MICROCHIP
MASTER'S

618 ICD

Audio Spectrum Analyzer
PIC18FXXX Hands On Workshop

MPLAB® IDE V6.0

MPLAB ICD 2

MPLAB C18

PIC18FXXX Hands On Workshop Agenda

- PIC18FXXXX architecture, peripherals and special features
- PICmicro® product overview including future products
- PIC18FXXXX development tool overview
- Audio Spectrum Analyzer Demo Board design
- Lab 1 - Install MPLAB 6.0, MPLAB ICD 2, MPLAB C18, Demo Board, Create Project, Compile and Run, Display Message
- Lab 2 - Develop a traffic light
- Lab 3 - A/D Sampling ISR, Fill A/D sample buffer
- Lab 4 - Apply DFT to A/D sample buffer, scale and display DFT results.
- Lab 5 - Extra credit- Add Automatic Gain Control

PIC18FXXX Workshop Appendix A-D

- The following Appendix topics are available for your reference, but will not be presented today:
  - Appendix A: Optimizing C source code for compiler efficiency
  - Appendix B: PIC18FXXXX Instruction Set, PIC16/17 migration
  - Appendix C: PIC18FXXXX Flash Programming Tips
  - Appendix D: PIC18FXXXX Peripheral Calculation Spreadsheet

Microchip Technology Inc.

Company Overview

MICROCHIP MASTER'S

Corporate Overview

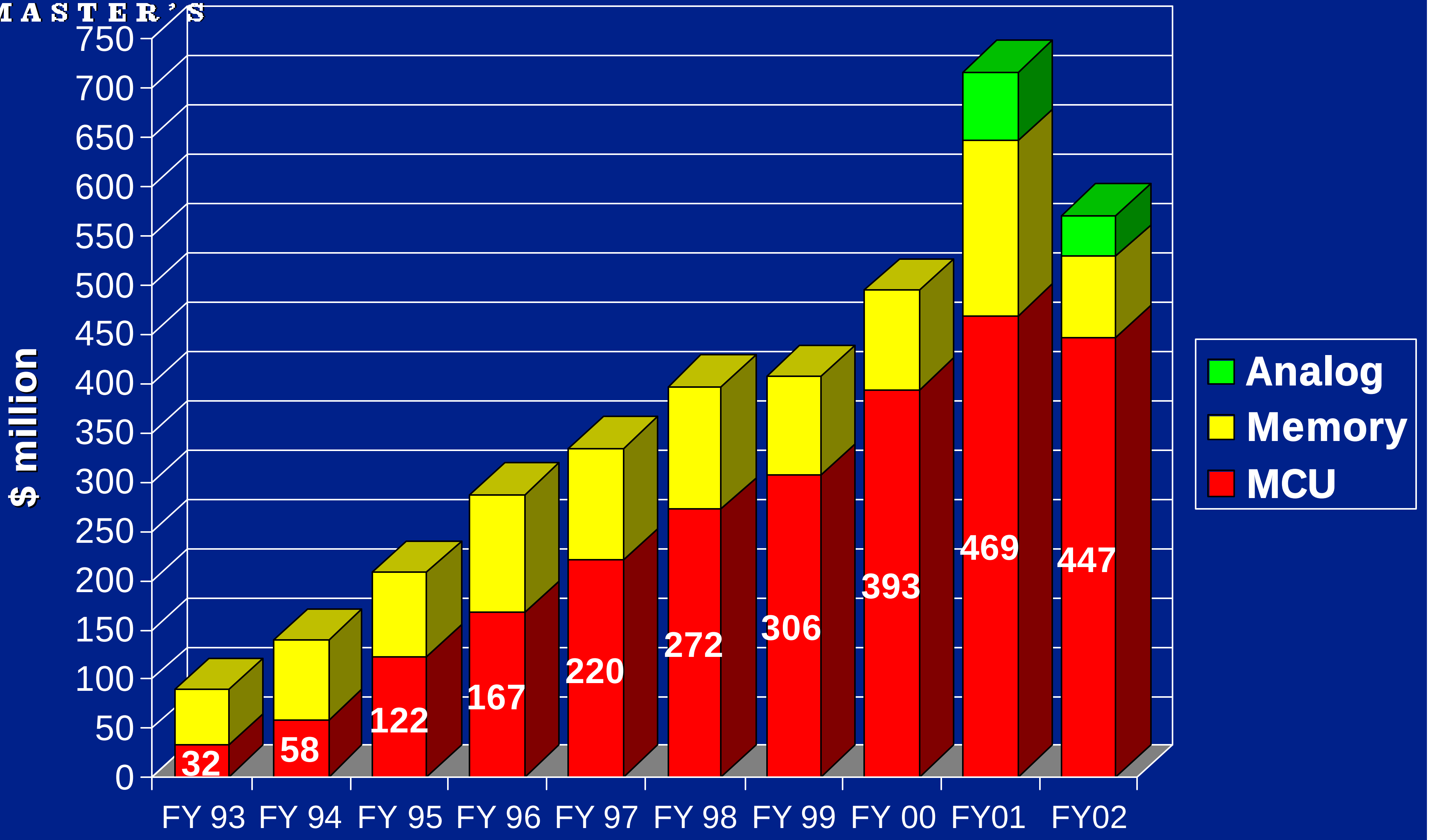
- Leading semiconductor manufacturer:
  - of high-performance, field-programmable 8-bit & 16-bit RISC Microcontrollers
  - of Analog & Interface products
  - of related Memory products
  - for high-volume embedded control applications
- $572 million in product sales in FY02
- More than 3,000 employees
- Headquartered near Phoenix in Chandler, AZ

"The Silicon Desert"

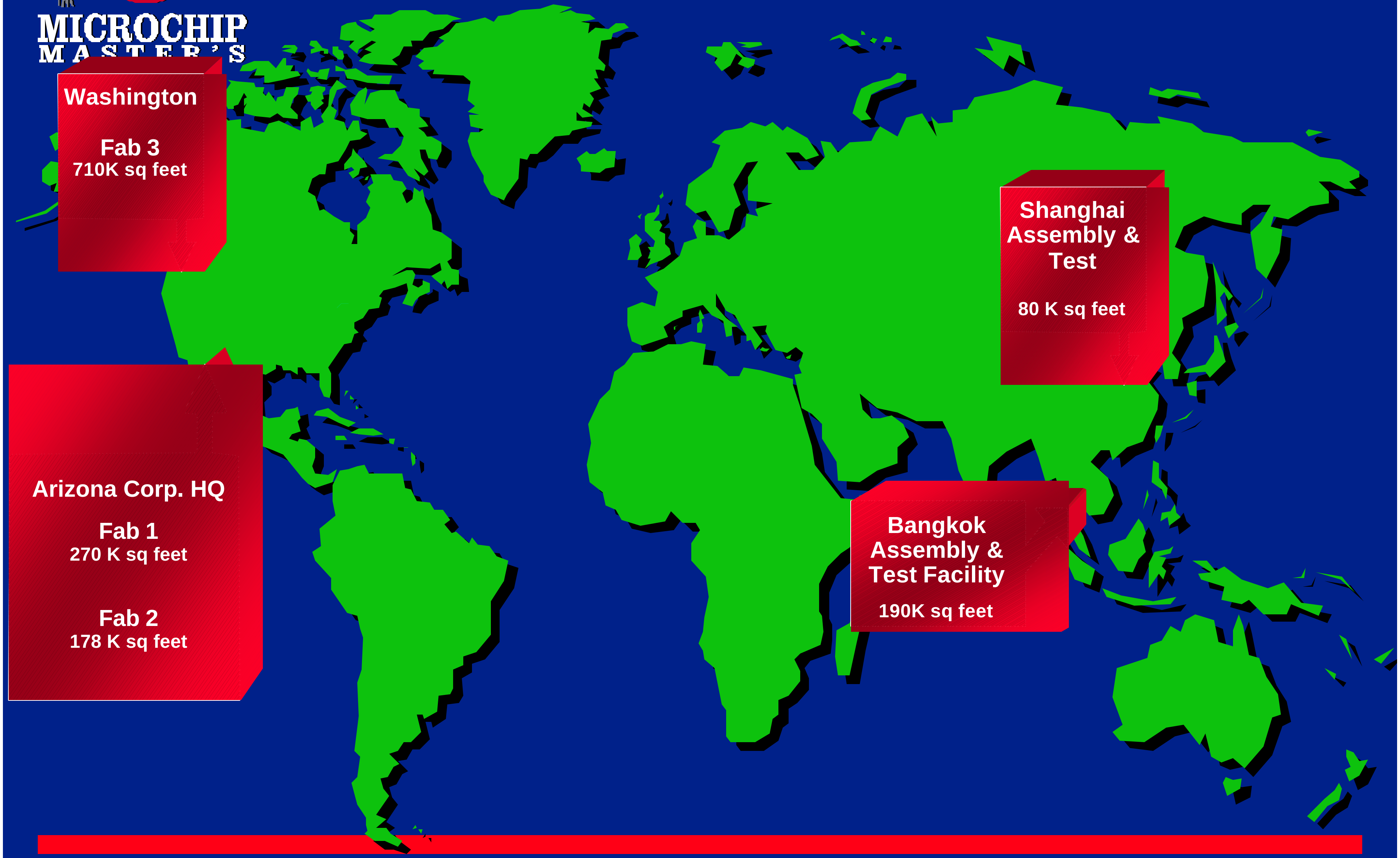
History of the PICmicro® Microcontroller

1989	Pioneered field-programmable MCU: PIC16C5X family
1990	Shipped 1 millionth OTP PICmicro® device
1991	Introduced MPLAB® IDE -- the world's first Windows 3.0 based development system
1992	Offered ROM program memory to PICmicro customer base
1994	Introduced *Enhanced* FLASH PICmicro MCUs
1996	Introduced the world's first 8-pin microcontrollers
	Ranked #5 in 8-bit MCU market share
1997	Achieved #2 ranking in 8-bit MCU market share
1999	Introduced PIC18CXXX enhanced core architecture
	Shipped 1 billionth PICmicro MCU
2000	Announced comprehensive FLASH PICmicro product roadmap
2001	Shipped 200,000th development system
2002	Shipped 2 billionth PICmicro MCU

Annual Net Sales Growth



Worldwide Manufacturing Locations



Washington

Fab 3
710K sq feet

Arizona Corp. HQ

Fab 1
270 K sq feet

Fab 2
178 K sq feet

Shanghai Assembly & Test

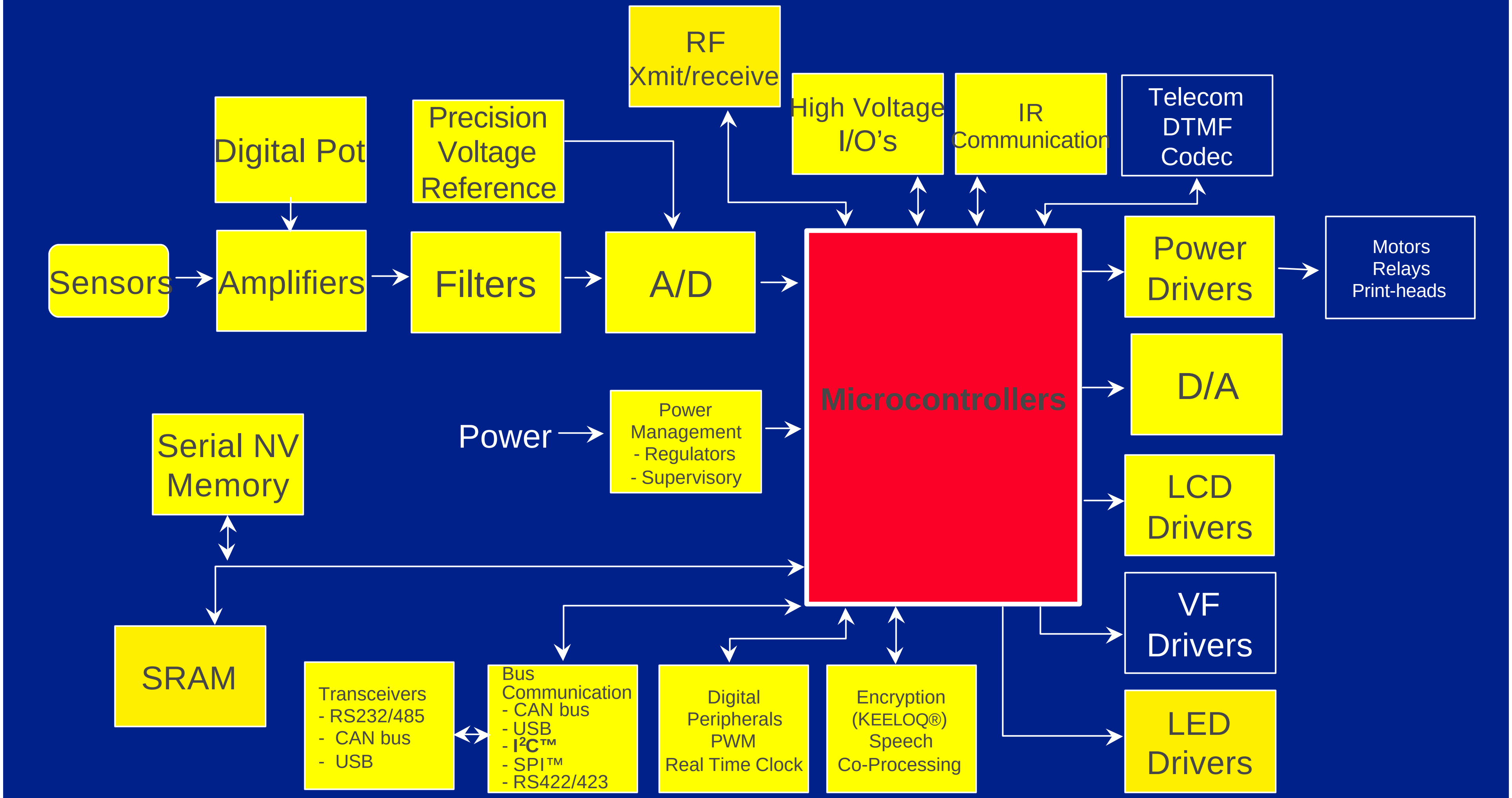
80 K sq feet

Bangkok Assembly & Test Facility

190K sq feet

MICROCHIP MASTER'S

Existing PICmicro® MCU Core and Peripheral Blocks



RF Xmit/receive

High Voltage I/O's

IR Communication

Telecom DTMF Codec

Digital Pot

Precision Voltage Reference

Sensors

Amplifiers

Filters

A/D

Microcontrollers

Power Drivers

Motors
Relays
Print-heads

D/A

Power

Power Management
- Regulators
- Supervisory

Serial NV Memory

LCD Drivers

VF Drivers

SRAM

Transceivers
- RS232/485
- CAN bus
- USB

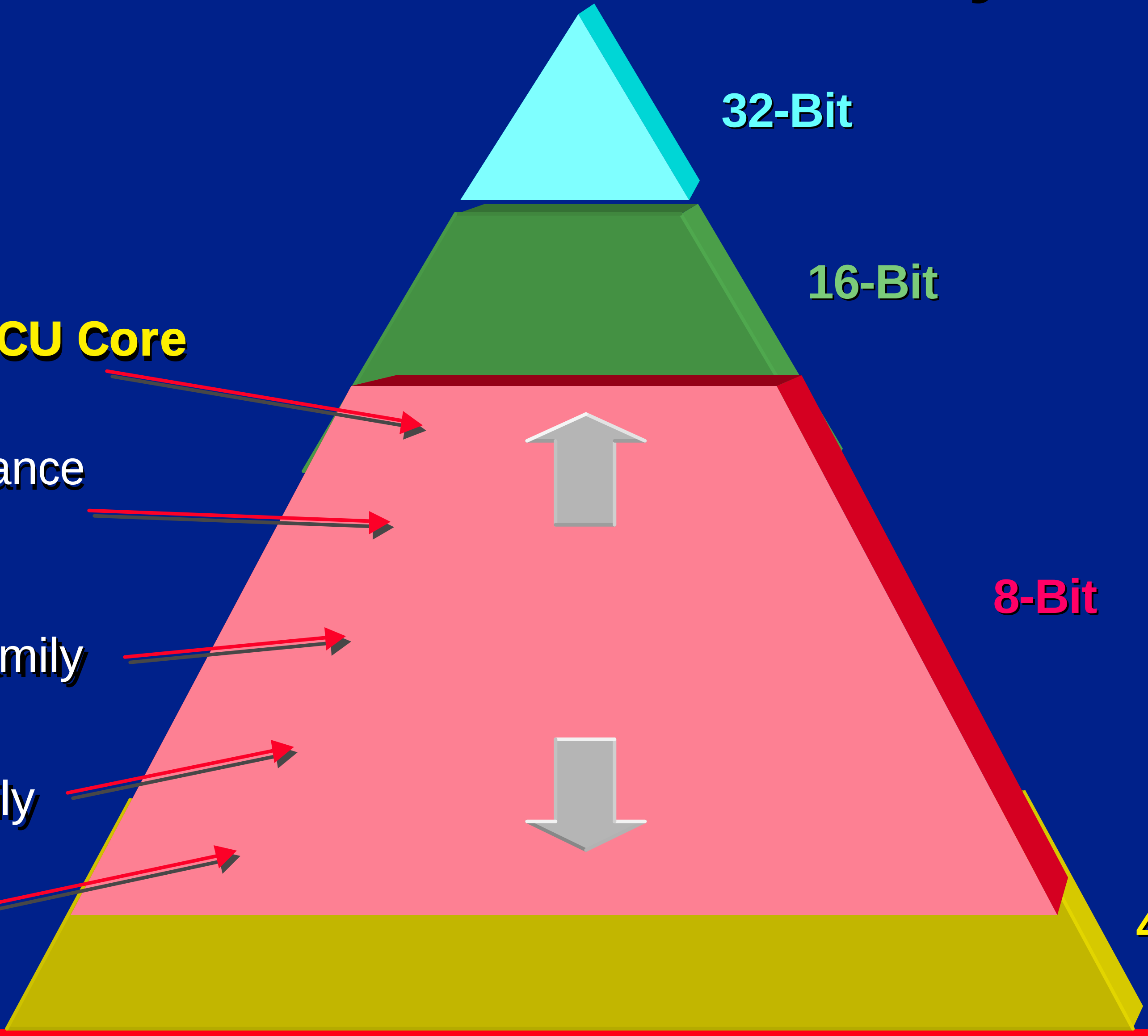
Bus Communication
- CAN bus
- USB
- I^{2}C™
- SPI™
- RS422/423

Digital Peripherals
PWM
Real Time Clock

Encryption
(KEELOQ®)
Speech
Co-Processing

LED Drivers

Microcontroller Market Pyramid



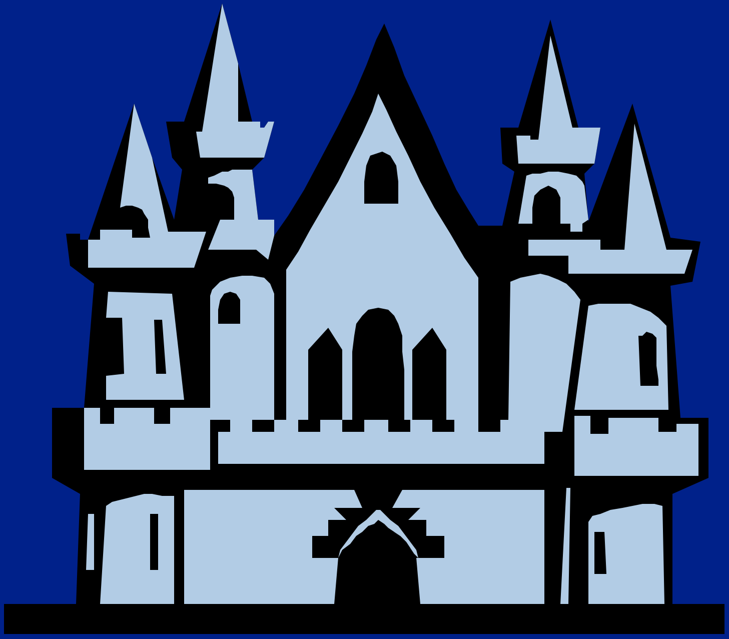
PIC18CXXX
Enhanced MCU Core

PIC17CXXX
High-Performance Family

PIC16CXX
Mid-Range Family

PIC16C5X
Baseline Family

PIC12CXXX
8-Pin Family



PIC18 Architecture And Peripherals

PIC18 Architecture
Features

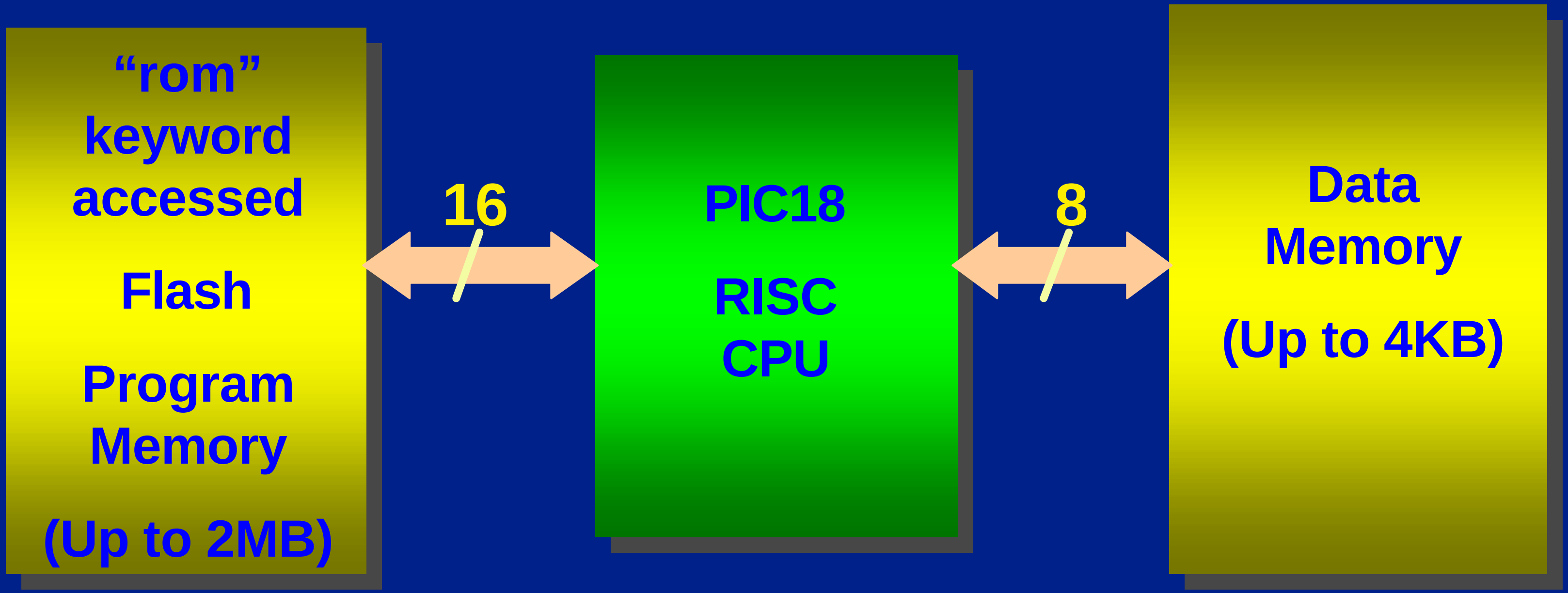
- High Performance 8-bit RISC CPU
- 40 MHz / 10 MIPs sustained operation
- 2.0V to 5.5V operation
- Linear Program Memory addressing to 2MB
- Linear Data Memory addressing to 4KB
- 3 Data Pointers with 5 addressing modes
- Relative conditional branch instructions

