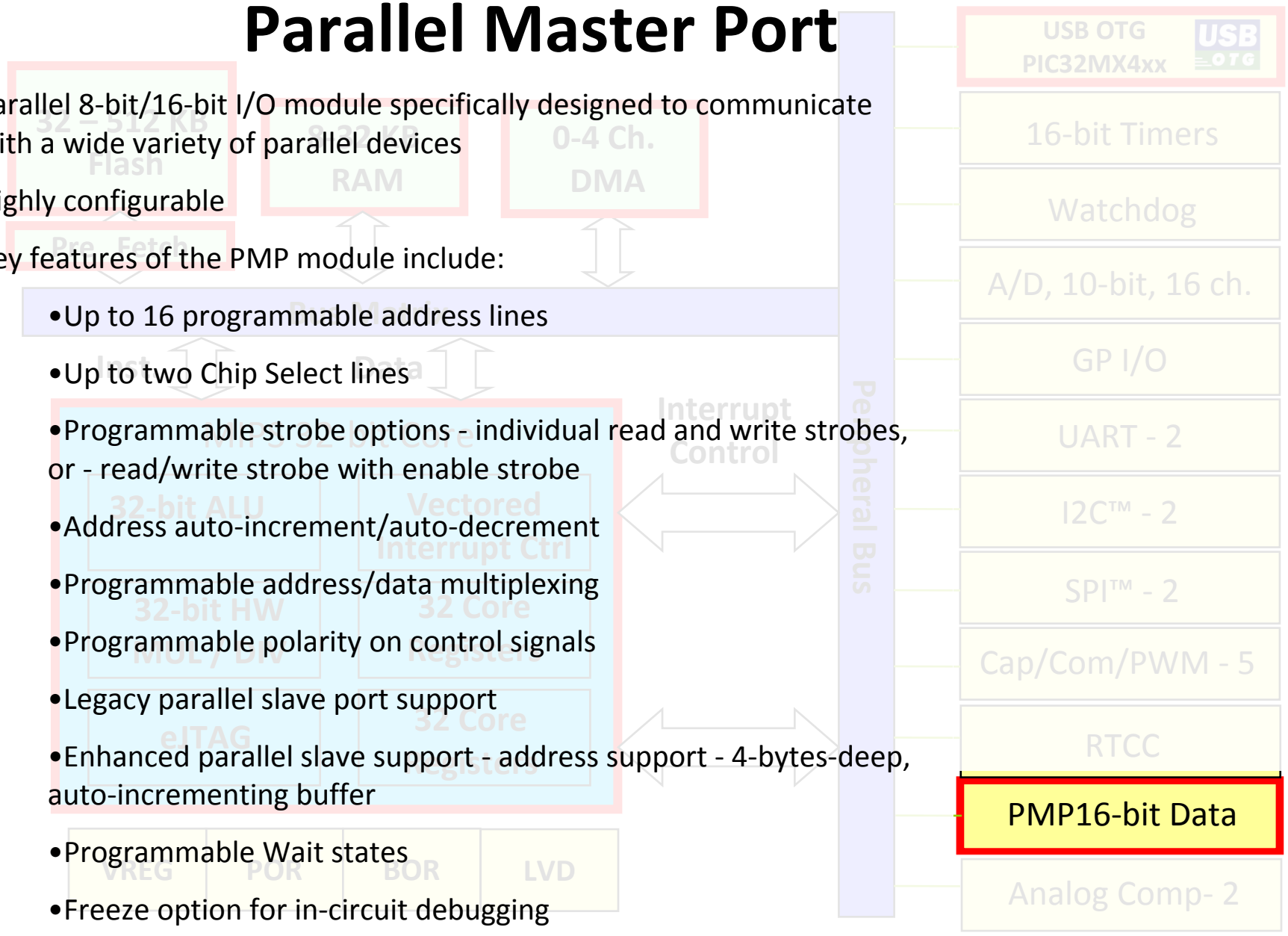
Parallel Master Port

Parallel 8-bit/16-bit I/O module specifically designed to communicate with a wide variety of parallel devices

Highly configurable

Key features of the PMP module include:

- Up to 16 programmable address lines
- Up to two Chip Select lines
- Programmable strobe options - individual read and write strobes, or - read/write strobe with enable strobe
- Address auto-increment/auto-decrement
- Programmable address/data multiplexing
- Programmable polarity on control signals
- Legacy parallel slave port support
- Enhanced parallel slave support - address support - 4-bytes-deep, auto-incrementing buffer
- Programmable Wait states
- Freeze option for in-circuit debugging



Analog Comparator

Can be configured in a variety of ways.

Selectable inputs available include:

- Analog inputs multiplexed with I/O pins
- On-Chip Internal Absolute Voltage Reference (IVREF)
- Comparator Voltage Reference (CVREF)

Outputs can be inverted

Selectable interrupt generation