PIC18 Architecture
Features (Continued)

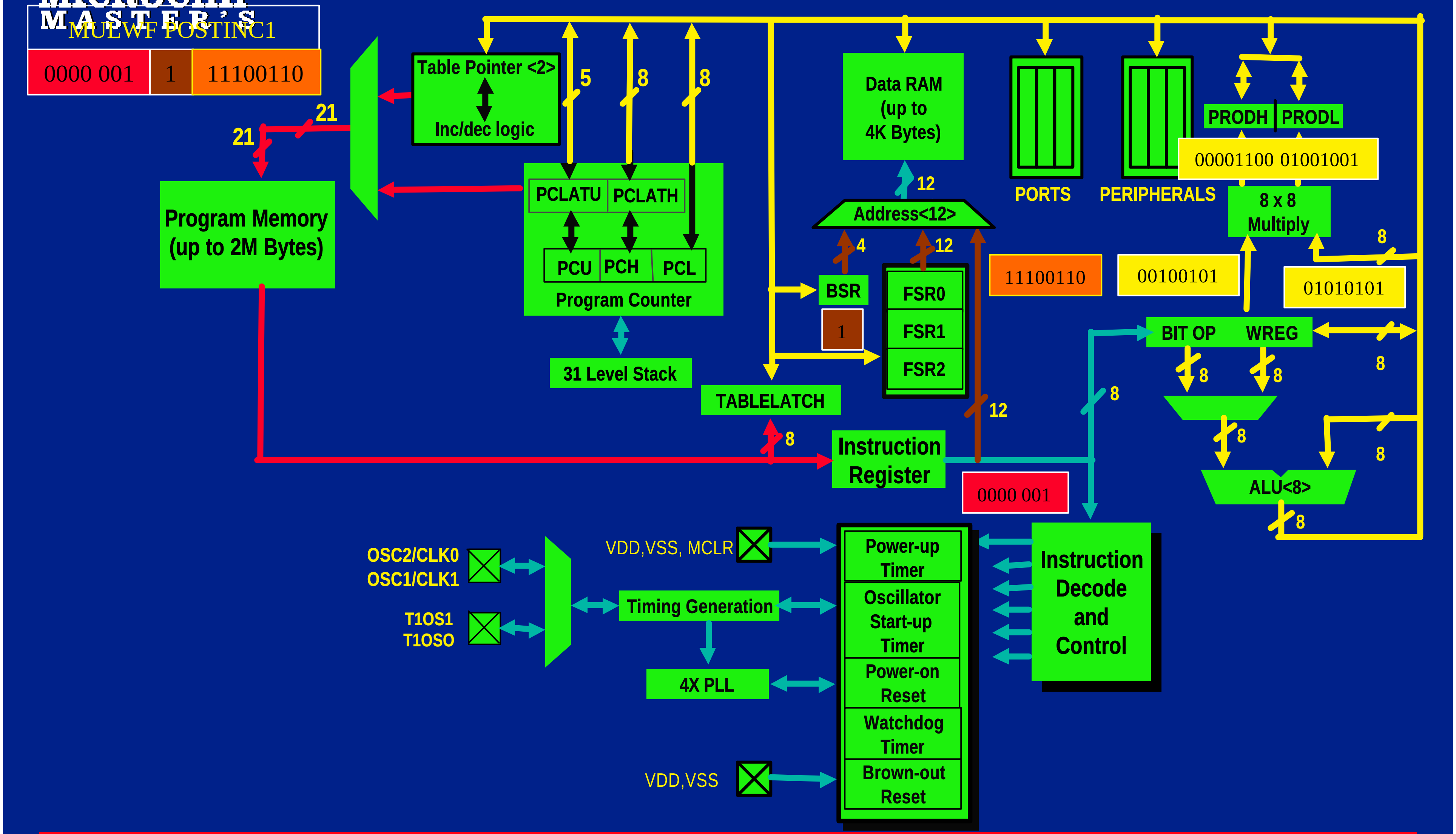
- Up to 10MIPS @ 10MHz with 4X PLL
- Enhanced Flash memory
  - 2 Seconds Programming Time
  - Low Cost MPLAB-ICD-II Support
  - Flexible Program Memory Protection
- And Many More...

PIC18 Architecture
Harvard Architecture

- Separate memory spaces for instructions and data
  - Increased throughput
  - Different program and data bus widths are possible



PIC18 Block Diagram



PIC18 Architecture

Oscillator

- Various oscillator modes

LP	Low Power Crystal (200KHz max)
XT	Crystal/Resonator (4MHz max)
HS	High Speed Crystal/Resonator (40MHz max)
HS + PLL	HS + 4X PLL (10MHz max)
RC	External RC (4MHz max)
RCIO	RC with OSC2 as I/O (4MHz max)
EC	External Clock (40MHz max)
ECIO	EC with OSC2 as I/O (40MHz max)
INTOSC	Internal RC Oscillator (30/500 kHz, 1/4/8 MHz)

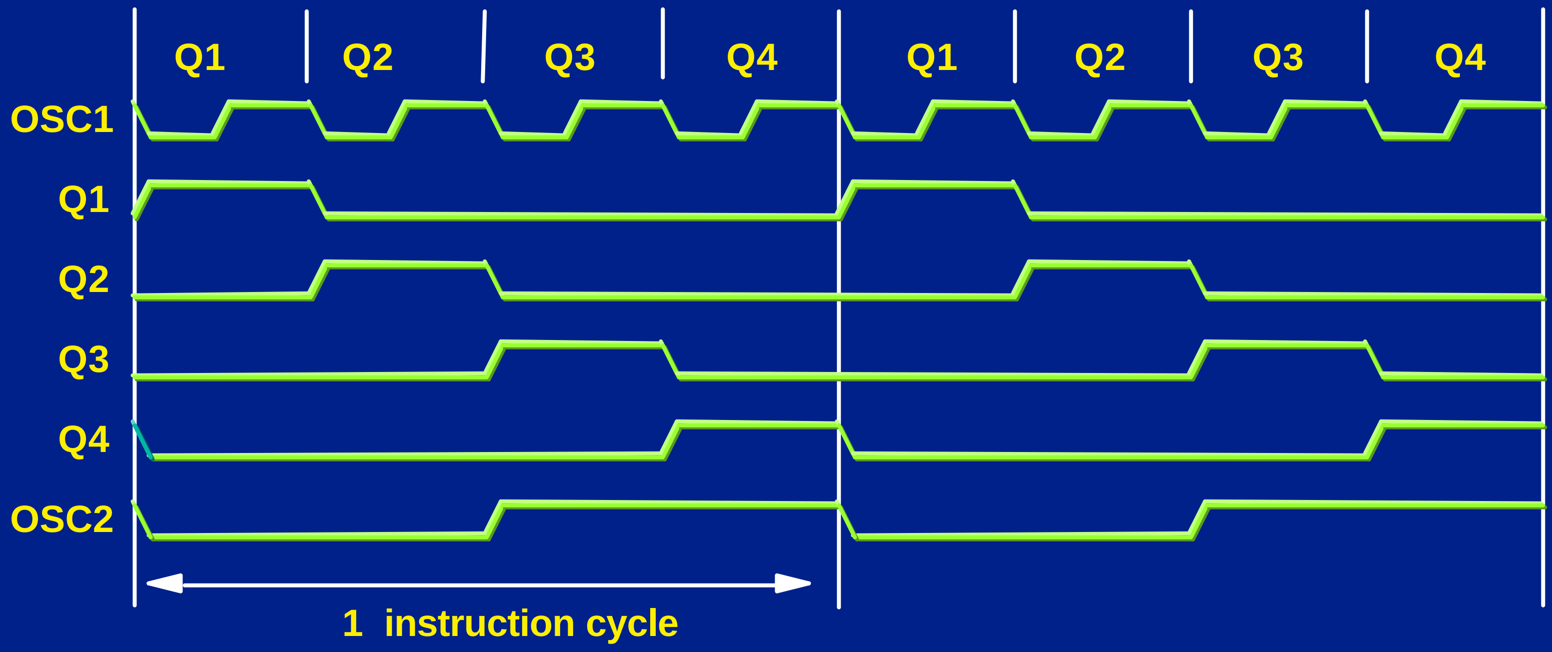
Secondary Oscillator Mode

Modes selected by Configuration registers

PIC18 Architecture

Clocking Scheme

- Instruction cycle = 1/4 of clock input frequency
- 100 ns Instruction cycle at 40 MHz clock



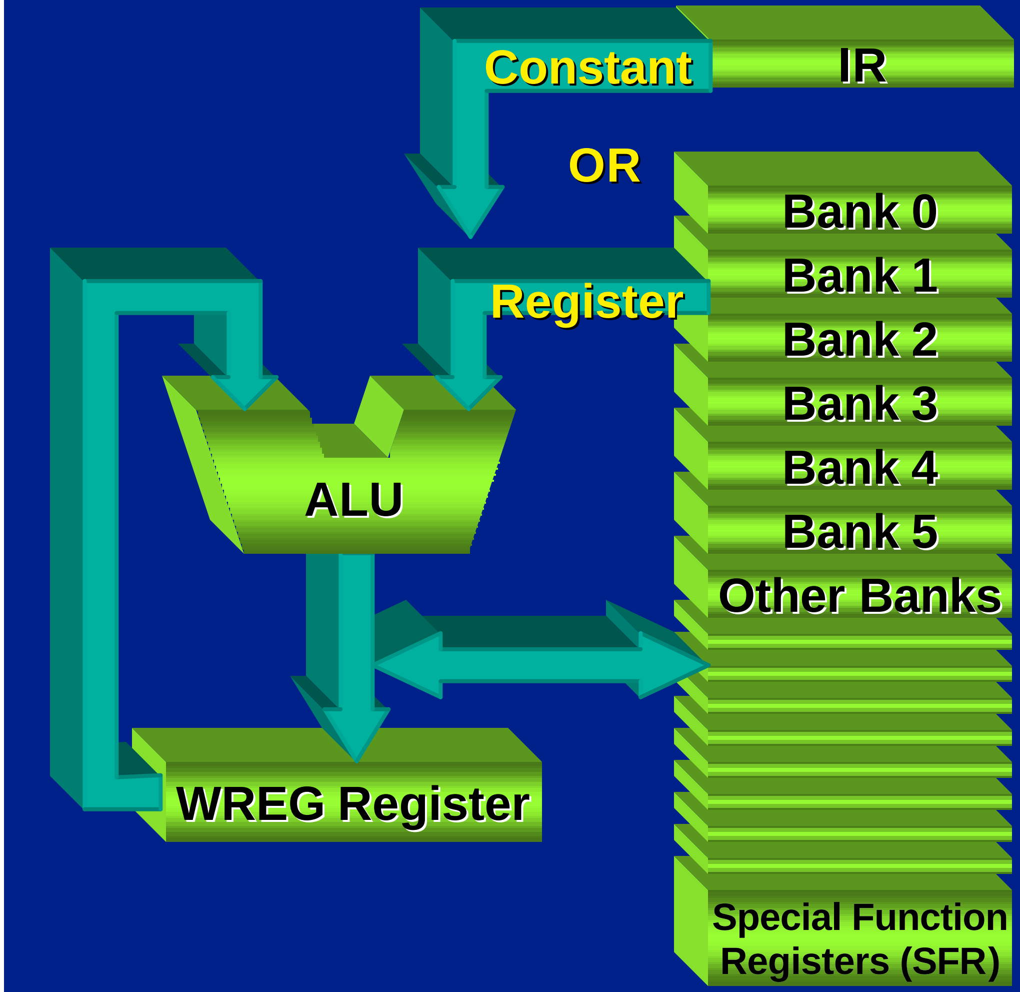
PIC18 Architecture
Instruction Pipeline

- Allows overlap of fetch and execution
- Makes single cycle execution
- Program branches (e.g. **GOTO**, **CALL** or Write to PC) take two or three cycles

	T_{CY0}	T_{CY1}	T_{CY2}	T_{CY3}	T_{CY4}
1. **MOVWF PORTB**	Fetch 1	Exec. 1			
2. **RCALL SUB_1**		Fetch 2	Exec. 2		
2b. ***Forced NOP***			Fetch 2b	Forced NOP	
3. **BSF PORTA, RA3**				Fetch SUB_1	Exec. SUB_1
					Fetch SUB_1+1

PIC18 Architecture

ALU



- Operates on WREG and a Register or Constant
- Multi-Byte calculation using **ADDWFC** etc.

PIC18 Architecture

8 x 8 Hardware Multiplier

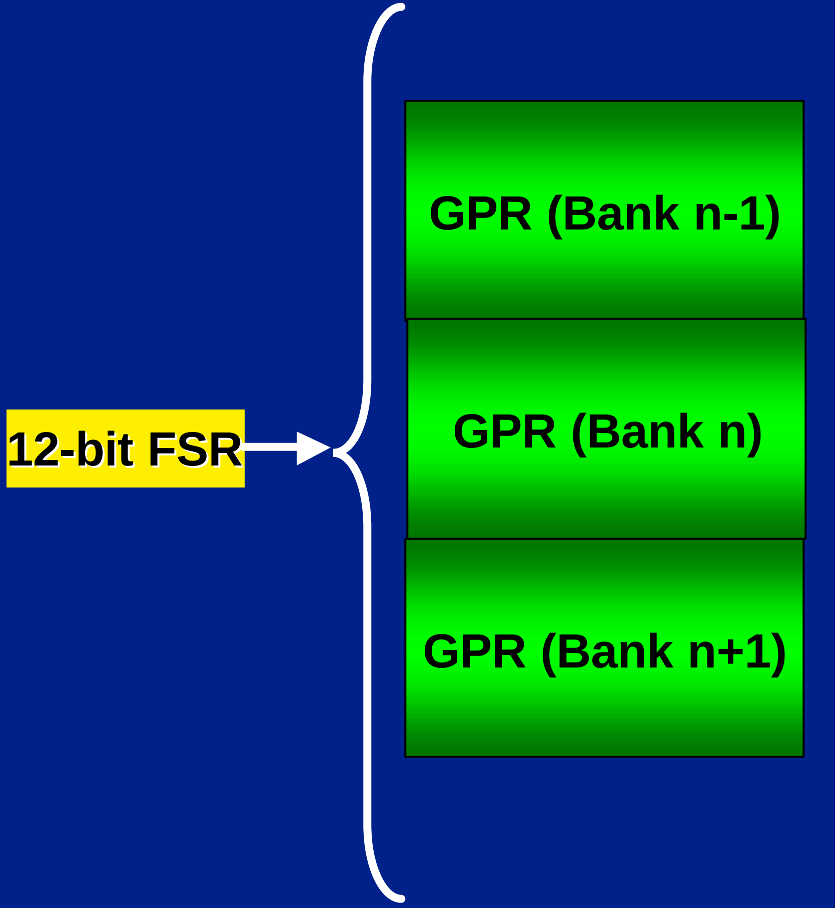
- Single Cycle Hardware Multiplier
- Performs
  - WREG X Register
  - WREG X Constant
- 16-bit result stored in PRODH:PRODL
- Integer arithmetic operation
- Unsigned operation

PIC18 Architecture
Computation Performance

Function	Prog Words (estimated)	RAM (estimated)	Max Time (uS) @ 10MIPS
8 x 8 unsigned multiply	1	-	0.1
16 X 16 unsigned multiply	30	7	3
16 X 16 signed multiply	40	8	4
32 x 32 signed multiply	140	18	15
32 / 16 signed divide	450	9	42
Float Add (IEEE 32bit)	320	12	7
Float Mul (IEEE 32bit)	350	13	10
Float Div (IEEE 32bit)	130	14	32
Sqrt (32bit)	320	10	57
Sin (32bit)	420	11	241

PIC18 Architecture
Indirect Access

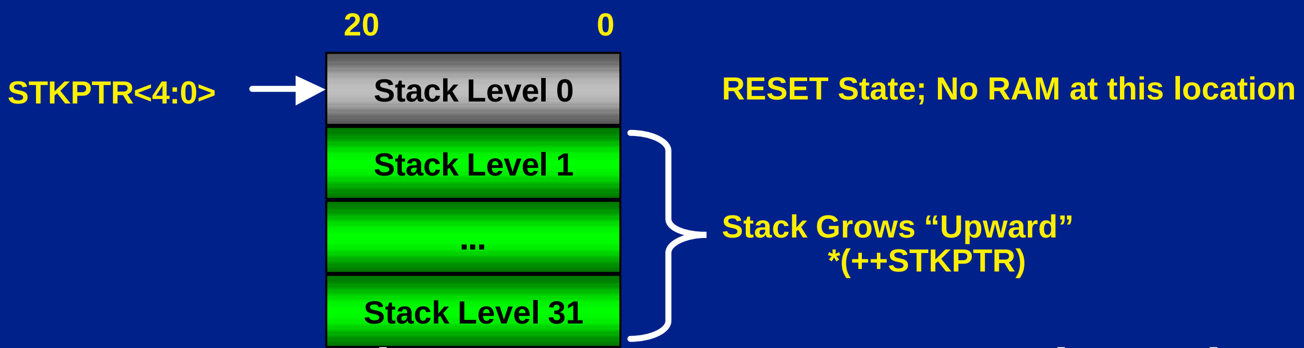
- Indirect Addressing
  - Three 12-bit FSRs
  - FSRnH:FSRnL (0 ≤ n ≤ 2)
- Linear access to 4KB
- Special Instruction to load FSRn in 2 cycles
- De-reference operations
  - Unchanged
  - Pre/Post Increment
  - Post Decrement
  - Indexed by WREG (signed)



PIC18 Architecture

Stack Memory

- Hardware stack - 31 levels deep
  - Separate memory, pointed by STKPTR
  - Used by **`CALL`**, **`RCALL`**, INT, **`RETURN`**, **`RETFIE`**

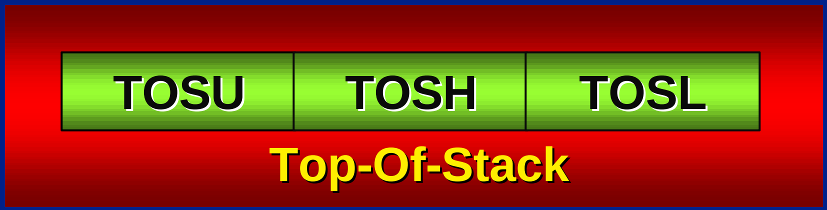


- Software stack uses FSRn, not hardware stack
  - Uses general purpose RAM, pointed by FSRn
  - Used to store local variables for re-entrant functions

PIC18 Architecture

Accessing HW Stack

- 5-bit Stack Ptr addresses 21-bit wide stack
- Top-Of-Stack = TOSU:TOSH:TOSL
  - Readable & Writeable => RTOS Friendly
- **PUSH** puts current PC on Top-Of-Stack
- **POP** discards Top-Of-Stack
- When enabled, Stack OV resets the device
- Stack Underflow returns 00000h



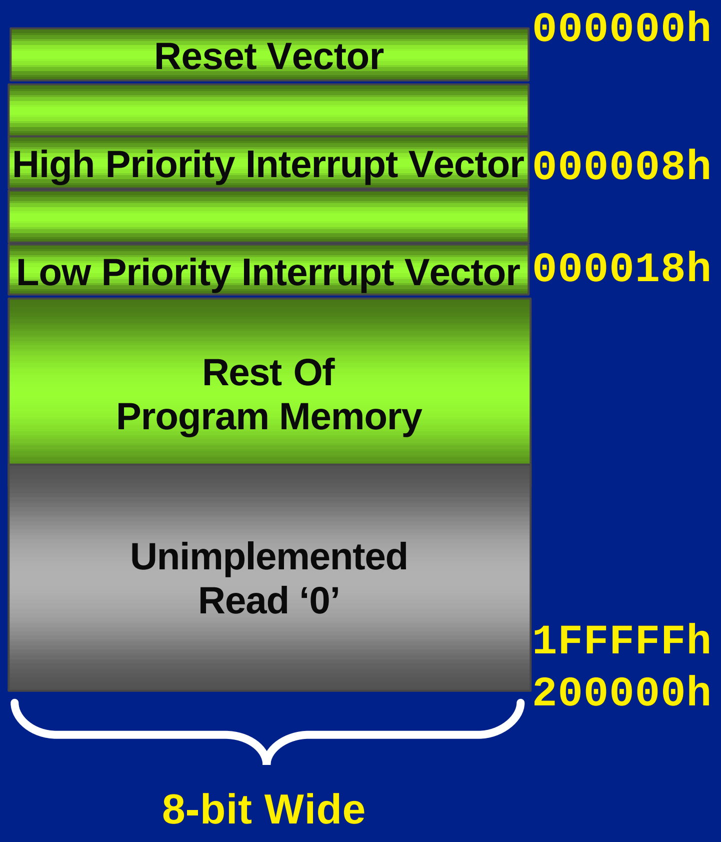
TOSU TOSH TOSL

Top-Of-Stack

PIC18 Architecture

Program Memory

- Up to 2M x 8 in size*
- Linear access
- Two Interrupt Vectors
- Self programmable*
- Programmable over entire voltage range
- Flexible Code Protection Modes*
- 100 K erase/writes (typical)*
- > 40 years retention (typical)

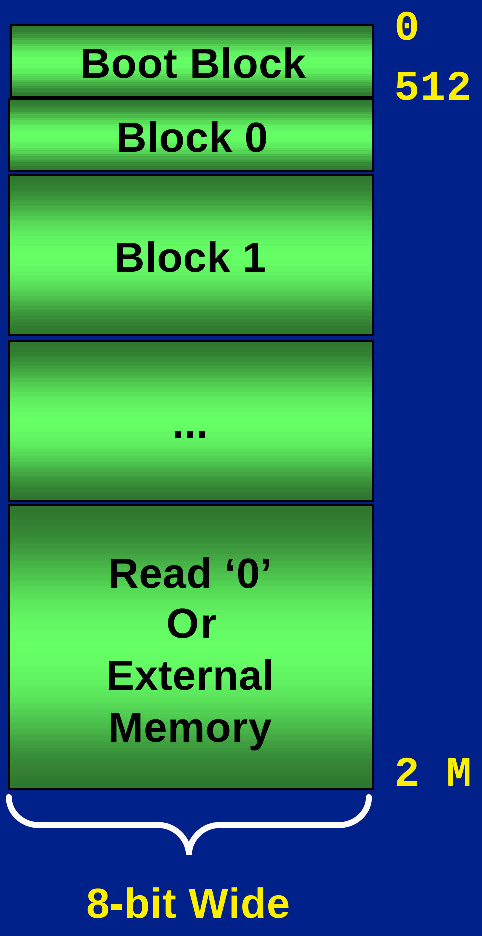


* Note: Check your device datasheet

PIC18 Architecture

Program Memory Organization

- Divided into blocks
- 512 bytes of Boot block*
- Block size varies by device
  - 8KB on PIC18F452
- Blocks erased in bulk or 64* bytes
  - Bulk erase in ICSP™ programming mode (4.5 - 5.5V)
- Code protection by block
- Internal Read/Write protection by block



0

Boot Block

512

Block 0

Block 1

...

Read ‘0’ Or External Memory

2 M

8-bit Wide

* Note: Check your device datasheet

PIC18 Architecture

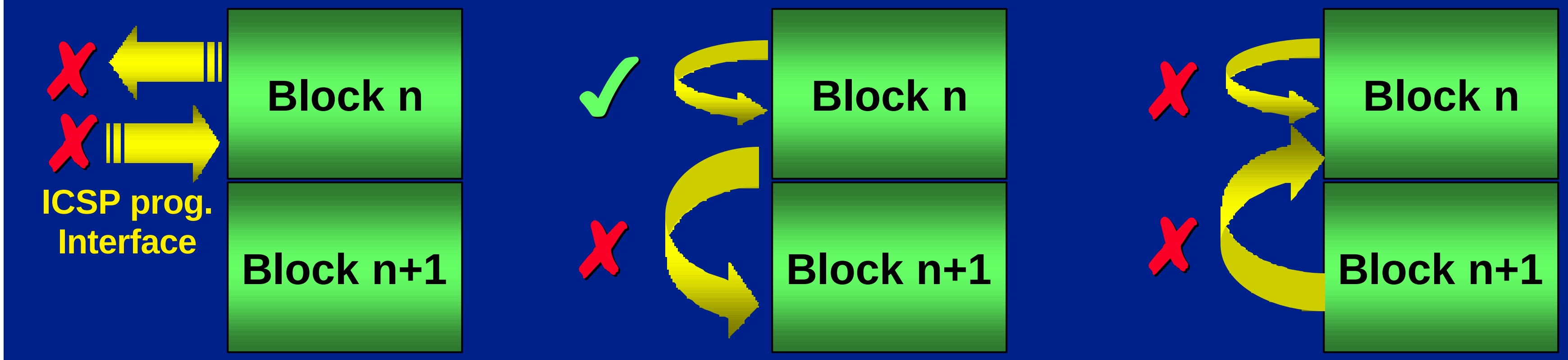
Program Memory : Protection

Three types of Protection Scheme:

Code Protection

Internal Read Protection

Internal Write Protection



ICSP programming mode
Read and Write disabled

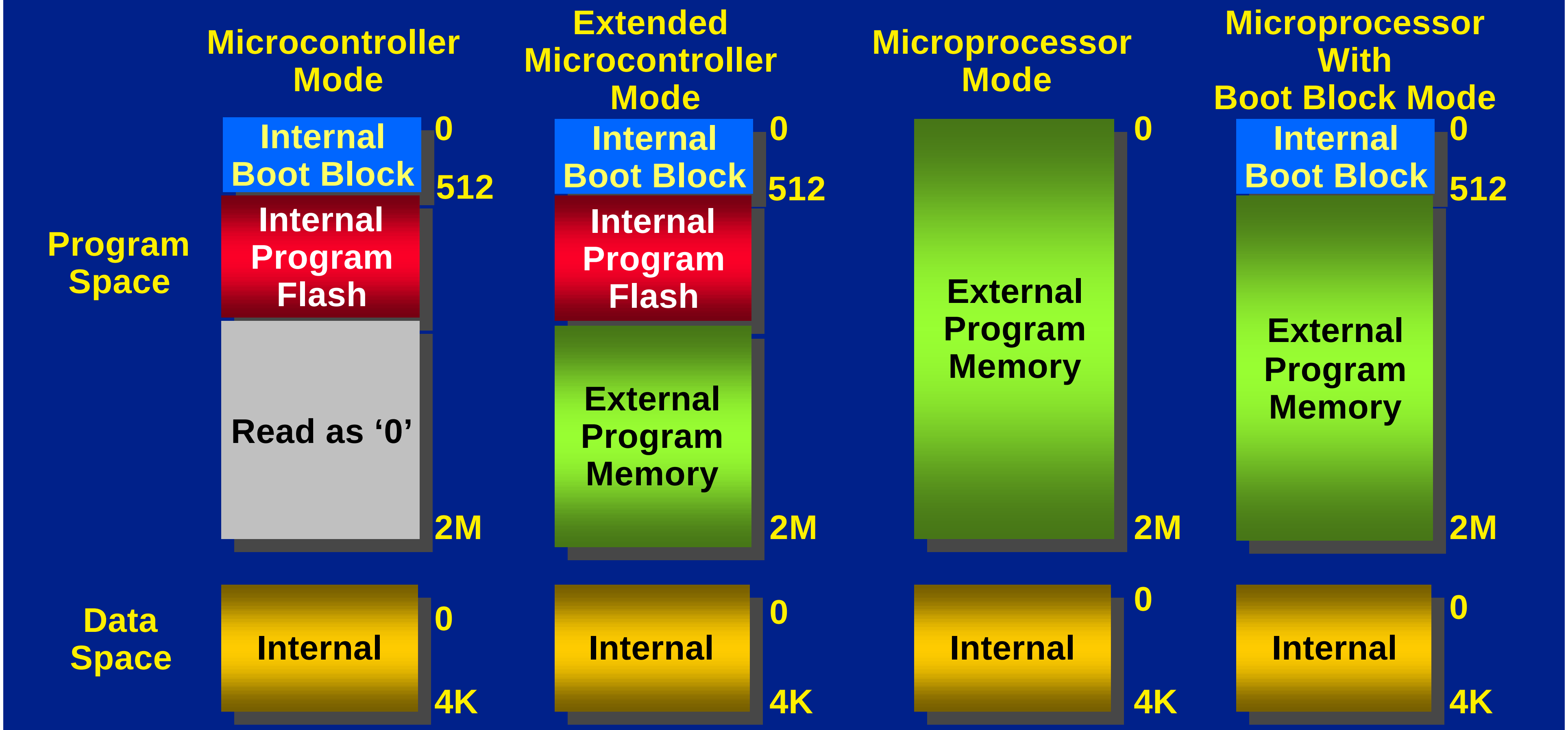
Reads from same block OK,
reads from other blocks disabled

Self Write to this block are disabled

PIC18 Architecture

Program Memory Modes

Four Modes:



Program Space

Data Space

Microcontroller Mode

- Internal Boot Block (0 to 512)
- Internal Program Flash
- Read as ‘0’ (to 2M)
- Data: Internal (0 to 4K)

Extended Microcontroller Mode

- Internal Boot Block (0 to 512)
- Internal Program Flash
- External Program Memory (to 2M)
- Data: Internal (0 to 4K)

Microprocessor Mode

- External Program Memory (0 to 2M)
- Data: Internal (0 to 4K)

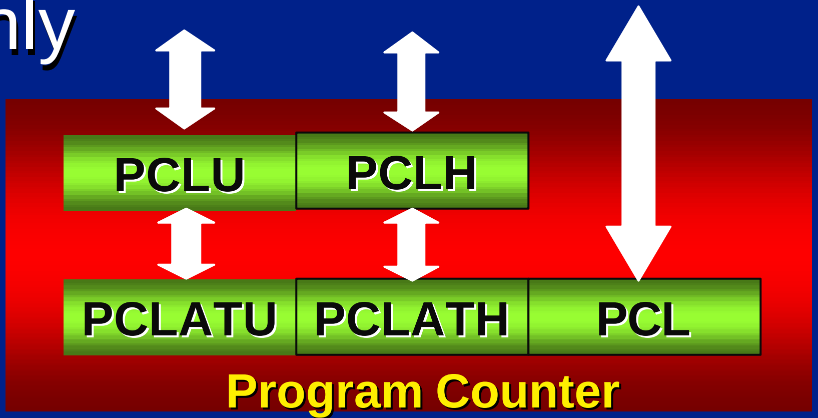
Microprocessor With Boot Block Mode

- Internal Boot Block (0 to 512)
- External Program Memory (to 2M)
- Data: Internal (0 to 4K)

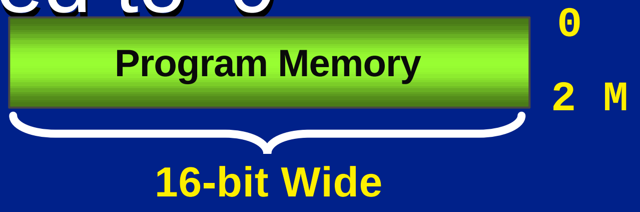
Note: Check your device datasheet

PIC18 Architecture
Accessing Program Memory

- 21-bit Divided into PCU:PCH:PCL
    - PCL is readable/writeable
    - PCU:PCH is readable/writeable via shadow registers only



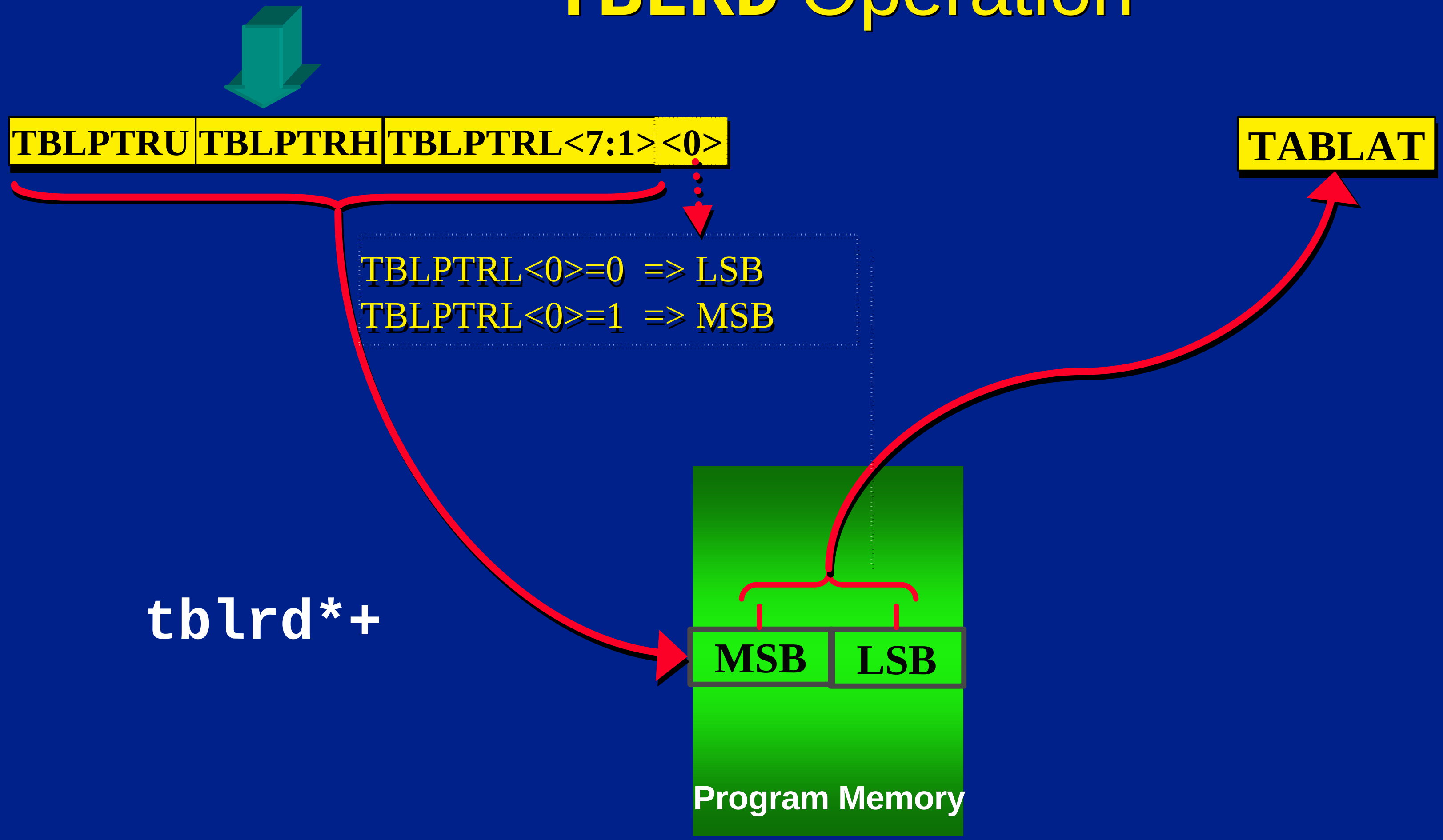
- PCL<0> is forced to ‘0’



PIC18 Architecture

Reading Program Memory

TBLRD Operation



TBLPTRL<0>=0 => LSB
TBLPTRL<0>=1 => MSB

```
tblrd*+
```

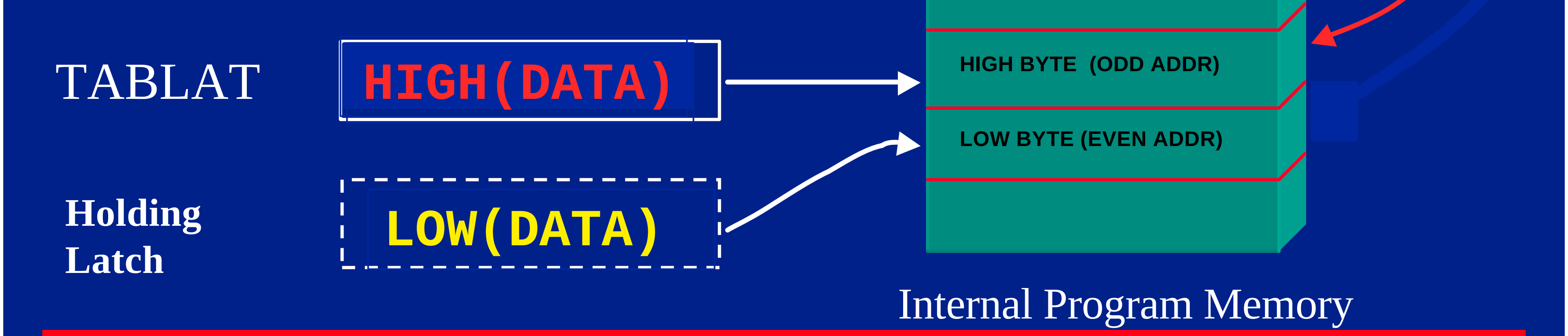
PIC18 Architecture

Writing to Program Memory

Table Pointer | TBLPTRU | TBLPTRH | TBLPTRL

```
movff  LOW(DATA),TABLAT
tblwt*+
movff  HIGH(DATA),TABLAT
tblwt*
```

See Appendix C for more information



Internal Program Memory

PIC18 Architecture
Accessing Program Memory (Cont.)

- **TBLPTR** is used to address program memory
  - Divided in TBLPTRU:TRBLPTRH:TBLPTRL
- **TBLRD** is used to read a byte
- **TBLWT** is used to load write buffer
  - EECON1 register controls actual write cycle
  - Protected against “run-away” code
- Erase block size 32 or 64 bytes*
- 8 bytes written at a time

`* Note: Check your device datasheet`

Table Pointer Operations

- To enhance flexibility of table operations, the TBLPTR automatically increment and decrement during read/write operations
- PIC18 devices have 4 modify modes for TBLPTR

tblwt*	**tblrd***	no change
tblwt*+	**tblrd*+**	auto post increment
tblwt*-	**tblrd*-**	auto post decrement
tblwt+*	**tblrd+***	auto pre increment

PIC18 Architecture

Data EEPROM

- Size ranges from 64 to 1024 bytes
- 1 M erase/write cycles (typical)
- > 40 years retention (typical)
- Read and Written at byte boundary
  - Automatic Erase-Before-Write
- Protection against “run-away” code
- Code Protection And Internal Write Protection
- Accessed via EEADR, EEDATA and EECONn registers

PIC18 Architecture
Configuration

- Configuration Registers at **300000h**
- Bit(s) enable/define mode(s)
- Written one byte at a time
- Writeable in all modes
  - Special “Configuration Write Protect” bit
- Most bits can be written to either ‘**1**’ or ‘**0**’
  - Code, Read and Write Protection bits can be written ‘**1**’ -> ‘**0**’ only
  - Bulk Erase required to reset Code, Read and Write Protection bits to a ‘**1**’

Specifying Configuration Information in Source File

- Create “config.asm” file and include in project:

```
#include p18f452.inc
__CONFIG _CONFIG1L,0xFF
__CONFIG _CONFIG1H,_OSCS_OFF_1H&_HSPLL_OSC_1H
__CONFIG _CONFIG2L,_BOR_OFF_2L&_BORV_20_2L&_PWRT_OFF_2L
__CONFIG _CONFIG2H,_WDT_OFF_2H&_WDTPS_128_2H
__CONFIG _CONFIG3L,0xFF
__CONFIG _CONFIG3H,_CCP2MX_OFF_3H
__CONFIG _CONFIG4L,_STVR_ON_4L&_LVP_OFF_4L&_DEBUG_OFF_4L
__CONFIG _CONFIG4H,0xFF
__CONFIG _CONFIG5L,_CP0_OFF_5L&_CP1_OFF_5L&_CP2_OFF_5L&_CP3_OFF_5L
__CONFIG _CONFIG5H,_CPB_OFF_5H&_CPD_OFF_5H
__CONFIG _CONFIG6L,_WRT0_OFF_6L&_WRT1_OFF_6L&_WRT2_OFF_6L&_WRT3_OFF_6L
__CONFIG _CONFIG6H,_WRTC_OFF_6H&_WRTB_OFF_6H&_WRTD_OFF_6H
__CONFIG _CONFIG7L,_EBTR0_OFF_7L&_EBTR1_OFF_7L&_EBTR2_OFF_7L&_EBTR3_OFF_
__CONFIG _CONFIG7H,_EBTRB_OFF_7H
END
```

C Programmer's Interface

Accessing Peripheral Control and Status Bits

- All peripheral control bits set up in <processor>.h file as:

<peripheral register name>bits.<bit name>

- Example:
  - GIEH bit of INTCON can be accessed by:

INTCONbits.GIEH

Reset Vector

- Located at 0x00000, compiler automatically initializes variables
- Calls main() after variable initialization
- Loops back and calls main() again if main exits
- Generally, main() should stay in loop and not exit:

```
void main(void){
     // Place your initialization code here

  while(1){
          // Place your main loop here
          }
  }
```

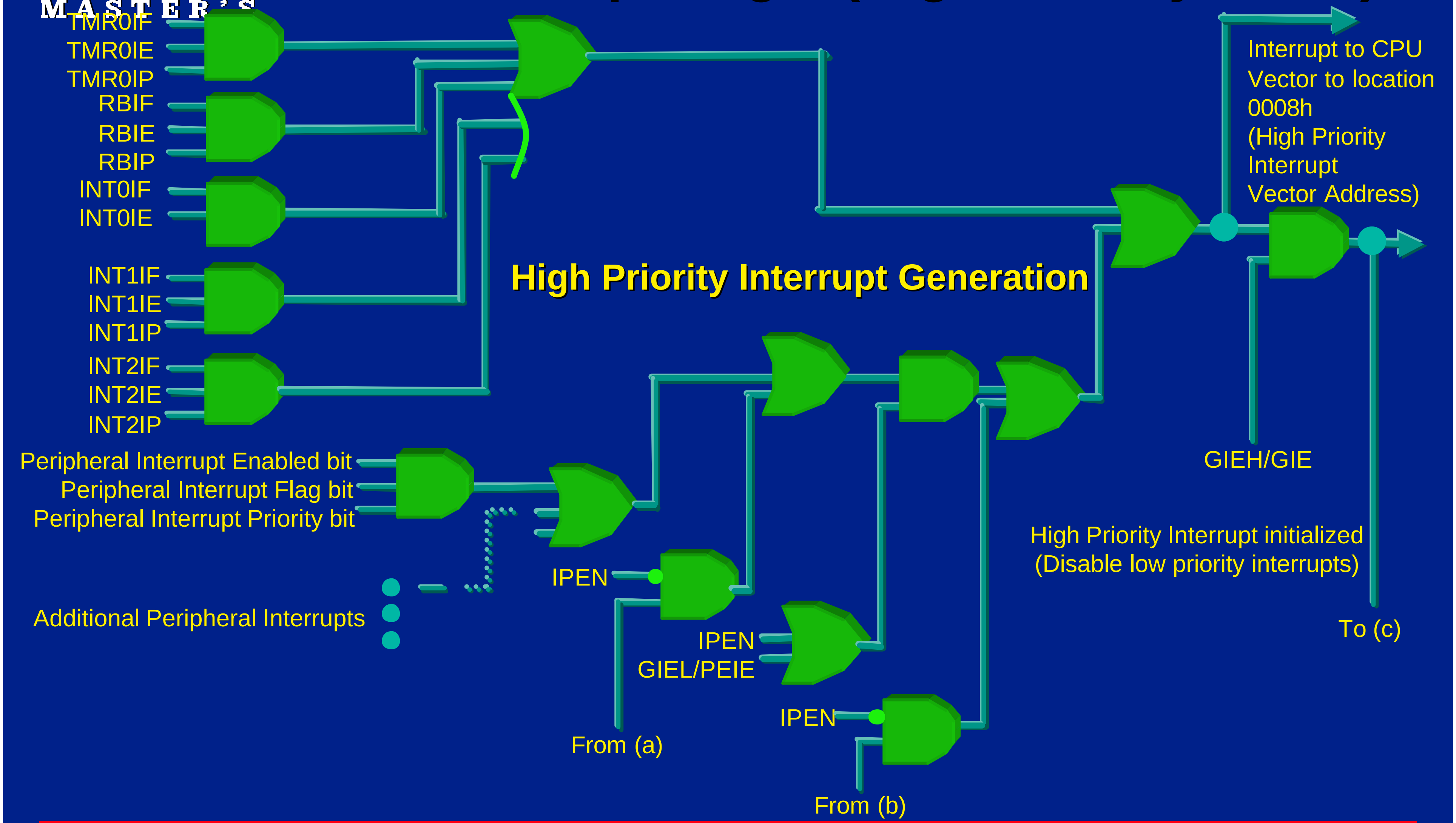
PIC18 Architecture

Interrupt Overview

- Interrupt Sources can individually
  - Assigned to high or low priority vector
    - High Priority Vector at **000008h** (Default)
    - Low Priority Vector at **000018h**
  - Polled or interrupt driven
- Automatic context save WREG, STATUS and BSR on High Priority Interrupt
- Most interrupts wake processor from sleep
- Fixed interrupt latency is three instruction cycles

PIC18 Architecture

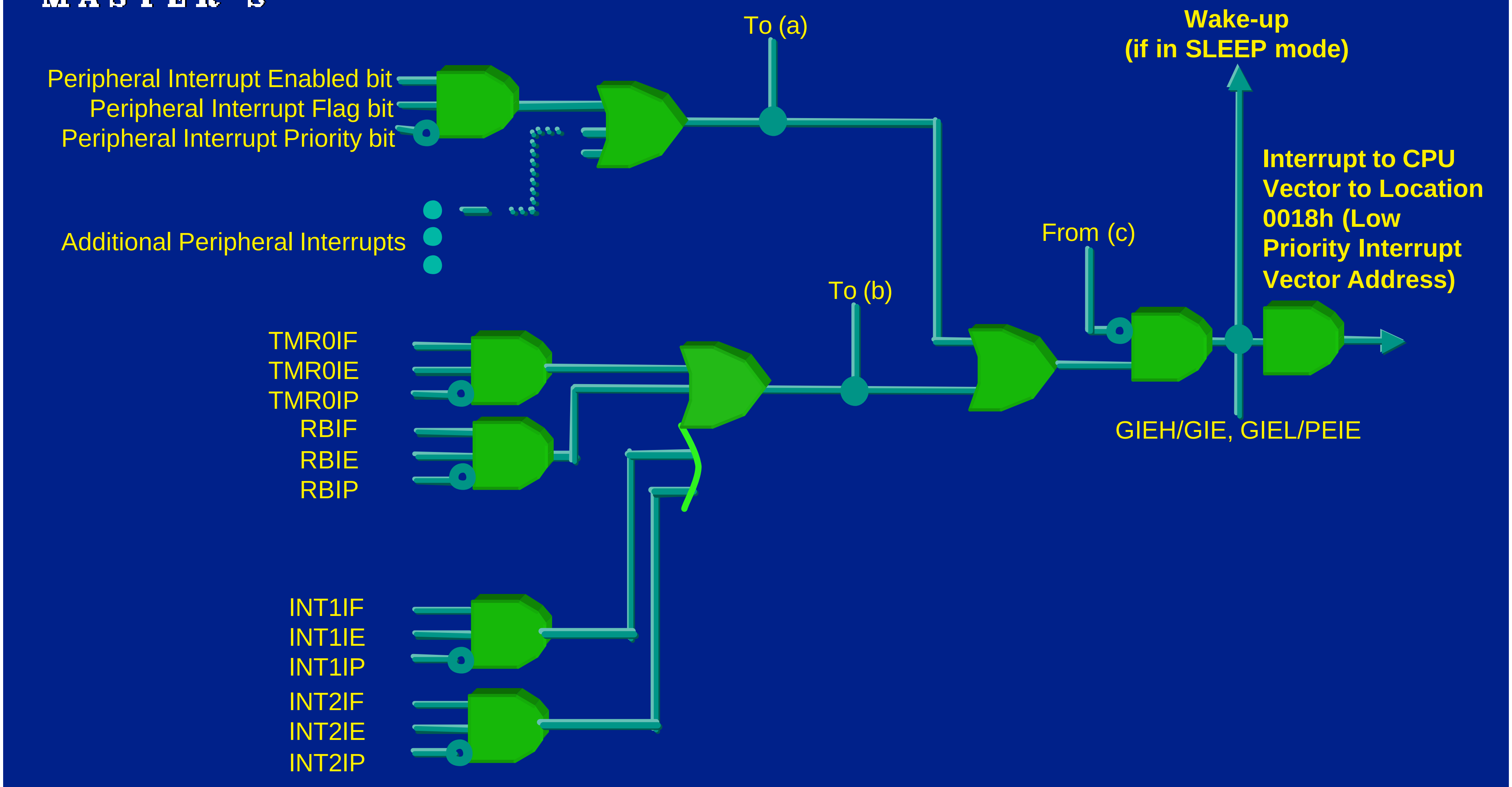
Interrupt Logic (High Priority Level)



TMR0IF
TMR0IE
TMR0IP
RBIF
RBIE
RBIP
INT0IF
INT0IE
INT1IF
INT1IE
INT1IP
INT2IF
INT2IE
INT2IP
Peripheral Interrupt Enabled bit
Peripheral Interrupt Flag bit
Peripheral Interrupt Priority bit
Additional Peripheral Interrupts
IPEN
IPEN
GIEL/PEIE
IPEN
From (a)
From (b)
High Priority Interrupt Generation
Interrupt to CPU
Vector to location
0008h
(High Priority
Interrupt
Vector Address)
GIEH/GIE
High Priority Interrupt initialized
(Disable low priority interrupts)
To (c)

PIC18 Architecture

Interrupt Logic (Low Priority Level)



Interrupt Priority Enable

- New bit added to the RCON register - IPEN

R/W-0	R/W-0	U-0	R/W-1	R/W-1	R/W-1	R/W-0	R/W-0
IPEN	LWRT	-	RI	TO	PD	POR	BOR
bit7	6	5	4	3	2	1	0

- Enables / Disables Interrupt Priority and 16C Compatibility
  - If IPEN=0, priority is disabled and the interrupts are compatible with 16C (default)
  - If IPEN=1, priority is enabled and the interrupts are NOT compatible with 16C

- Registers have been added to set priority for each interrupt source, except INT0.

Peripheral Interrupt Control Registers

	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
	bit7	6	5	4	3	2	1	0

	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
	bit7	6	5	4	3	2	1	0

	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
IPR1	PSPIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP
	bit7	6	5	4	3	2	1	0

	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
PIR2	-	-	-	-	BCLIF	LVDIF	TMR3IF	CCP2IF
	bit7	6	5	4	3	2	1	0

	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
PIE2	-	-	-	-	BCLIE	LVDIE	TMR3IE	CCP2IE
	bit7	6	5	4	3	2	1	0

	U-0	U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1
IPR2	-	-	-	-	BCLIP	LVDIP	TMR3IP	CCP2IP
	bit7	6	5	4	3	2	1	0

GIE PEIE In Compatibility Mode

- When IPEN=0 Compatibility Mode
  - INTCON<7> is GIE
  - INTCON<6> is PEIE
  - Note: definition exactly same as 16C INTCON

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GIE/GIEH	**PEIE**/GIEL	T0IE	INT0E	RBIE	T0IF	INT0F	RBIF
bit7	6	5	4	3	2	1	

GIEH & GIEL In Priority Mode

- When IPEN=1 Priority Interrupt Mode

- INTCON<7> is GIEH
- INTCON<6> is GIEL

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GIE/**GIEH**	PEIE/**GIEL**	T0IE	INT0E	RBIE	T0IF	INT0F	RBIF
bit7	6	5	4	3	2	1	

- High Priority Interrupt Enable GIEH replaces GIE
- Low Priority Interrupt Enable GIEL replaces PEIE

High Priority Interrupts

- High Priority Vector uses shadow registers for automatic context save / restore:

```
#pragma code HighVector=0x8
void HighVector (void)
   { _asm GOTO high_priority_interrupt _endasm}

#pragma code // return to default code section

#pragma interrupt high_priority_interrupt save=[symbol]
void high_priority_interrupt (void){
   // Place your high priority interrupt code here
}
```

Low Priority Interrupts

- Low Priority Vector - compiler saves context and restores it with “interruptlow” pragma

```
#pragma code lowVector=0x18
void LowVector (void)
{
_asm GOTO low_priority_interrupt _endasm
}
#pragma code

#pragma interruptlow low_priority_interrupt save=[symbol]
void low_priority_interrupt (void){
   // Place your low priority interrupt code here
}
```

Interrupt Context Save / Restore

- High priority interrupt uses Hardware shadow registers to save and restore WREG,BSR,STATUS.
- Low priority interrupt uses the software stack to manually save WREG,BSR,STATUS.
- You need to add save=[symbol or section] if your ISR is complicated by:
  - Accessing a calculated index within an array
  - Calls other user functions
  - Performs complex math (*,/,float)
  - Accesses a ROM qualified variable

Guidelines for ISR Save Context

ISR Code Behavior	Symbol or Section added to ISR Save List
Call functions that are also called within main code paths	`section(".tmpdata"), PROD`
Access values in Program Memory such as an array declared with the ROM keyword	`TABLPTR, TABLAT`
Performs Multiplication or accesses a calculated index of an array	`PROD`
Executes Division, 16 bit or greater Multiplication, Floating Point, Scientific functions	`section("MATH_DATA")`

Example: ISR accesses a calculated array index and executes a division within the ISR:

```
#pragma interrupt sample_adc save=PROD, section("MATH_DATA")
```

Large Arrays and Structures

- Linker attempts to fit each variable into a default 256 byte section
- Need to create a larger protected section for arrays and structures larger than 256 bytes:
- Modify <processor name>.lkr file as follows:

```
DATABANK NAME=gpr2       START=0x200 END=0x2FF
DATABANK NAME=big_array1 START=0x300 END=0x4FF  PROTECTED
DATABANK NAME=gpr5       START=0x500 END=0x5FF
SECTION  NAME=big_array  RAM=big_array1
```

Large Arrays and Structures (cont.)

- Add #pragma to use new section in source.c

```
#pragma udata big_array // Select large section
   unsigned char test[456];
#pragma udata  // Return to normal section
```

- Access these large (>256 byte) arrays and structures through pointers or a variable based index (array[index] or *array)
  - Avoid fixed element addressing on these large arrays and structures (ex: array[2])
- Pointers are more code efficient than array indexing

Peripherals

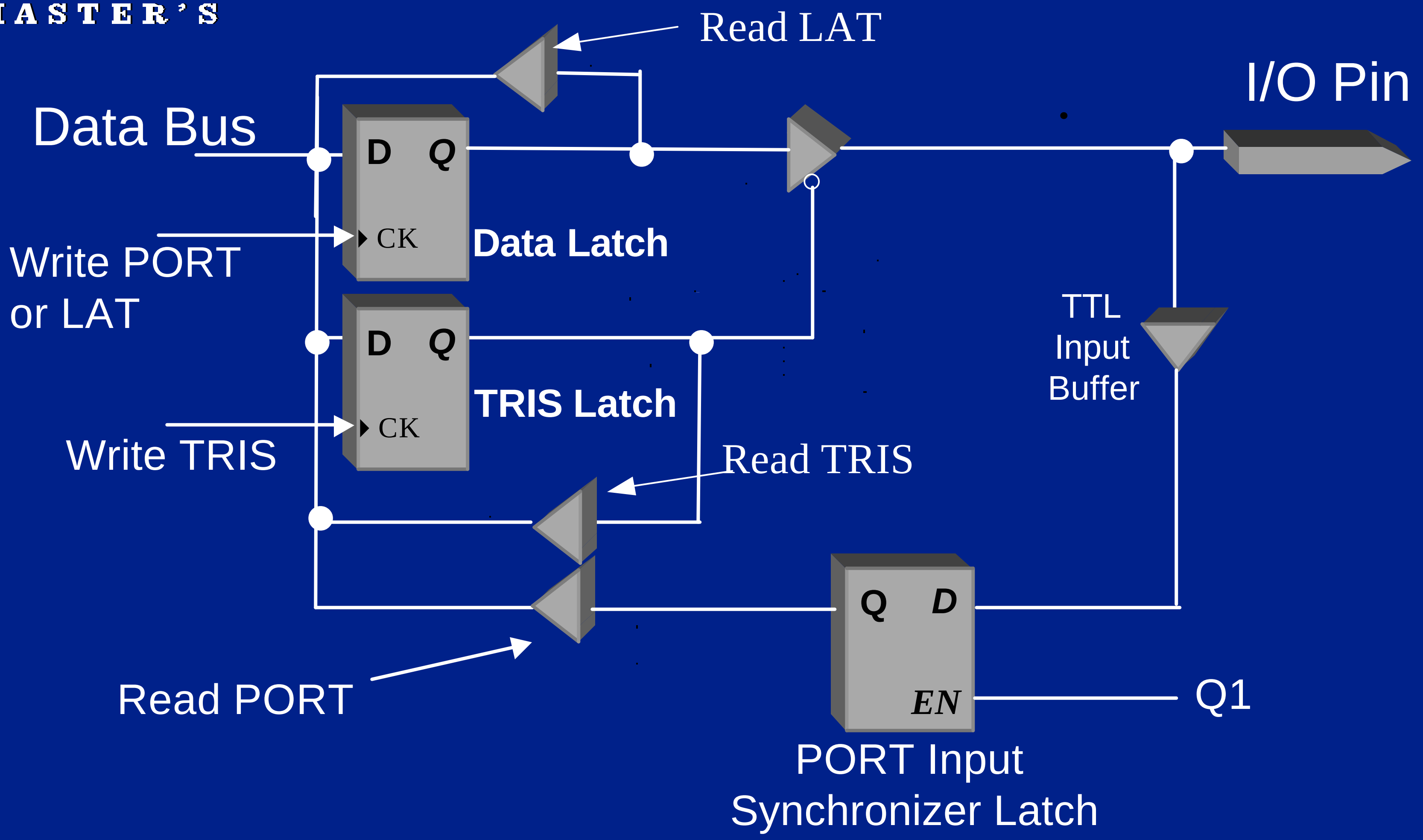
PIC18 Peripherals

- Digital I/O Ports
- Timer0, 1, 2, 3
- Compare/Capture/PWM (CCP)
- Analog-To-Digital Converter
- Analog Comparator
- Addressable USART (AUSART)
- Master Synchronous Serial Port (MSSP)
- External Memory Access (EMA)
- Controller Area Network (CAN)

PIC18 Peripherals
Digital I/O Ports

- Up to 68 bi-directional I/O pins
- High sink/source capability (up to 25mA)
- Direct bit (pin) manipulation (single-cycle)
- Each port pin has:
  - Individual direction control (TRISA~TRISJ)
  - Data Latch (LATA~LATJ - read-modify-writes)
  - Port Register (PORTA~PORTJ reads value on pins)
- All I/O pins have ESD protection

Port Latch Block Diagram



I/O pins have ESD protection diodes

I/O Pin Direction

- Direction of I/O pins controlled by individual TRIS bits
  - 1 = Input (default power on reset state)
  - 0 = Output
- Example

```
TRISAbits.TRISA5 = 0; // Make RA5 output
TRISB = 0b11110000;  // Make RB0:3 outputs,
                       // RB4:7 inputs
```

Reading / Writing I/O Ports

- Reading a I/O port or bit uses the PORT register

```
if (PORTCbits.RC2) // Execute if RC2 = 1
if (PORTC == 0b11110000) // Check for F0
```

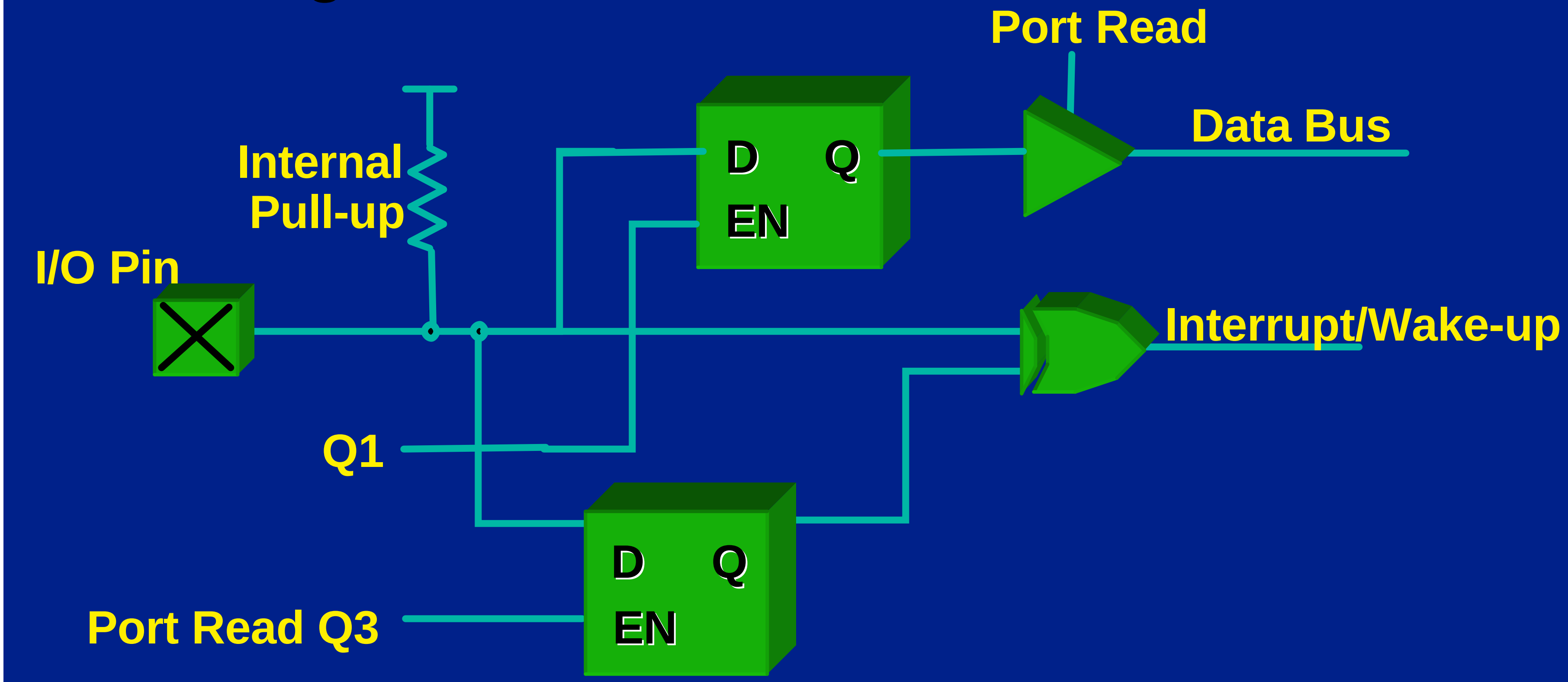
- Writing to an I/O port or bit should use LAT register

```
LATAbits.LATA0 = 1; // Set RA0
LATB = 0xFF;   // Set all of PORTB output
               // pins to a logic one
```

PIC18 Peripherals

PORTB : Interrupt on Change

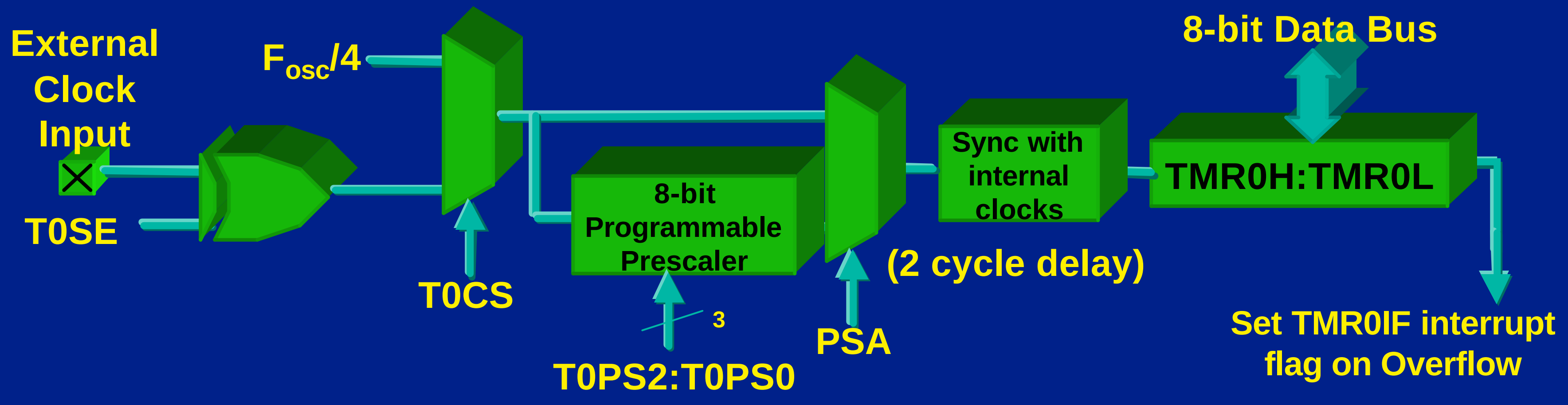
- Internal Pull-Ups and Wakeup/Interrupt On Change feature



PIC18 Peripherals

Timer0

- 8-bit/16-bit Timer/Counter
    - 16-bit Read and Writes
- 8-bit Software Programmable Prescaler
- Internal or External clock select
- Interrupt on overflow from **FFh/FFFFh** to **00h**



Timer 0 Setup

	bit 7							bit 0
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0

TMR0ON	**Timer 0 On/Off Control** 1 = Enables Timer 0 0 = Stops Timer 0
T08BIT	**Timer 0 8-bit / 16-bit Select** 1 = Timer 0 configured for 8-bit mode 1 = Timer 0 configured for 16-bit mode
T0CS	**Timer 0 Clock Source Select** 1 = Transition on T0CKI pin (counter mode) 0 = Internal Instruction cycle (timer mode)
T0SE	**Timer 0 Source Edge Select** 1 = Increment on High -> Low T0CKI transition 0 = Increment on Low -> High T0CKI transition
PSA	**Timer 0 Prescaler Asignment** 1 = Timer 0 Prescaler is NOT assigned, prescaler bypassed 0 = Timer 0 Prescaler assigned and enabled
T0PS2:T0PS0	**Timer 0 Prescaler Selection** 111 = 1:256 011 = 1:16 110 = 1:128 010 = 1:8 101 = 1:64 001 = 1:4 100 = 1:32 000 = 1:2

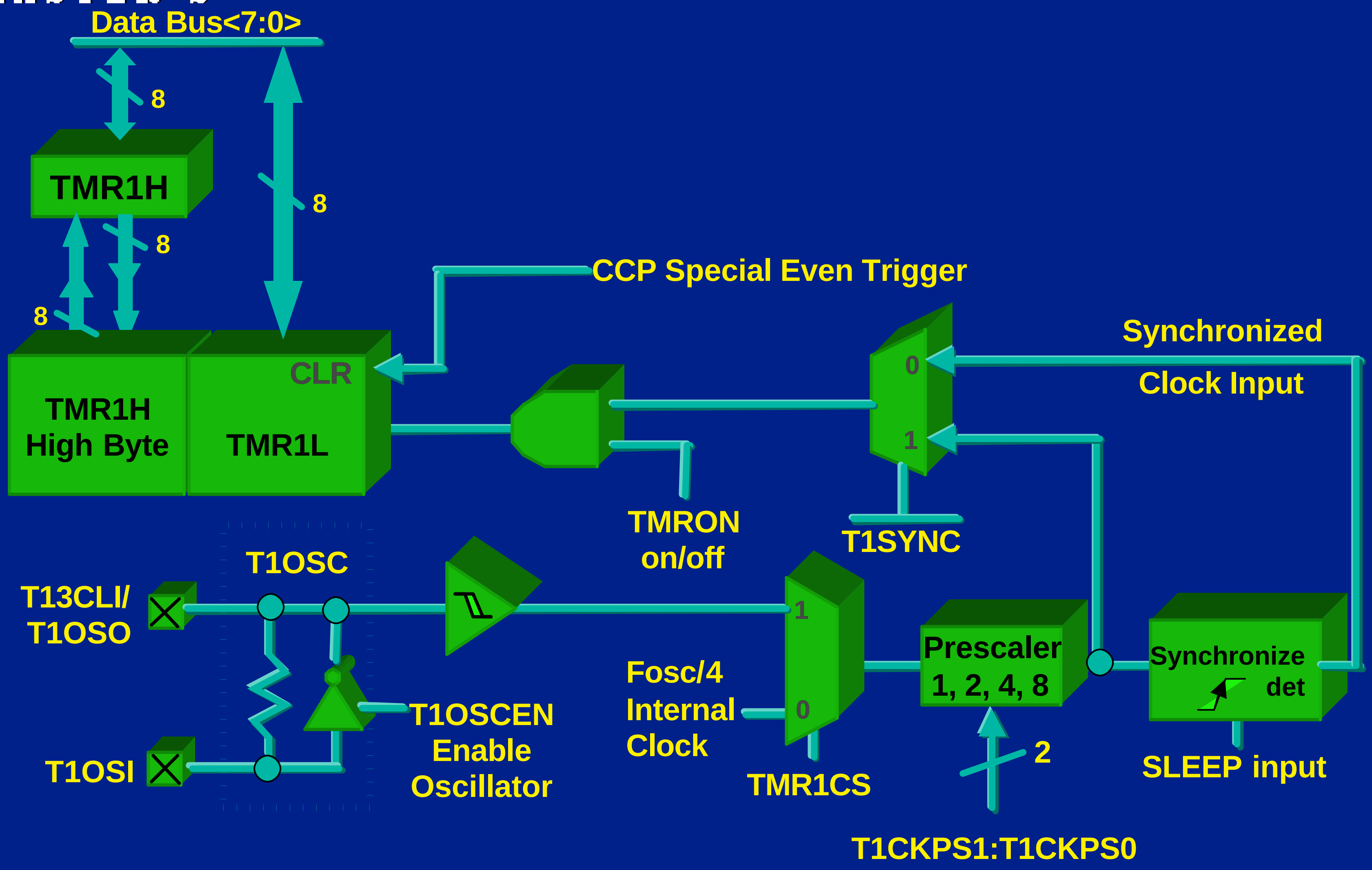
PIC18 Peripherals

Timer1 and Timer3

- 16-bit Timer / Counter
- Consists of two readable and writeable 8-bit registers
  - 16-bit Read / Write mode eliminates hazards
- ÷1, ÷2, ÷4, or ÷8 Prescaler
- Timer, Synchronous or Asynchronous Counter
- Timer1 can also operate from an external crystal with its built in oscillator feature.
- Interrupt on overflow from **FFFFh** to **`0000h`**

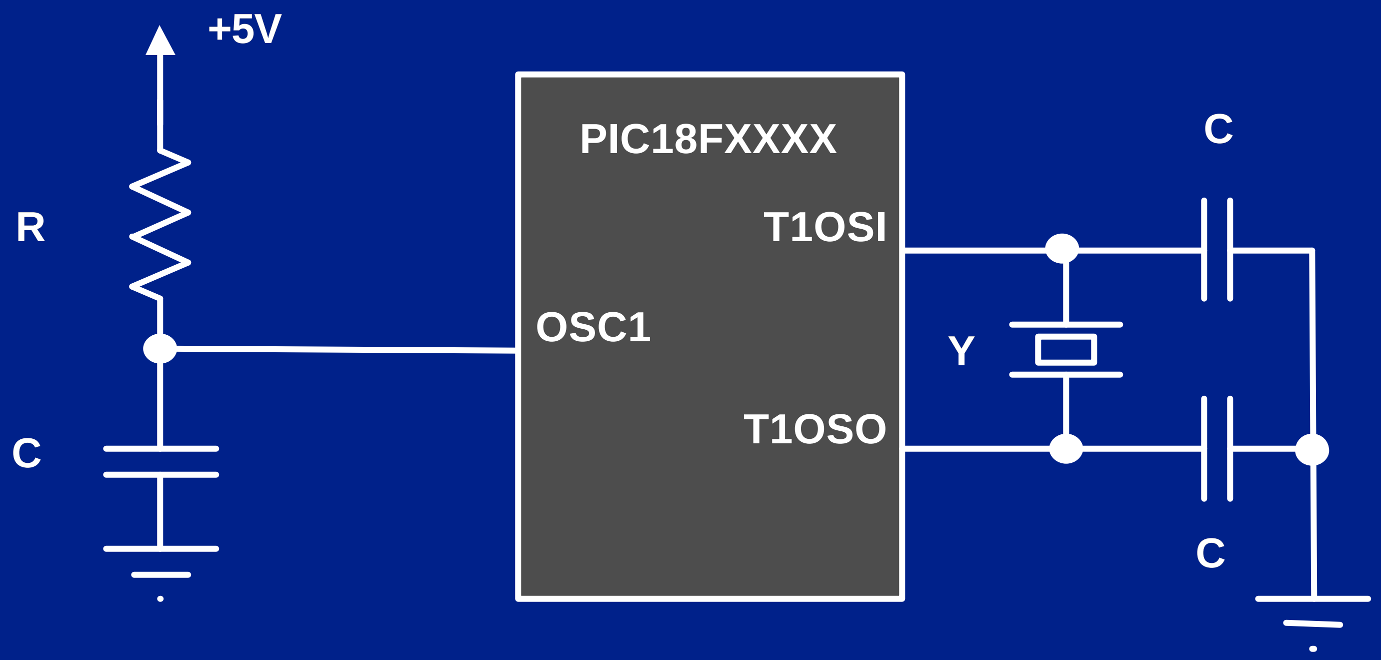
PIC18 Peripherals

Timer1 and Timer3 (Continued)



PIC18FXXX MCU Peripherals

TMR1 as a Real Time Clock



Preload TMR1H register for faster overflows:

TMR1H=80h → 1 second overflow
TMR1H=C0h →0.5 second overflow

See Application Note AN580 for more info.

Timer 1 Setup

T1CON

bit 7							bit 0
RD16	-	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNCH	TMR1CS	TMR1ON

RD16	**16-bit Read/Write Mode Enable** 1 = Enables Read/Write of Timer 1 in one 16-bit operation 0 = Enables Read/Write of Timer 1 in two 8-bit operations
T1CKPS1:T1CKPS0	**Timer 1 Input Clock Prescale Selection** 11 = 1:8 01 = 1:2 10 = 1:4 00 = 1:1
T1OSCEN	**Timer 1 Oscillator Enable** 1 = Timer 1 oscillator is enabled 0 = Timer 1 oscillator is disabled
T1SYNCH	**Timer 1 External Clock Synchronization Selection** 1 = Do NOT synchronize external clock 0 = Synchronize external clock input
TMR1CS	**Timer 1 Clock Source Selection** 1 = External clock from RC0/T1OSC0/T13CKI (counter) 0 = Internal Instruction Cycle
TMR1ON	**Timer 1 On / Off Selection** 1 = Enables Timer 1 0 = Disables Timer 1

Timer 3 Setup

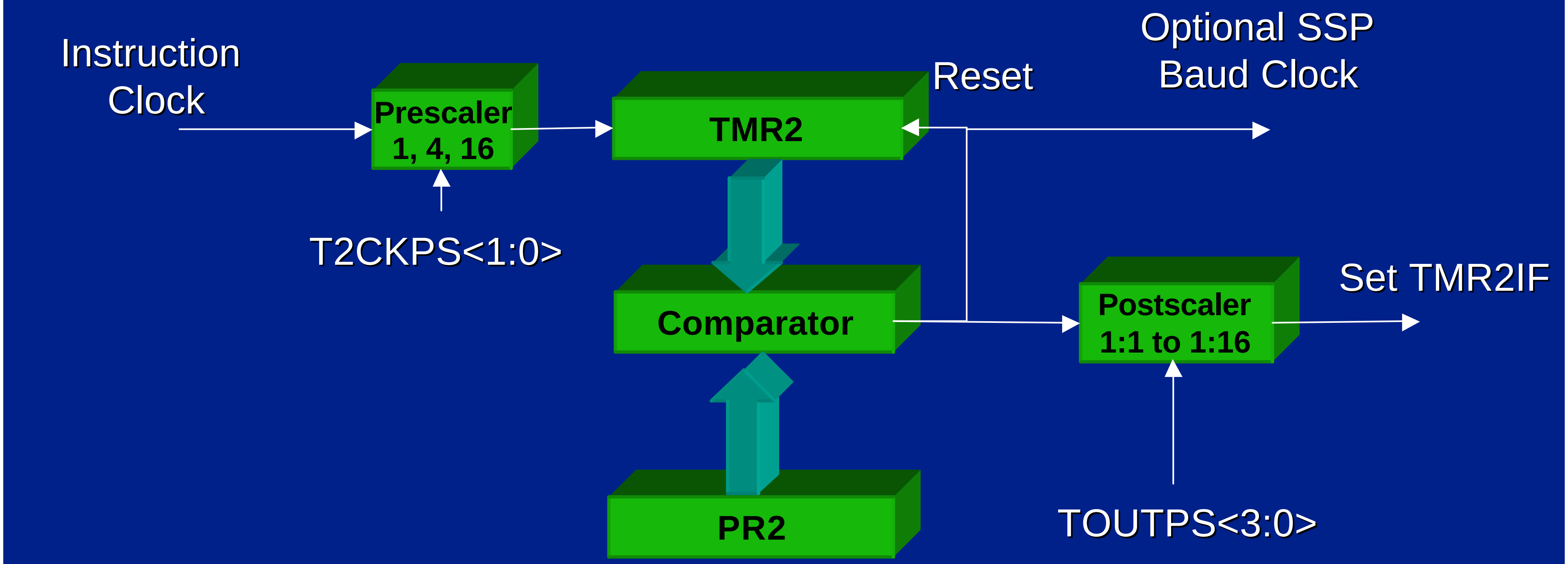
	bit 7							bit 0
T3CON	RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	T3SYNCH	TMR3CS	TMR3ON

RD16	**16-bit Read/Write Mode Enable** 1 = Enables Read/Write of Timer 3 in one 16-bit operation 0 = Enables Read/Write of Timer 3 in two 8-bit operations
T3CCP2:T3CCP1	**Timer 3 and Timer 3 CCP Timebase Selection** 1X = Timer 3 is Capture/Compare clock source for all CCPs 10 = Timer 3 is Capture/Compare clock source for CCP2, Timer 1 is Capture/Compare clock source for CCP1 01 = Timer 1 is Capture/Compare clock source for all CCPs
T3CKPS1:T3CKPS0	**Timer 3 Input Clock Prescale Selection** 11 = 1:8 01 = 1:2 10 = 1:4 00 = 1:1
T3SYNCH	**Timer 3 External Clock Synchronization Selection** 1 = Do NOT synchronize external clock 0 = Synchronize external clock input
TMR3CS	**Timer 3 Clock Source Selection** 1 = External clock from RC0/T1OSC0/T13CKI (counter) 0 = Internal Instruction Cycle
TMR3ON	**Timer 3 On / Off Selection** 1 = Enables Timer 1 0 = Disables Timer 1

PIC18 Peripherals
Timer2 and Timer4

- 8-bit Timers with prescaler and postscaler
- TMR2 used as time base for PWM mode of CCP module
- TMR2/TMR4 are readable & writable
- TMR2/TMR4 increments until they match period PR2/PR4, then resets to 00h
- TMR2/TMR4 match with PR2/PR4 generates an interrupt through postscaler
- TMR2 can serve as baud clock for MSSP

PIC18 Peripherals
TMR2 Timer: Period Register



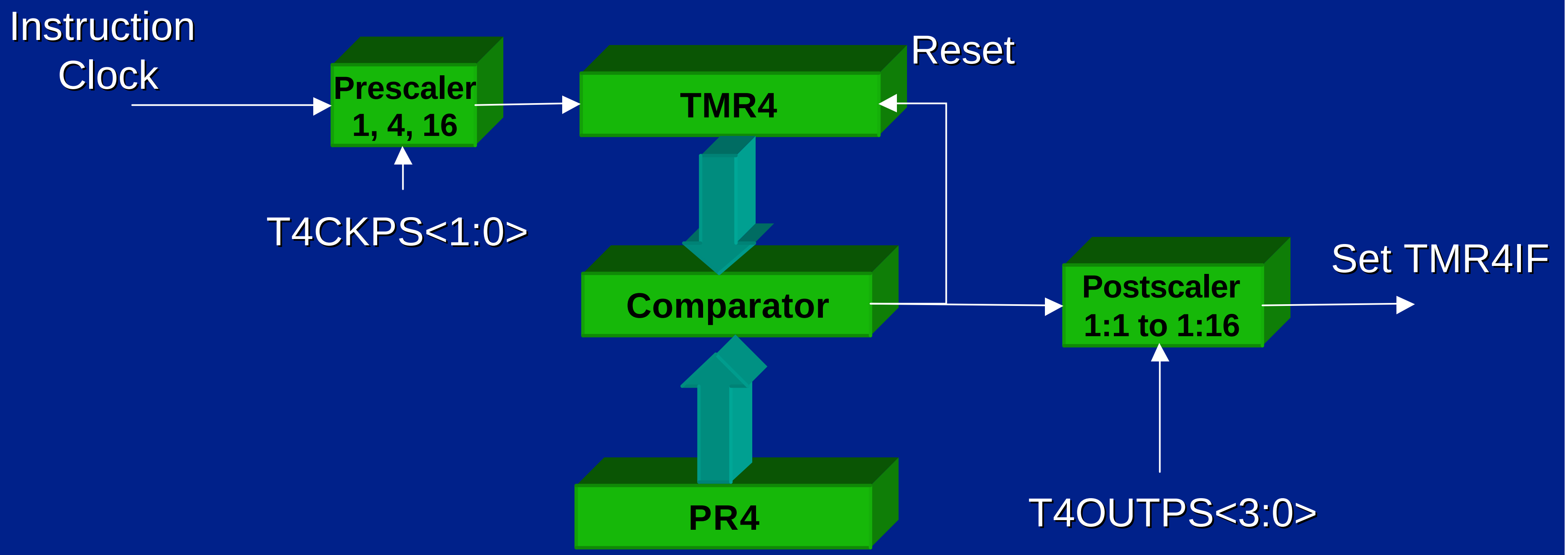
Timer 2 Setup

T2CON Register Format

bit 7							bit 0
-	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0

TOUTPS<3:0>	**Select Timer 2 Postscaler:** 0000 = 1:1 Postscale 0001 = 1:2 Postscale …. 1111 = 1:16 Postscale
TMR2ON	**Timer 2 On / Off Control:** 0 = Timer 2 is Off 1 = Timer 2 is On
T2CKPS1	**Select Timer 2 Prescaller:** 00 = Prescaller is 1 01 = Prescaller is 4 1X = Prescaller is 16

PIC18 Peripherals
TMR4 Timer: Period Register



Timer 4 Setup

T4CON Register Format

bit 7							bit 0
-	T4OUTPS3	T4OUTPS2	T4OUTPS1	T4OUTPS0	TMR4ON	T4CKPS1	T4CKPS0

T4OUTPS<3:0>	**Select Timer 4 Postscaler:** 0000 = 1:1 Postscale 0001 = 1:2 Postscale …. 1111 = 1:16 Postscale
TMR4ON	**Timer 4 On / Off Control:** 0 = Timer 4 is Off 1 = Timer 4 is On
T4CKPS1	**Select Timer 4 Prescaller:** 00 = Prescaller is 1 01 = Prescaller is 4 1X = Prescaller is 16

Timer 2 Interrupts

RCON Register

bit 7							bit 0
IPEN	-	-	~RI	~TO	~PD	~POR	~BOR

IPEN	**Interrupt Priority Level Enable:** 1 = Enable Interrupt Priority Levels 0 = Disable Interrupt Priority Levels

INTCON Register

bit 7							bit 0
GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF

GIE/GIEH	**Global Interrupt Enable** **IPEN=0** 1 = Enable Unmasked Interrupts 0 = Disable all interrupts	 **IPEN=1** 1 = Enables High Priority Interrupts 0 = Disables High Priority Interrupts
PEIE/GIEL	**Peripheral Interrupt Enable** **IPEN = 0** 1 = Enables Unmasked Peripheral Interrupts 0 = Disables Peripheral Interrupts	 **IPEN = 1** 1 = Enables Low Priority Interrupts 0 = Disables Low Priority Interrupts

Timer 2 Interrupts Continued

PIR1 (Peripheral Interrupt Request Flag) Register

bit 7							bit 0
PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF

TMR2IF	Timer 2 to PR2 Match Interrupt Flag 1 = TMR2 to PR2 Match Interrupt Occurred 0 = No TMR2 to PR2 Match Occurred

PIE1 (Peripheral Interrupt Enable) Register

bit 7							bit 0
PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE

TMR2IE	Timer 2 to PR2 Match Interrupt Enable 1 = Enable TMR2 to PR2 Match Interrupts 0 = Disable TMR2 to PR2 Match Interrupts

IPR1 (Peripheral Interrupt Priority) Register

bit 7							bit 0
PSPIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP

TMR2IP	Timer 2 to PR2 Match Interrupt Priority Selection 1 = TMR2 to PR2 Match Assigned to High Priority Interrupt 0 = TMR2 to PR2 Match Assigned to Low Priority Interrupt

TMR2 Initialization Example

- 200 uS / 5 Khz high priority interrupt, 40 Mhz clock / 10 Mhz instruction clock:

```
T2CON = 0b00001101;   // 4:1 pre 2:1 postscale
PR2 = 249;            // 250 count TMR2 period
RCON = 0b10000000;    // Enable Priority
PIE1 = 0b00000010;    // Enable TMR2 interrupt
IPR1 = 0b00000010;    // TMR2 high priority
PIR1bits.TMR2IF = 0;  // Optional to eliminate
TMR2 = 0;             // first interrupt
INTCON = 0b10000000;  // Turn on interrupts
```

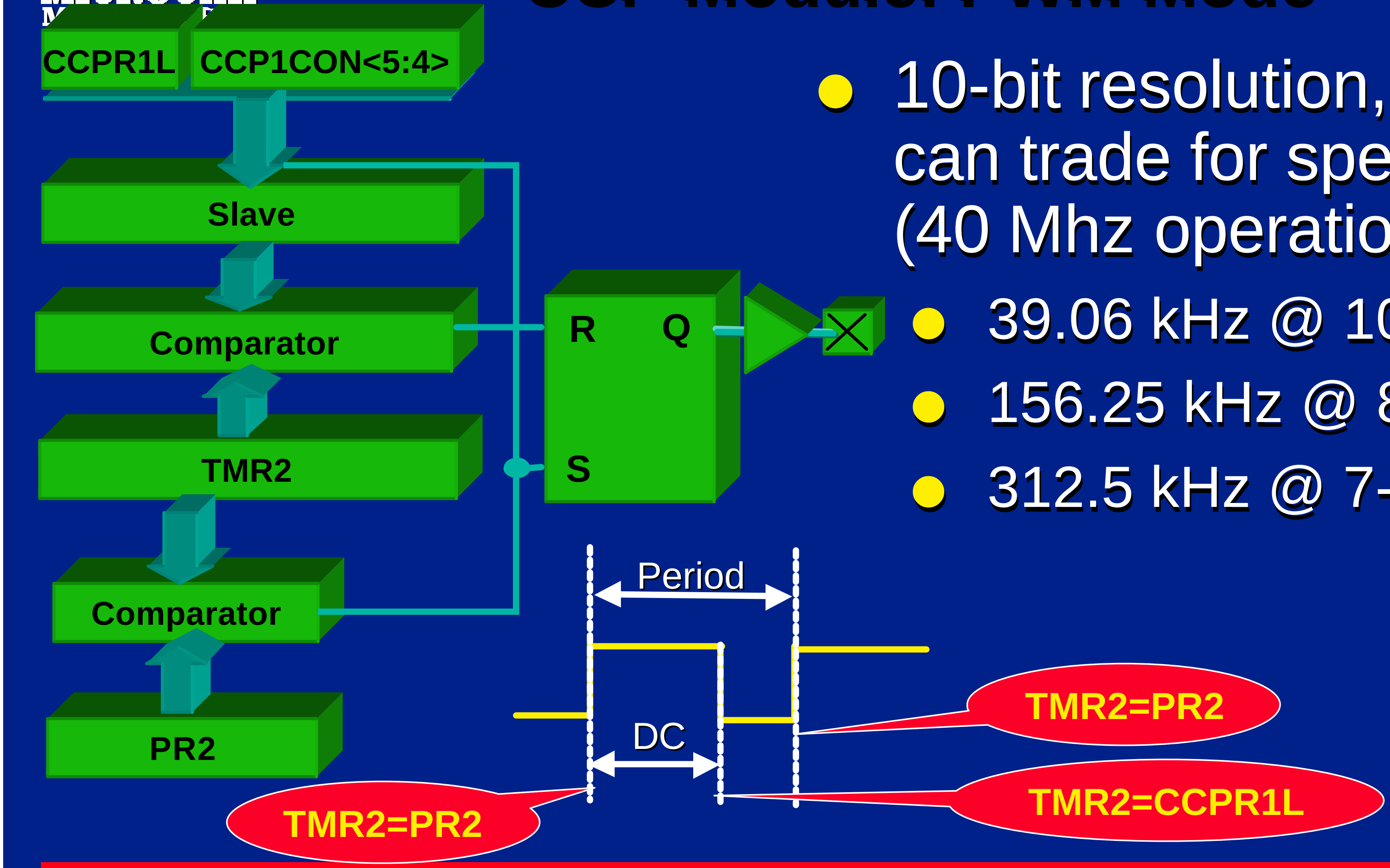
10,000,000 / (4 (prescale) * 2 (postscale) * 250 (period)) = 5,000 Khz or 200 uS period

Timer 2 ISR Example

- Test and Clear PIR1bits.TMR2IF:

```
void high_priority_interrupt(void){
   if (PIR1bits.TMR2IF){
      PIR1bits.TMR2IF = 0;
      // execute Timer 2 service code here
      }
   else if (<other high priority peripherals>){
      // Clear other peripheral bits
      // execute peripheral service code here
      }
   else Reset(); // Hit interrupt without valid
}         // flag - illegal condition so restart
```

PIC18 Peripherals
CCP Module: PWM Mode



- 10-bit resolution, can trade for speed (40 Mhz operation)
  - 39.06 kHz @ 10-bit
  - 156.25 kHz @ 8-bit
  - 312.5 kHz @ 7-bit

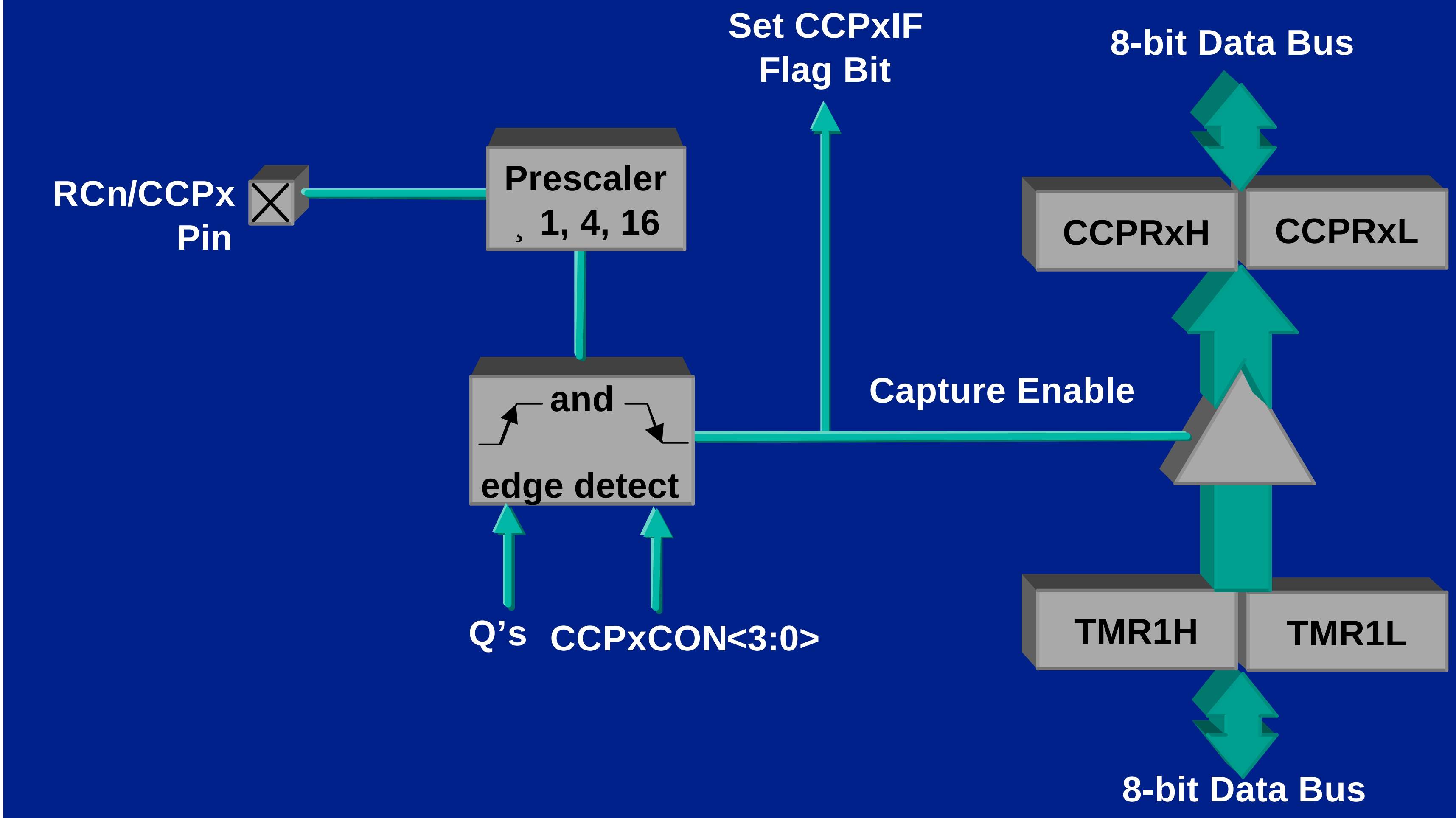
PIC18 Peripherals

CCP Module: Input Capture Mode

- Captures 16-bit TMR1 value when an event occurs on CCPx pin:
  - Every falling edge
  - Every rising edge
  - Every 4th rising edge
  - Every 16th rising edge
- Capture generates an interrupt

PIC18 Peripherals

CCP Module: Input Capture Mode *(continued)*



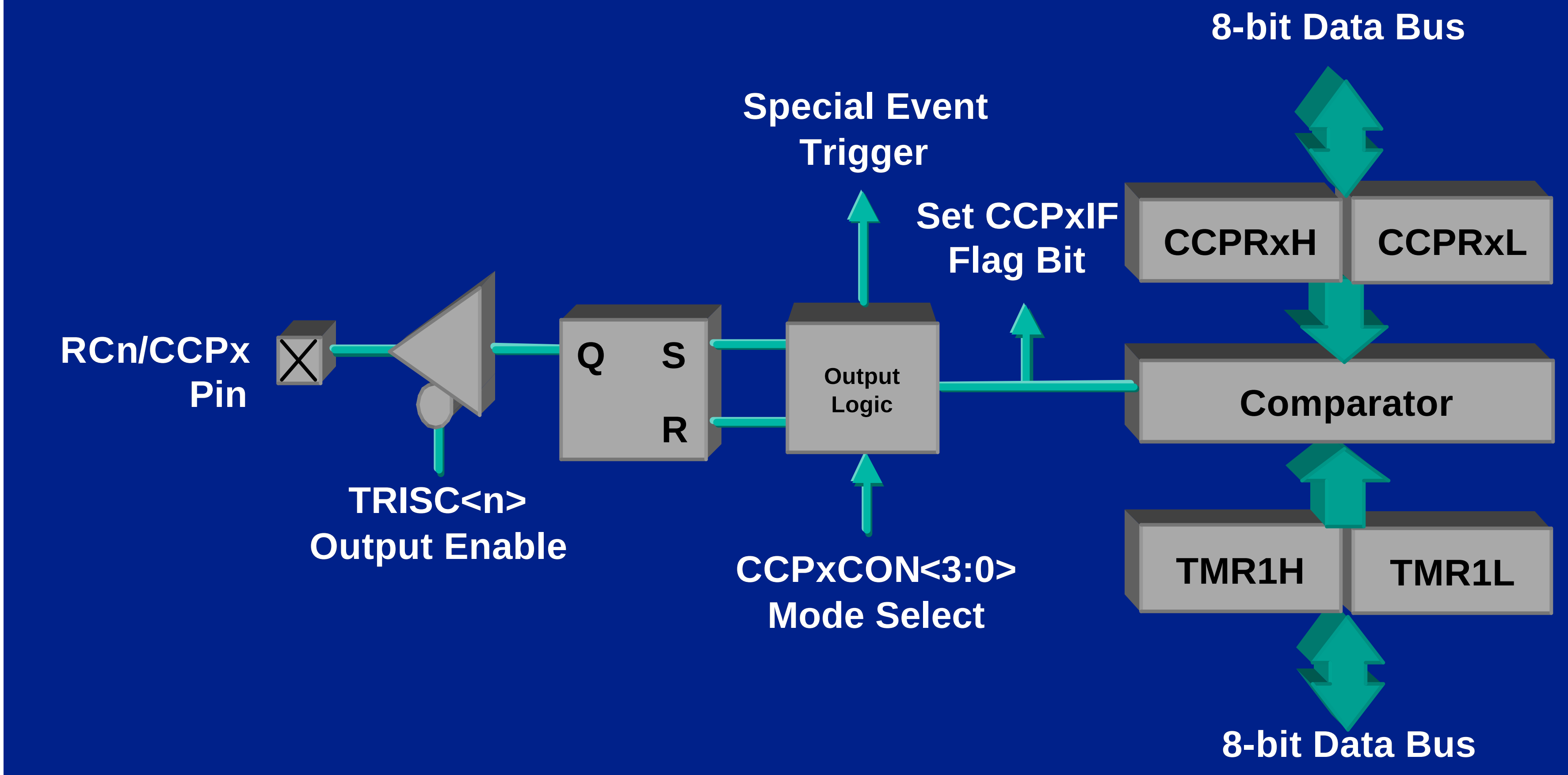
PIC18 Peripherals

CCP Module: Output Compare Mode

- 16-bit CCPRx register value is compared to TMR1, and on match the CCPx pin is
  - Driven High/Low
  - Toggled
  - Unchanged
- Compare match generates interrupt
- Special event trigger clears TMR1 and can start A/D conversion

PIC18 Peripherals

CCP Module: Output Compare Mode *(continued)*



CCP1 Setup

CCP1CON

bit 7							bit 0
-	-	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0

DC1B1:DC1B0	**(2) LSBs of PWM Duty Cycle** **PWM Mode** -> (2) LSBs of a 10-bit Duty Cycle. The upper (8) bits (DC19:DC12) of the duty cycle are found in CCPR1L **Capture/Compare Modes** -> Unused
CCP1M3:CCP1M0	**CCP1 Mode Selection** 0000 = Capture/Compare/PWM 1 Disable (resets CCP1 module) 0001 = Reserved 0010 = Compare Mode, Toggle CCP1 output on match 0011 = Reserved 0100 = Capture Mode, every falling edge 0101 = Capture Mode, every rising edge 0110 = Capture Mode, Every 4th rising edge 0111 = Capture Mode, Every 16th rising edge 1000 = Compare Mode, force CCP1 output High on match 1001 = Compare Mode, force CCP1 output Low on match 1010 = Compare Mode, CCP1 output unchanged 1011 = Compare Mode, Trigger Special Event 11XX = PWM Mode

Note: Pin defaults to '0' when capture mode is engaged

CCP2 Setup

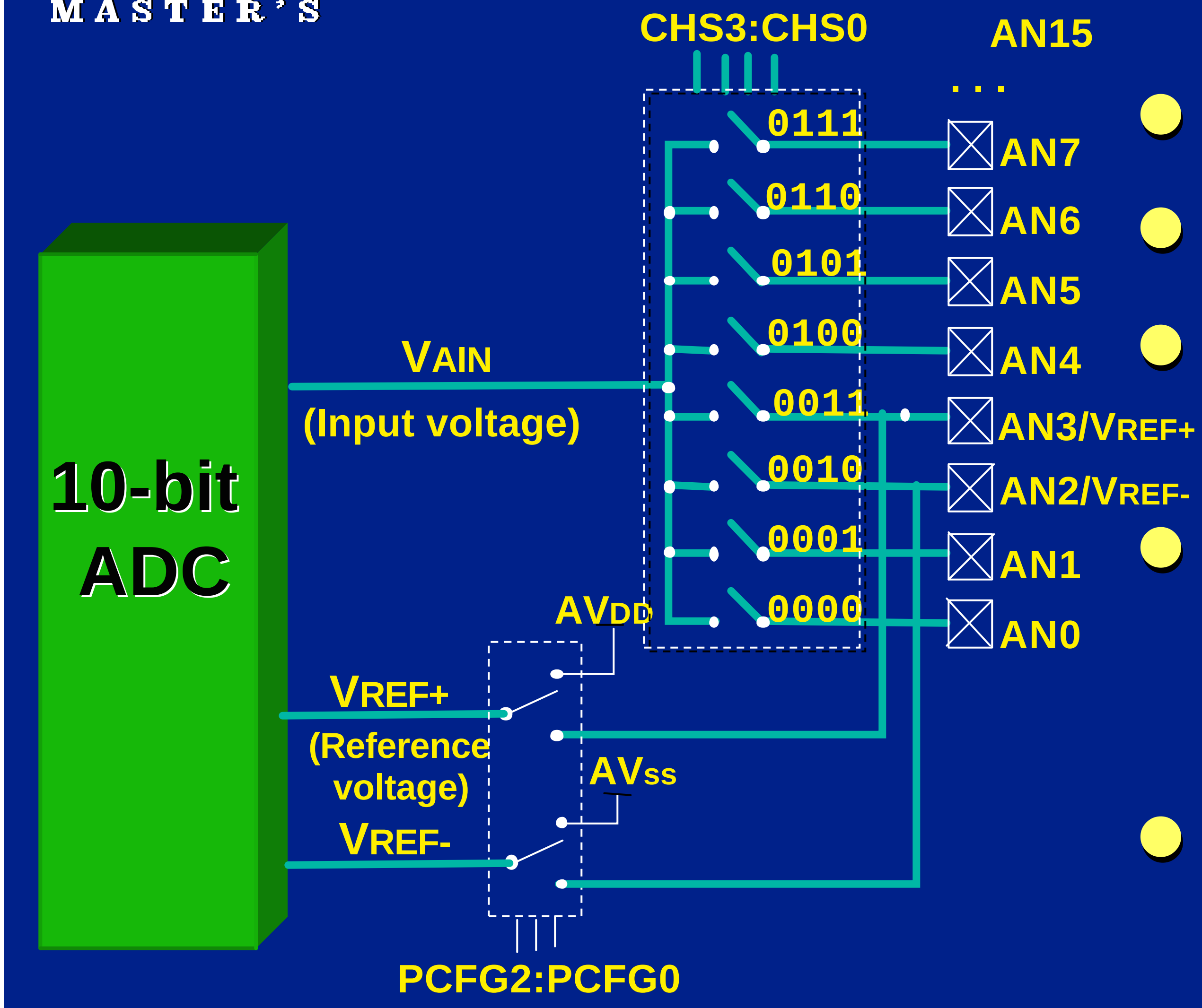
	bit 7							bit 0
CCP2CON	-	-	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0

DC2B1:DC2B0	**(2) LSBs of PWM Duty Cycle** **PWM Mode** -> (2) LSBs of a 10-bit Duty Cycle. The upper (8) bits (DC29:DC22) of the duty cycle are found in CCPR2L **Capture/Compare Modes** -> Unused
CCP2M3:CCP2M0	**CCP2 Mode Selection** 0000 = Capture/Compare/PWM 1 Disable (resets CCP2 module) 0001 = Reserved 0010 = Compare Mode, Toggle CCP2 output on match 0011 = Reserved 0100 = Capture Mode, every falling edge 0101 = Capture Mode, every rising edge 0110 = Capture Mode, Every 4th rising edge 0111 = Capture Mode, Every 16th rising edge 1000 = Compare Mode, force CCP2 output High on match 1001 = Compare Mode, force CCP2 output Low on match 1010 = Compare Mode, CCP2 output unchanged 1011 = Compare Mode, Trigger Special Event 11XX = PWM Mode

Note: Pin defaults to ‘0’ when capture mode is engaged

PIC18 Peripherals

10-bit ADC - Block Diagram



- Up to 16 ch.
- 10-bit ± 1 LSb
- Conversion during SLEEP
- Internal Or External Reference
- Up to 25ksps
  - 34 ksps without channel change

A/D Setup ADCON0

ADCON0

bit 7							bit 0
ADCS1	ADCS0	CSH2	CHS1	CHS0	GO_DONE	-	ADON

ADCS1:ADCS0 **Also** **ADCON1 ADCS2**	**A/D Conversion Clock Select (ADCON1 contains ADCS2)** ADCON1.ADCS2 = 0 00 = FOSC/2 01 = FOSC/8 10 = FOSC/32 11 = Frc Internal RC Oscillator	 ADCON1.ADCS2 = 1 00 = FOSC/4 00 = FOSC/16 00 = FOSC/64 11 = Frc Internal RC Oscillator
CH2:CH0	**Analog Channel Select Bits** 000 = Channel 0, AN0 001 = Channel 1, AN1 010 = Channel 2, AN2 011 = Channel 3, AN3 100 = Channel 4, AN4 101 = Channel 5, AN5 110 = Channel 6, AN6 111 = Channel 7, AN7	
GO_DONE	**A/D Conversion Status and Conversion Start** 1 = Conversion in progress, set this bit to start a conversion 0 = Conversion complete, result in ADRES, cleared by A/D converter	
ADON	**A/D Converter On / Off Selection** 1 = Enables A/D Converter 0 = Disables A/D Converter	

A/D Setup ADCON1

ADCON1

bit 7							bit 0
ADFM	ADCS2	-	-	PCFG3	PCFG2	PCFG1	PCFG0

ADFM	**A/D Result Format Selection** 1 = Right Justified. (6) MSBs of ADRESH are ‘0’ 0 = Left Justified, (6) LSBs of ADRESL are ‘0’
ADCS2	**See ADCON0 for Conversion Clock Selection**
PCFG3:PCFG0	**Analog Port Configuration Control**

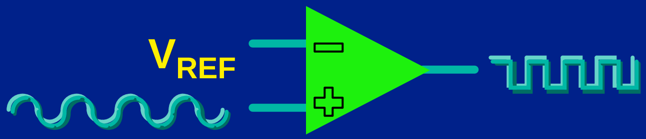
<3:0>	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	VREF+	VREF-	C / R
0000	A	A	A	A	A	A	A	A	VDD	VSS	8 / 0
0001	A	A	A	A	VREF+	A	A	A	AN3	VSS	7 / 1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5 / 0
0011	D	D	D	A	VREF+	A	A	A	AN3	VSS	4 / 1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3 / 0
0101	D	D	D	D	VREF+	D	A	A	AN3	VSS	2 / 1
011x	D	D	D	D	D	D	D	D	—	—	0 / 0
1000	A	A	A	A	VREF+	VREF-	A	A	AN3	AN2	6 / 2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6 / 0
1010	D	D	A	A	VREF+	A	A	A	AN3	VSS	5 / 1
1011	D	D	A	A	VREF+	VREF-	A	A	AN3	AN2	4 / 2
1100	D	D	D	A	VREF+	VREF-	A	A	AN3	AN2	3 / 2
1101	D	D	D	D	VREF+	VREF-	A	A	AN3	AN2	2 / 2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1 / 0
1111	D	D	D	D	VREF+	VREF-	D	A	AN3	AN2	1 / 2

Configuring Inputs as Digital or Analog

- Pins defined as digital enable the digital input buffer
  - Avoid voltages that reside below V_{IH} and above V_{IL} to prevent excessive current
  - PORT pin reads reflect the pin state
- Pins defined as analog disable the digital input buffer
  - Any voltage below Vdd and above Vss is fine
  - PORT pin reads will always be ‘0’
- All pins (D or A) can be digital outputs

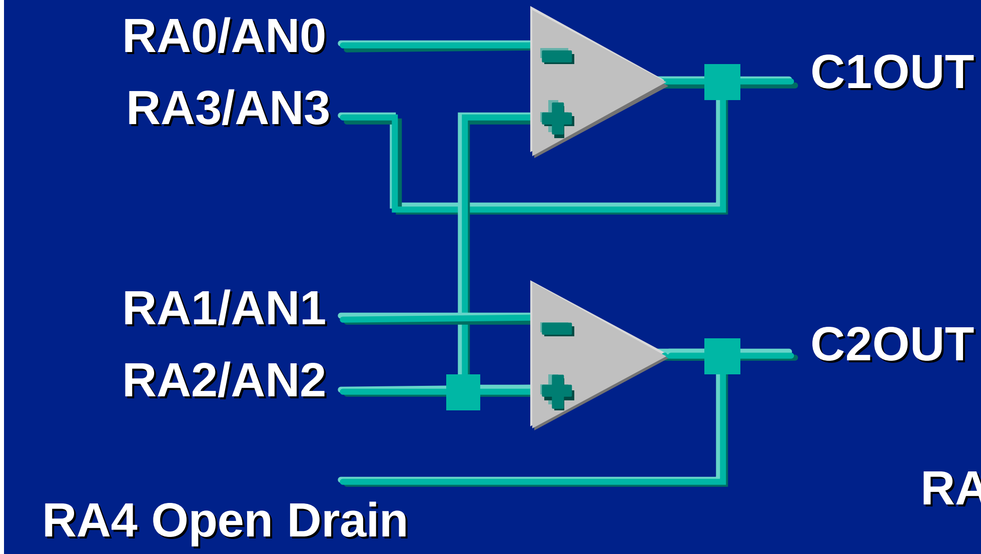
PIC18 Peripherals

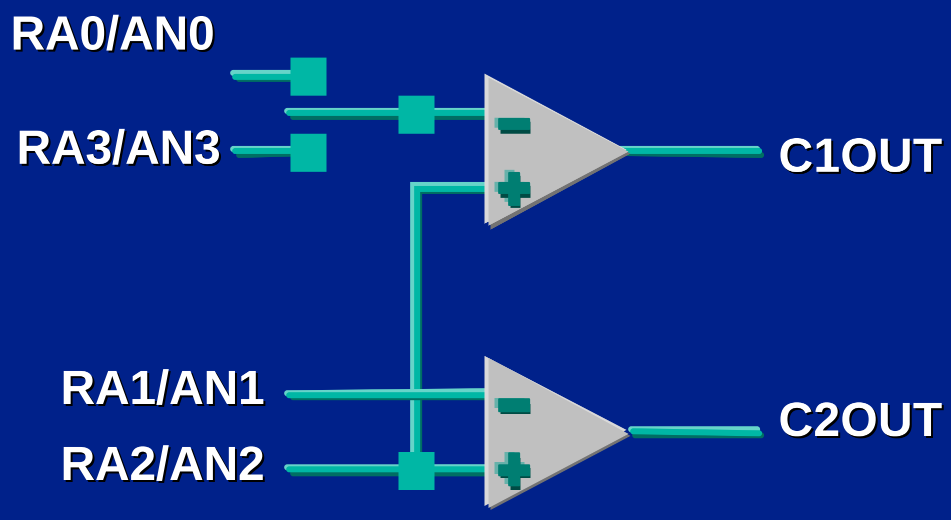
Analog Comparator Module



- Two Analog Comparators
- Programmable on-chip voltage reference
- Eight Programmable modes of operation
- Operates in SLEEP mode
- Generates interrupt / wake-up on output change
- Comparator output pin available

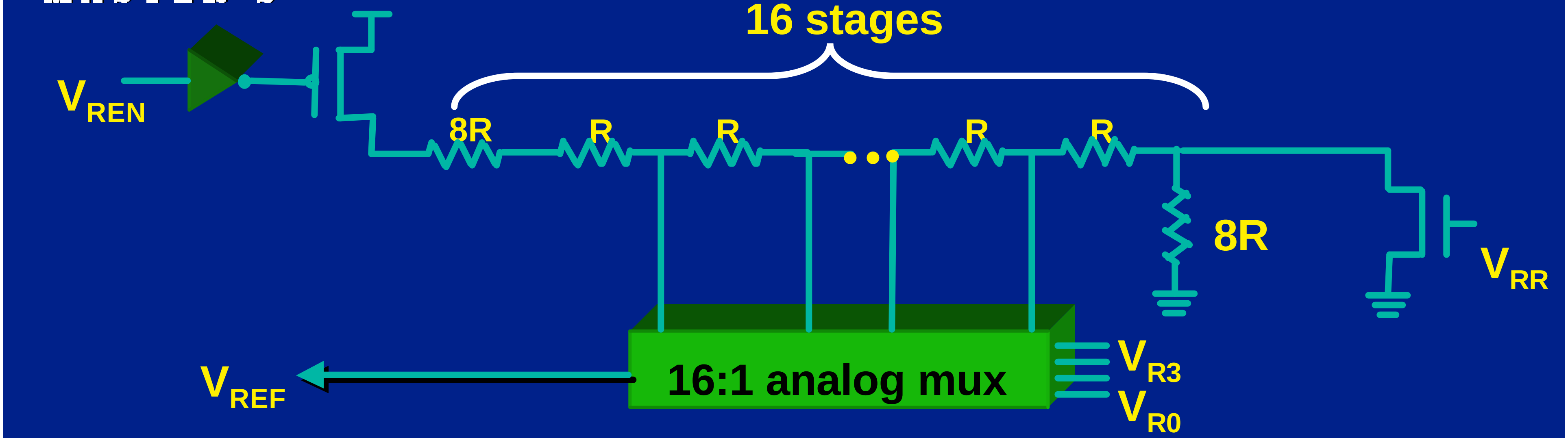
PICmicro MCU Peripherals
Analog Comparator Module *(continued)*





PIC18 Peripherals

Internal VREF: Block Diagram



- 24 or 32 step sizes
- Internal or External Voltage Reference
- Can be used as a D/A converter
- VREF can be directed to an output pin

Note: Check your device datasheet for availability

Comparator Setup

CMCON

bit 7							bit 0
C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0

C2OUT	**Comparator 2 Output Selection** **C2INV = 0:** **C2INV = 1:** 1 = C2 Vin+ > C2 Vin- 1 = C2 Vin+ < C2 Vin- 0 = C2 Vin+ < C2 Vin- 0 = C2 Vin+ > C2 Vin-
C1OUT	**Comparator 1 Output Selection** **C1INV = 0:** **C1INV = 1:** 1 = C1 Vin+ > C1 Vin- 1 = C1 Vin+ < C1 Vin- 0 = C1 Vin+ < C1 Vin- 0 = C1 Vin+ > C1 Vin-
C2INV	**Comparator 2 Output Inversion** 1 = C2 Output inverted 0 = C2 Output not inverted
C1INV	**Comparator 1 Output Inversion** 1 = C1 Output inverted 0 = C1 Output not inverted
CIS	**Comparator 1 Input Switch (when CM<2:0> = 110)** 1 = C1 Vin- connects to RF5/AN10, C2 Vin- connects to RF3/AN8 1 = C1 Vin- connects to RF6/AN11, C2 Vin- connects to RF4/AN9
CM<2:0>	**Comparator Mode Selection** See Comparator Mode Figure

Comparator Reference Setup

CVRCON

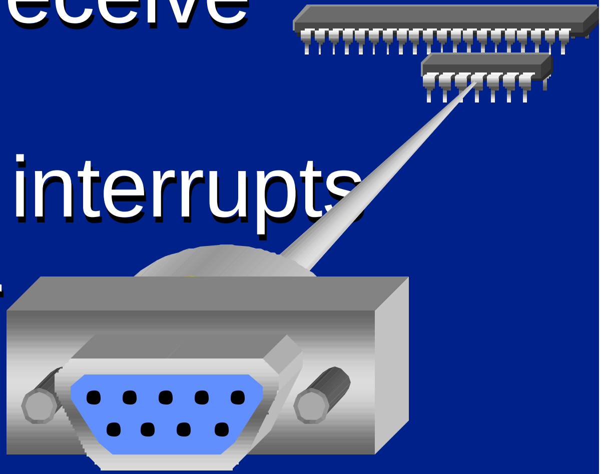
bit 7							bit 0
CVREN	CVROE	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0

CVREN	**Comparator Voltage Reference Enable** 1 = Enables CVREF Circuit, reference ON 0 = Disables CVREF Circuit, reference OFF
CVROE	**Comparator Output Enable** 1 = CVREF Voltage also driven onto RF5/CVREF pin 0 = CVREF disconnected from RF5/CVREF pin Note: TRISF<5> must be set to a ‘1’ (input)
CVRR	**Comparator VREF Source Selection** 1 = 0.00 CVRSRC to 0.75 CVRSRC with CVRSRC/24 step 1 = 0.25 CVRSRC to 0.75 CVRSRC with CVRSRC/32 step
CVR3:CVR0	**Comparator VREF Value Selection** When CVRR = 1 CVREF = (CVR<3:0>/24) * CVRSRC When CVRR = 0 CVREF = (0.25 + (CVR<3:0>/32))* CVRSRC

PIC18 Peripherals

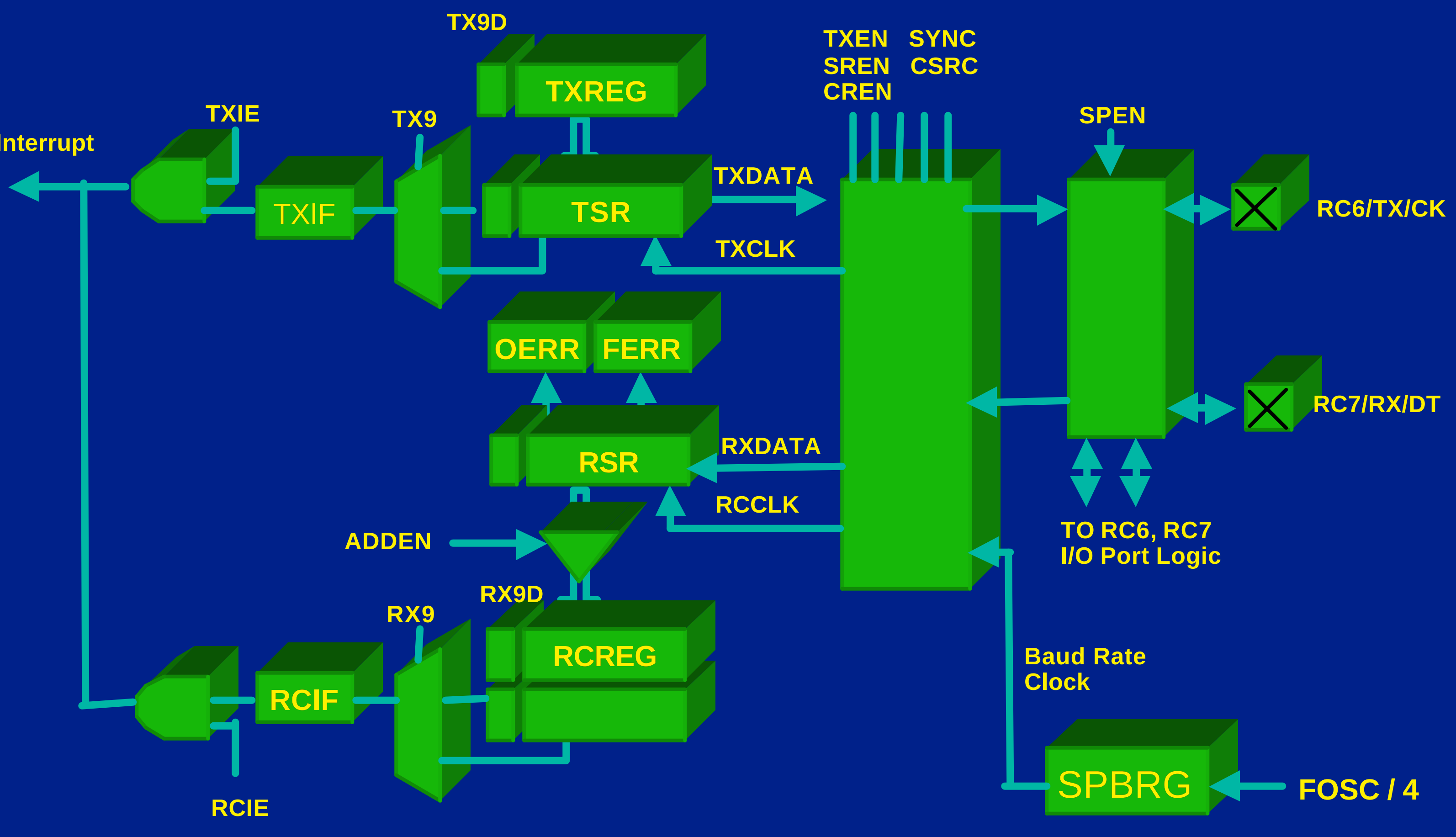
Addressable USART (AUSART)

- Full-duplex Asynchronous Or Half-duplex Synchronous
- 9-bit Addressable mode
- Double-buffered transmit and receive buffers
- Separate transmit and receive interrupts
- Dedicated baud rate generator
- Max bit rates @ 40MHz
  - Asynchronous: 625 kbps / 2.5 Mbps
  - Synchronous: 10 Mbps



PIC18 Peripherals

USART Block Diagram



UART Tx Setup

	bit 7							bit 0
TXSTA	CVREN	TX9	TXEN	SYNC	-	BRGH	TRMT	TX9D

CSRC	**Clock Source Selection (synch mode only)** 1 = Master mode, clock generated by internal BRG 0 = Slave mode, clock derived from external
TX9	**9-bit / 8-bit Mode Transmission Selection** 1 = 9-bit Transmission Format 0 = 8-bit Transmission Format
TXEN	**Transmit Enable (overridden by SREN/CREN in SYNC mode)** 1 = Transmitter Enabled 0 = Transmitter Disabled
SYNC	**Synchronous / Asynchronous Selection** 1 = Synchronous Mode 0 = Asynchronous Mode
BRGH	**High / Low Baud Rate Selection** 1 = High Speed Baud Rate, FOSC / 16 0 = Low Speed Baud Rate, FOSC / 64
TRMT	**Transmit Shift Register Status** 1 = Transmit Shift Register Empty 0 = Transmit Shift Register Full
TX9D	**9th Bit of Transmit Data (valid only in 9-bit mode)** Written before TXREG, used for parity or address/data

UART Rx Setup

	bit 7							bit 0
RCSTA	SPEN	RXD	SREN	CREN	ADDEN	FERR	OERR	RX9D

SPEN	**Serial Port Enable** 1 = Serial Port Enabled, Uses RX and TX as serial port pins 0 = Serial Pore Disabled, RX and TX general purpose I/Os
RX9	**9-bit / 8-bit Mode Reception Selection** 1 = 9-bit Reception Format 0 = 8-bit Reception Format
SREN	**Single Receive Enable (Synchronous Mode Only)** 1 = Enable a Single Receive 0 = Disable Single Receive, cleared when reception completed
CREN	**Continuous Receive Enable** 1 = Enables Receiver; Continuous Reception in Synch mode, overriding SREN 0 = Disables Receiver in Asynchronous Mode, SREN controls Synch mode
ADDEN	**Address Detect Enable** 1 = Enables 9-bit Address Detection, Interrupt and load RCREG when bit 9 is ‘1’ 0 = Disables Address Detection, all bytes received
FERR	**Framing Error** 1 = Framing Error Occurred in this byte, clear by read RCREG + receive next byte 0 = No Framing Error
OERR	**Overrun Error** 1 = Overrun Error, cleared by clearing CREN 0 = No Overrun Error
RX9D	**9th Bit of Received Data (valid only in 9-bit mode)** Read before TXREG, used for parity or address/data

UART Baud Rate Generator

- Separate Resource does not use any timers
- Divides (FOSC / 16 or 64) by 1 to 256

$$\text{Baud Rate} = \frac{\text{FOSC}}{64 * (\text{SPBRG} + 1)}$$

Low Speed Mode
TXSTAbits.BRGH = 0

$$\text{Baud Rate} = \frac{\text{FOSC}}{16 * (\text{SPBRG} + 1)}$$

High Speed Mode
TXSTAbits.BRGH = 1

UART Buffers

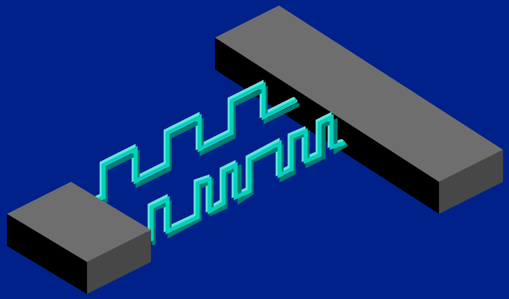
- Load TXREG with byte to be transmitted
  - Buffer empty ONLY when PIR1bits.TXIF is set
- Read received byte from RCREG
  - Received data ONLY when PIR1bits.RCIF is set

```
void putchar(value){
   while (PIR1bits.TXIF == 0);// Wait for empty FIFO
   TXREG = value;
   }
```

PIC18 Peripherals

Master Synchronous Serial Port

- Operates in either SPI™ or I²C™ mode
- SPI Mode
  - Programmable baud rate
  - Maximum baud rates (@ 40MHz)
    - Master: 10 Mbps
    - Slave: 2.5 Mbps Single Byte Tx
  - All four SPI modes supported (0,0;0,1;1,0;1,1)



- I²C Mode
  - Supports standard (100kHz), fast (400kHz), and Microchip's 1MHz I²C standards
  - Hardware Master/Slave implementation

SPI is a trademark of Motorola Semiconductor
I²C is a trademark of Philips Semiconductors

MSSP SPI Mode Setup

	bit 7							bit 0
SSPSTAT	SMP	CKE	D_A	P	S	R_W	UA	BF

SMP	**Input Sample Control** 1 = Input sampled at the end of data output time 0 = Input sampled at the middle of data output time
CKE	**Clock Edge Selection** If CKP = 0 If CKP = 0 1 = Data transmitted on SCLK rising edge 1 = Data transmitted on SCLK falling edge 0 = Data transmitted on SCLK falling edge 0 = Data transmitted on SCLK rising edge
D_A	**Data / Address bit used ONLY in I2C mode, unused in SPI mode**
P	**Stop bit used ONLY in I2C mode, unused in SPI mode**
S	**Start bit used ONLY in I2C mode, unused in SPI mode**
R_W	**Read / Write bit used ONLY in I2C mode, unused in SPI**
UA	**Update Address bit used ONLY in I2C mode, unused in SPI mode**
BF	**Buffer Full (Receive mode only)** 1 = Receive complete, SSBUF is full 0 = Receive not complete, SSBUF is empty

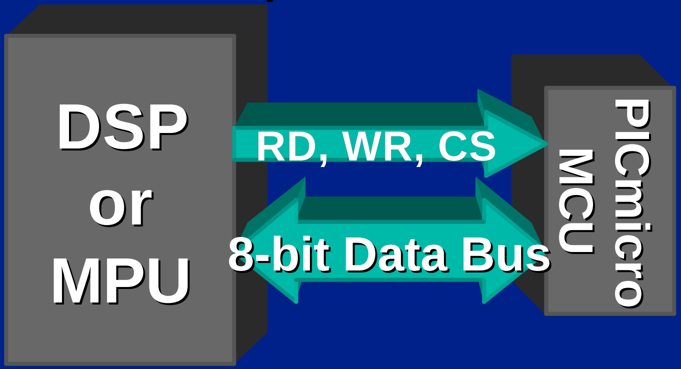
MSSP SPI Mode Setup Cont.

	bit 7							bit 0
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0

WCOL	**Write Collision Detection (Master Mode Only – Must be cleared in software)** 1 = The SSPBUF register was written while still transmitting a previous word 0 = No write collision
SSPOV	**Receive Overflow Indicator (Slave Mode Only – Must be cleared in software)** 1 = A new byte has been received from the master before the previous byte was read from SSPBUF. In case of overflow, the data is lost and SSPBUF must be read to clear overflow condition. Slave transmitter applications should also read SSBUF after each byte 0 = No Slave Receive Overflow
SSPEN	**Synchronous Serial Port Enable** 1 = Enables serial port and configures SCK, SDO, SDI and SS as serial port pins 0 = Disables serial port; allows SCK, SDO SDI and SS to be used as general purpose I/Os
SCP	**Clock Polarity Selection** 1 = Idle state for clock is a high level 0 = Idle state for clock is a low level
SSPM3:SSPM0	**Synchronous Serial Port Mode Selection** 0101 = SPI Slave Mode, Clock – SCLK, SS Control Disabled, SS is GPIO 0100 = SPI Slave Mode, Clock = SCLK, SS Control enabled 0011 = SPI Master Mode, Clock = Timer 2 Output / 2 0010 = SPI Master Mode, Clock = FOSC/64 0001 = SPI Master Mode, Clock = FOSC/16 0000 = SPI Master Mode, Clock = FOSC/4 **NOTE: Other combinations used in I2C mode or reserved**

PICmicro MCU Peripherals
Parallel Slave Port

- Provides an 8-bit interface such that the PICmicro MCU may be used as a peripheral to a microprocessor
- Three I/O on PORTE act as Chip Select, Read, and Write lines
- PORTD is the data bus
- Separate read and write interrupts available
- Currently available on most 40-pin, 14-bit core devices

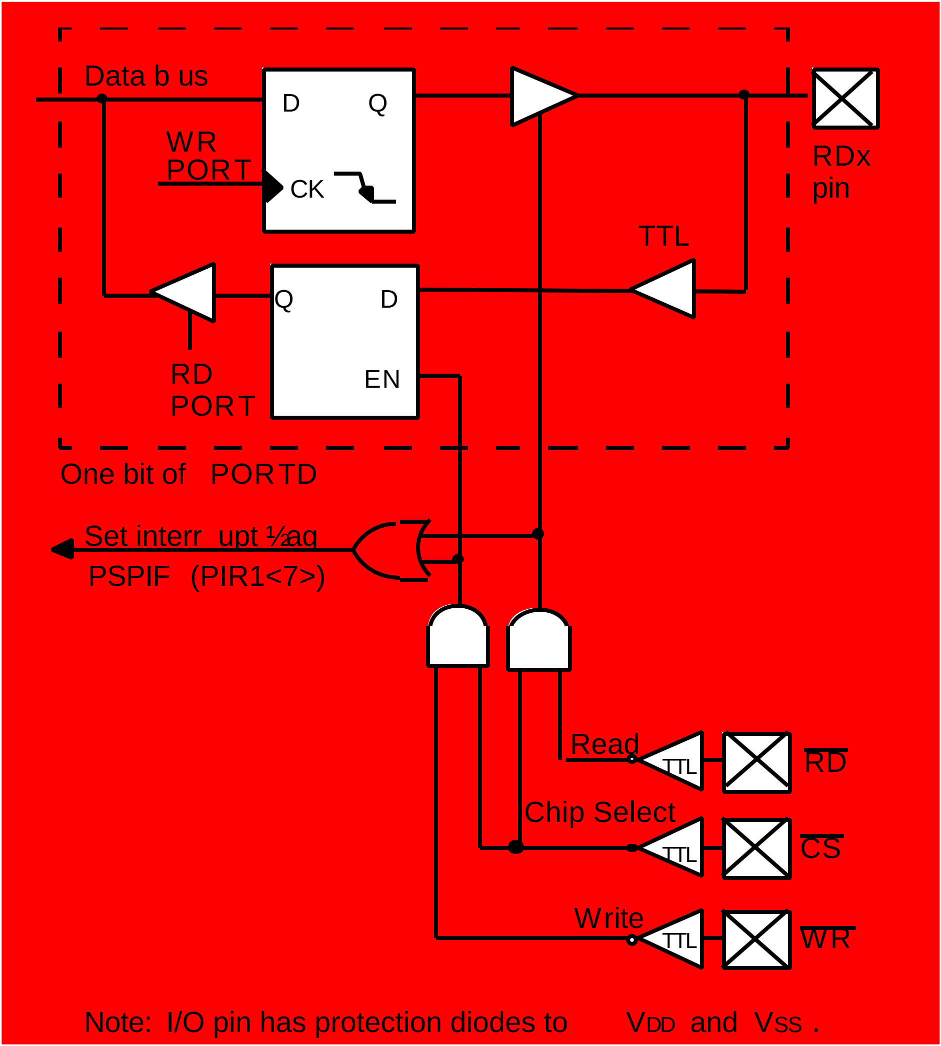


PICmicro MCU Peripherals
Parallel Slave Port: MCU Interface

- Direct interface to 8-bit microprocessor data bus
- Asynchronous operation (to external world)
- Interrupt generated on external read or write operation on parallel port
- Uses Port D and Port E
  - Port D: Data bus
  - Port E: Control signals (read, write, and chip select)

PICmicro MCU Peripherals

Parallel Slave Port: Block Diagram



Parallel Slave Port Setup

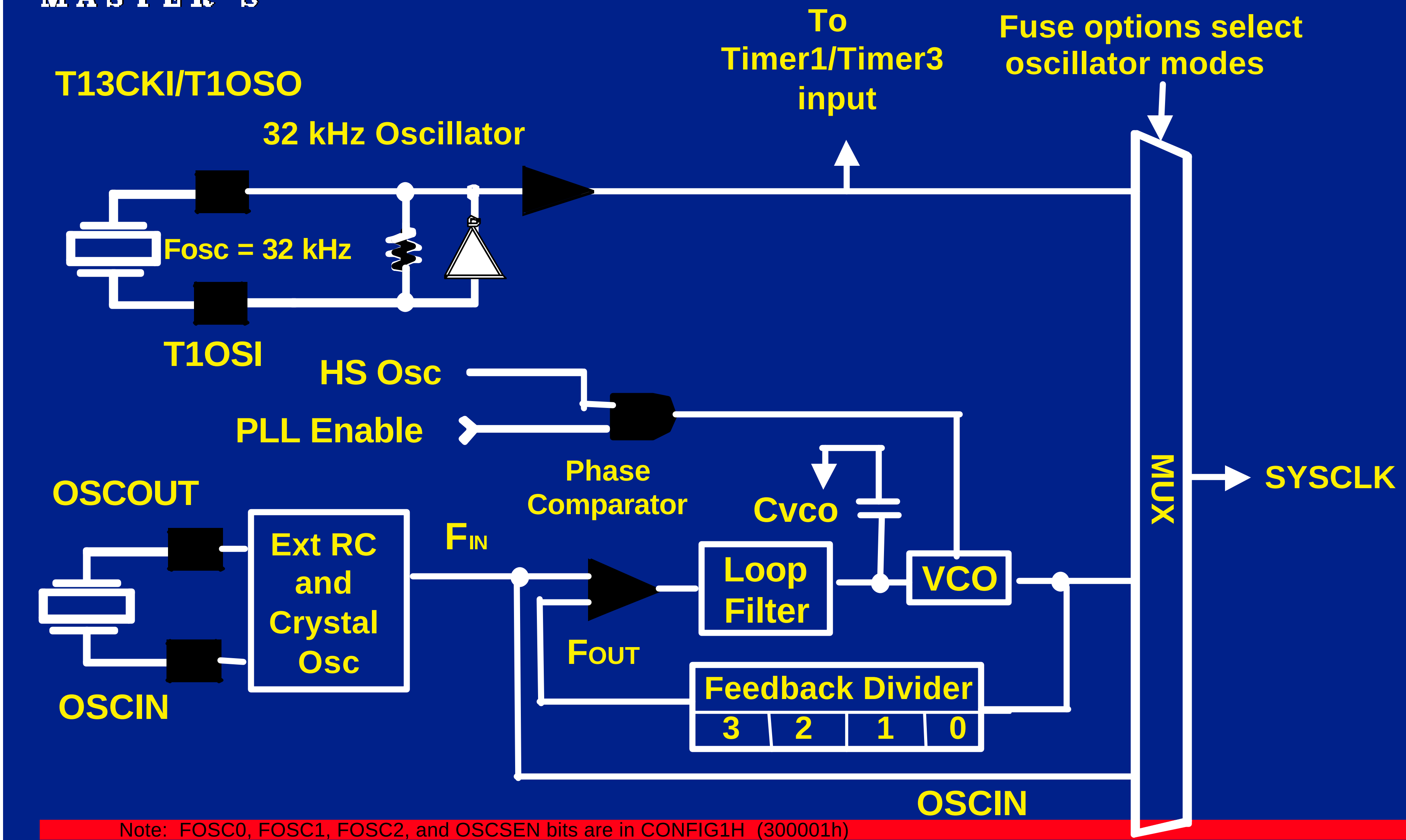
TRISE

bit 7							bit 0
IBF	OBF	IBOV	PSPMODE	-	TRISE2	TRISE1	TRISE0

IBF	**Input Buffer Full Status** 1 = A word has been received from the master into PORTD and is waiting to be read 0 = No word has been received from the master
OBF	**Output Buffer Full Status** 1 = The PORTD output buffer still holds a previously written word 0 = The PORTD output buffer has been read by the master and is now empty
IBOV	**Input buffer Overflow Detect Status (Must Be Cleared In Software)** 1 = The master wrote a byte before a previously written byte was read from PORTD 0 = No write overflow occurred
PSPMODE	**Parallel Slave Port Mode Selection** 1 = Enable Parallel Slave Port 0 = Disable Parallel Slave Port, PORTD and PORTE General Purpose I/Os
TRISE2:TRISE0	**PORTE, Pins RE2:RE0 Direction Control** 1 = RE x set to input 0 = RE x set to output

Special Features

New Oscillator Modes
PIC18F452 Oscillator Block Diagram



Note: FOSC0, FOSC1, FOSC2, and OSCSEN bits are in CONFIG1H (300001h)

PIC18 Special Features

Programmable Low Voltage Detect

- Provides “Early Warning”
- Programmable internal or external reference
  - Up to 14 internal reference voltages (2 - 4.77V)
- Operates during SLEEP
  - Low Voltage condition wakes-up/interrupts MCU
- Software Controlled enable/disable
  - Useful for low power applications

PIC18 Special Features
Programmable Brown-Out RESET

- Monitors operating voltage range
- Resets MCU when Vdd is below reference voltage
  - Deasserts RESET after Vdd is above reference voltage
  - Programmable internal reference
    - Up to 4 voltages (2.0, 2.7, 4.2, 4.5)
- Enabled via Configuration register

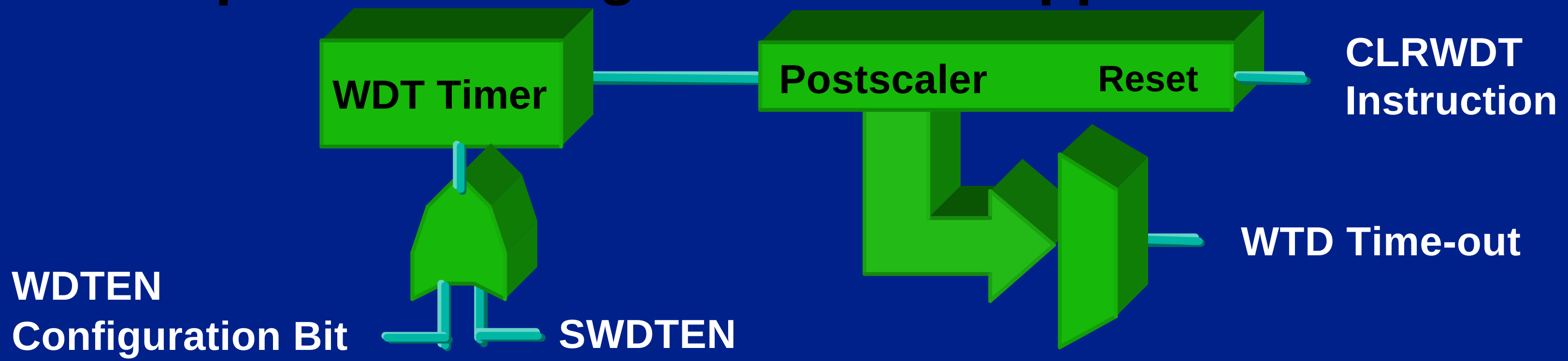
PIC18 Special Features

Watchdog Timer (WDT)

- Recovers from software malfunction
- Resets MCU if not attended on-time
  - Software must clear it periodically (**`CLRWDT`**)
- Programmable period
  - 18 ms to 3.0 s typical
- Configuration controlled postscaler
- Enabled via Configuration register or Software

Watchdog Enhancements Block Diagram

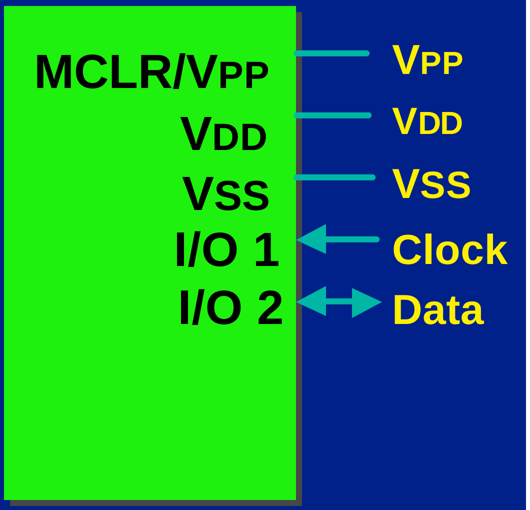
- The watchdog can be programmed on and off in software
  - **If Configuration bit WDTE = 1, the WDT cannot be turned off in software**
  - **If Configuration bit WDTE = 0, the software watchdog bit SWDTEN, can be used to enable/disable the WDOG timer**
  - **This is useful for applications that want to conserve power by turning off the WDT while in sleep or executing non-critical application code**



PIC18 Special Features

In-Circuit Serial Programming™

- Enhanced In-System Programming Method
- Uses only two pins to send/receive data
- Non-intrusive to normal operation
- Advantages of ICSP™ programming mode
  - Reduce cost of field upgrades
  - Calibrate and Serialize Systems during manufacturing
  - Reduce handling: Important for DIE and fine lead package



MCLR/VPP → VPP

VDD → VDD

VSS → VSS

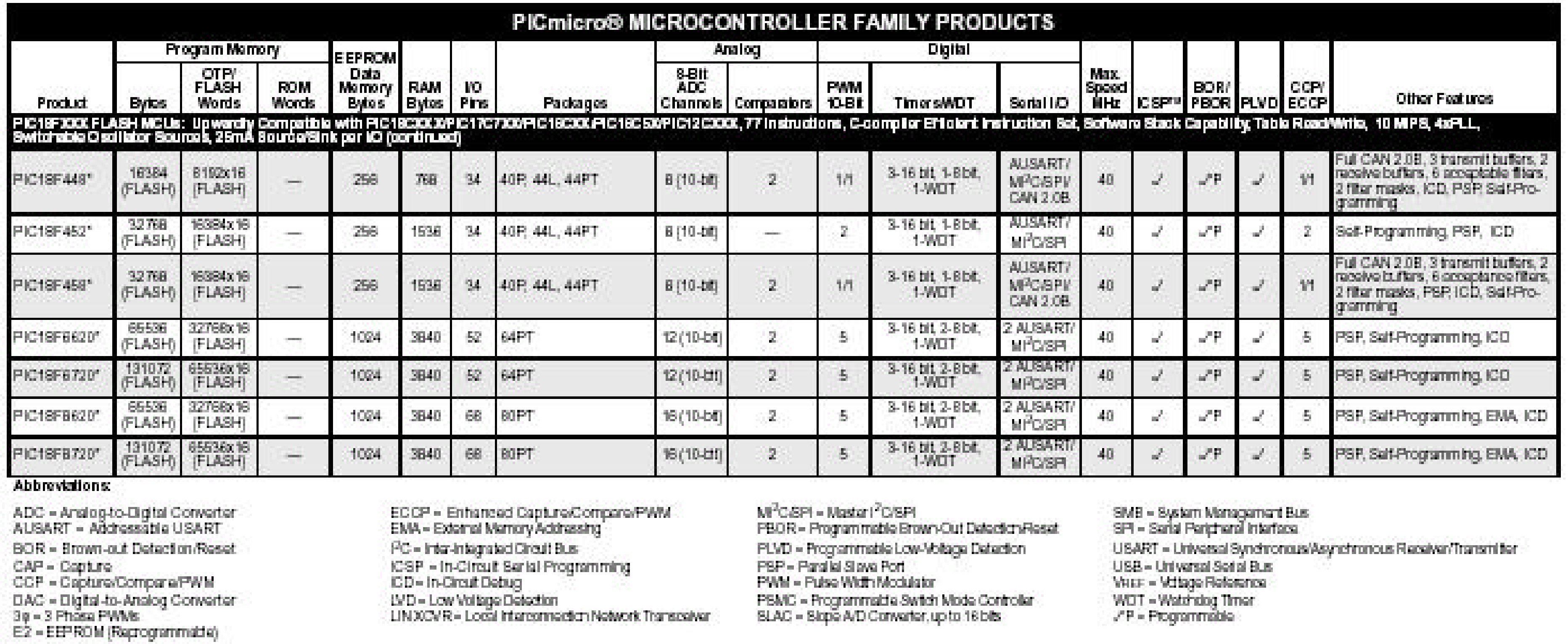
I/O 1 ← Clock

I/O 2 ↔ Data

PICmicro Line Card

PIC18FXXX Product Migration

PIC18FXXX Product Line Card



PICmicro® MICROCONTROLLER FAMILY PRODUCTS

PIC18FXXX FLASH MCUs: Upwardly Compatible with PIC18CXXX/PIC17C7XX/PIC16CXXX/PIC16C5X/PIC12CXXX, 77 Instructions, C-compiler Efficient Instruction Set, Software Stack Capability, Table Read/Write, 10 MIPS, 4xPLL, Switchable Oscillator Sources, 25mA Source/Sink per I/O (continued)

	Program Memory							Analog		Digital								
Product	Bytes	OTP/ FLASH Words	ROM Words	EEPROM Data Memory Bytes	RAM Bytes	I/O Pins	Packages	8-Bit ADC Channels	Comparators	PWM 10-Bit	Timers/WDT	Serial I/O	Max. Speed MHz	ICSP™	BOR/ PBOR	PLVD	CCP/ ECCP	Other Features
PIC18F448*	16384 (FLASH)	8192x16 (FLASH)	—	256	768	34	40P, 44L, 44PT	8 (10-bit)	2	1/1	3-16 bit, 1-8 bit, 1-WDT	AUSART/ MI^{2}C/SPI/ CAN 2.0B	40	✓	✓P	✓	1/1	Full CAN 2.0B, 3 transmit buffers, 2 receive buffers, 6 acceptable filters, 2 filter masks, ICD, PSP, Self-Programming
PIC18F452*	32768 (FLASH)	16384x16 (FLASH)	—	256	1536	34	40P, 44L, 44PT	8 (10-bit)	—	2	3-16 bit, 1-8 bit, 1-WDT	AUSART/ MI^{2}C/SPI	40	✓	✓P	✓	2	Self-Programming, PSP, ICD
PIC18F458*	32768 (FLASH)	16384x16 (FLASH)	—	256	1536	34	40P, 44L, 44PT	8 (10-bit)	2	1/1	3-16 bit, 1-8 bit, 1-WDT	AUSART/ MI^{2}C/SPI/ CAN 2.0B	40	✓	✓P	✓	1/1	Full CAN 2.0B, 3 transmit buffers, 2 receive buffers, 6 acceptance filters, 2 filter masks, PSP, ICD, Self-Programming
PIC18F6620*	65536 (FLASH)	32768x16 (FLASH)	—	1024	3840	52	64PT	12 (10-bit)	2	5	3-16 bit, 2-8 bit, 1-WDT	2 AUSART/ MI^{2}C/SPI	40	✓	✓P	✓	5	PSP, Self-Programming, ICD
PIC18F6720*	131072 (FLASH)	65536x16 (FLASH)	—	1024	3840	52	64PT	12 (10-bit)	2	5	3-16 bit, 2-8 bit, 1-WDT	2 AUSART/ MI^{2}C/SPI	40	✓	✓P	✓	5	PSP, Self-Programming, ICD
PIC18F8620*	65536 (FLASH)	32768x16 (FLASH)	—	1024	3840	68	80PT	16 (10-bit)	2	5	3-16 bit, 2-8 bit, 1-WDT	2 AUSART/ MI^{2}C/SPI	40	✓	✓P	✓	5	PSP, Self-Programming, EMA, ICD
PIC18F8720*	131072 (FLASH)	65536x16 (FLASH)	—	1024	3840	68	80PT	16 (10-bit)	2	5	3-16 bit, 2-8 bit, 1-WDT	2 AUSART/ MI^{2}C/SPI	40	✓	✓P	✓	5	PSP, Self-Programming, EMA, ICD

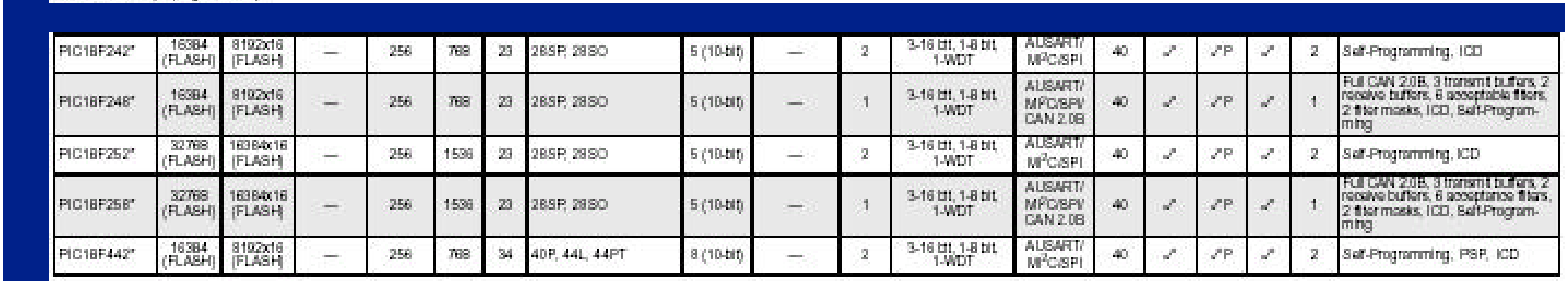
Abbreviations:

ADC = Analog-to-Digital Converter
AUSART = Addressable USART
BOR = Brown-out Detection/Reset
CAP = Capture
CCP = Capture/Compare/PWM
DAC = Digital-to-Analog Converter
3φ = 3 Phase PWMs
E2 = EEPROM (Reprogrammable)

ECCP = Enhanced Capture/Compare/PWM
EMA = External Memory Addressing
I^{2}C = Inter-Integrated Circuit Bus
ICSP = In-Circuit Serial Programming
ICD = In-Circuit Debug
LVD = Low Voltage Detection
LINXCVR = Local Interconnection Network Transceiver

MI^{2}C/SPI = Master I^{2}C/SPI
PBOR = Programmable Brown-Out Detection/Reset
PLVD = Programmable Low-Voltage Detection
PSP = Parallel Slave Port
PWM = Pulse Width Modulator
PSMC = Programmable Switch Mode Controller
SLAC = Slope A/D Converter, up to 16 bits

SMB = System Management Bus
SPI = Serial Peripheral Interface
USART = Universal Synchronous/Asynchronous Receiver/Transmitter
USB = Universal Serial Bus
VREF = Voltage Reference
WDT = Watchdog Timer
✓P = Programmable

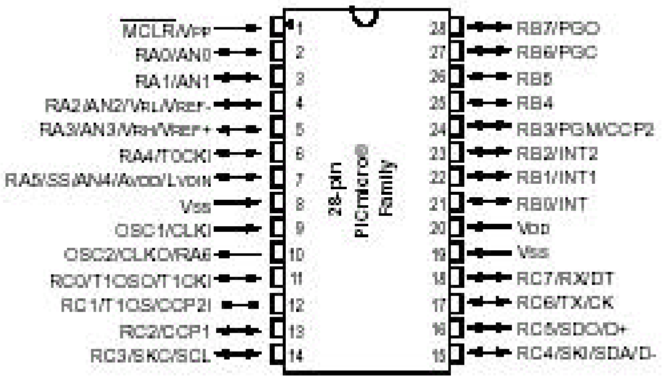


PIC18F242*	16384 (FLASH)	8192x16 (FLASH)	—	256	768	23	28SP, 28SO	5 (10-bit)	—	2	3-16 bit, 1-8 bit, 1-WDT	AUSART/ MI^{2}C/SPI	40	✓	✓P	✓	2	Self-Programming, ICD
PIC18F248*	16384 (FLASH)	8192x16 (FLASH)	—	256	768	23	28SP, 28SO	5 (10-bit)	—	1	3-16 bit, 1-8 bit, 1-WDT	AUSART/ MI^{2}C/SPI/ CAN 2.0B	40	✓	✓P	✓	1	Full CAN 2.0B, 3 transmit buffers, 2 receive buffers, 6 acceptable filters, 2 filter masks, ICD, Self-Programming
PIC18F252*	32768 (FLASH)	16384x16 (FLASH)	—	256	1536	23	28SP, 28SO	5 (10-bit)	—	2	3-16 bit, 1-8 bit, 1-WDT	AUSART/ MI^{2}C/SPI	40	✓	✓P	✓	2	Self-Programming, ICD
PIC18F258*	32768 (FLASH)	16384x16 (FLASH)	—	256	1536	23	28SP, 28SO	5 (10-bit)	—	1	3-16 bit, 1-8 bit, 1-WDT	AUSART/ MI^{2}C/SPI/ CAN 2.0B	40	✓	✓P	✓	1	Full CAN 2.0B, 3 transmit buffers, 2 receive buffers, 6 acceptance filters, 2 filter masks, ICD, Self-Programming
PIC18F442*	16384 (FLASH)	8192x16 (FLASH)	—	256	768	34	40P, 44L, 44PT	8 (10-bit)	—	2	3-16 bit, 1-8 bit, 1-WDT	AUSART/ MI^{2}C/SPI	40	✓	✓P	✓	2	Self-Programming, PSP, ICD

PICmicro 28/40 Pin Device Compatibility

PIN AND CODE COMPATIBILITY CHART (CONTINUED)

28-pin PICmicro® MCU Family

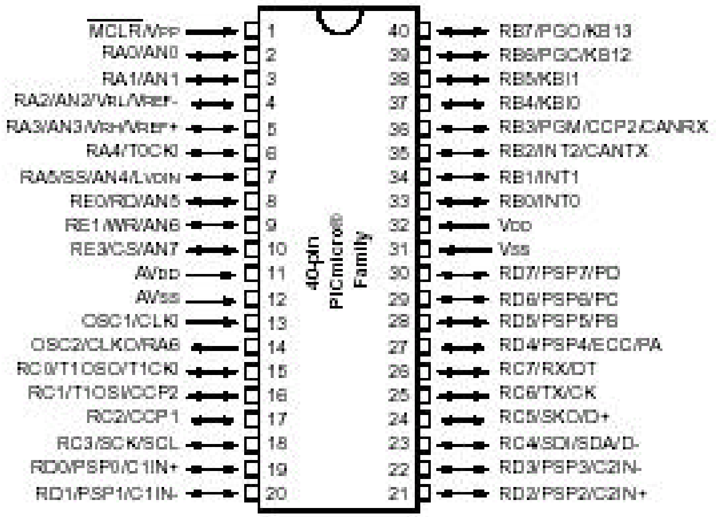


PIC16C62B
PIC16CR63
PIC16C63A
PIC16C66
PIC16CR72
PIC16C72A
PIC16F72
PIC16C73B
PIC16C76
PIC16F73

PIC16F76
PIC16C642
PIC16C745
PIC16C773
PIC16F870
PIC16F872
PIC16F873
PIC16F873A

PIC16F876
PIC16F876A
PIC18C242
PIC18C252
PIC18F2220
PIC18F2320
PIC18F242
PIC18F252
PIC18F248
PIC18F258

40-pin PICmicro® MCU Family

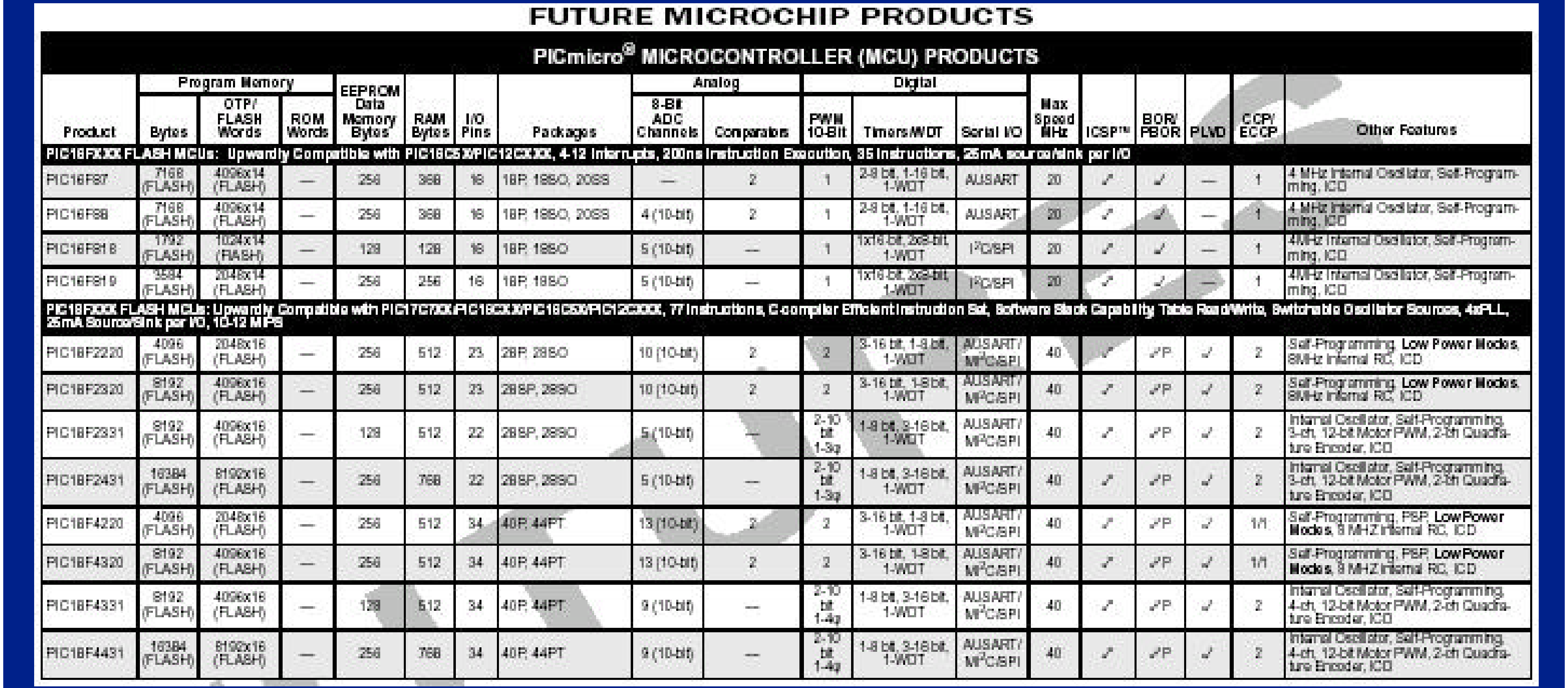


PIC16CR65
PIC16C65B
PIC16C67
PIC16C662
PIC16C74B
PIC16C77
PIC16F74
PIC16F77

PIC16C765
PIC16C774
PIC16F871
PIC16F874
PIC16F874A
PIC16F877
PIC16F877A

PIC18C442
PIC18C452
PIC18F4220
PIC18F4320
PIC18F442
PIC18F452
PIC18F448
PIC18F458

Future Products



FUTURE MICROCHIP PRODUCTS

PICmicro® MICROCONTROLLER (MCU) PRODUCTS

Product	Program Memory			EEPROM Data Memory Bytes	RAM Bytes	I/O Pins	Packages	Analog		Digital			Max Speed MHz	ICSP™	BOR/ PBOR	PLVD	CCP/ ECCP	Other Features
	Bytes	OTP/ FLASH Words	ROM Words					8-Bit ADC Channels	Comparators	PWM 10-Bit	Timers/WDT	Serial I/O						
PIC16FXXX FLASH MCUs: Upwardly Compatible with PIC16C5X/PIC12CXXX, 4-12 Interrupts, 200ns Instruction Execution, 35 Instructions, 25mA source/sink per I/O																		
PIC16F87	7168 (FLASH)	4096x14 (FLASH)	—	256	368	16	18P, 18SO, 20SS	—	2	1	2-8 bit, 1-16 bit, 1-WDT	AUSART	20	✓	✓	—	1	4 MHz Internal Oscillator, Self-Programming, ICD
PIC16F88	7168 (FLASH)	4096x14 (FLASH)	—	256	368	16	18P, 18SO, 20SS	4 (10-bit)	2	1	2-8 bit, 1-16 bit, 1-WDT	AUSART	20	✓	✓	—	1	4 MHz Internal Oscillator, Self-Programming, ICD
PIC16F818	1792 (FLASH)	1024x14 (FLASH)	—	128	128	16	18P, 18SO	5 (10-bit)	—	1	1x16-bit, 2x8-bit, 1-WDT	I^{2}C/SPI	20	✓	✓	—	1	4MHz Internal Oscillator, Self-Programming, ICD
PIC16F819	3584 (FLASH)	2048x14 (FLASH)	—	256	256	16	18P, 18SO	5 (10-bit)	—	1	1x16-bit, 2x8-bit, 1-WDT	I^{2}C/SPI	20	✓	✓	—	1	4MHz Internal Oscillator, Self-Programming, ICD
PIC18FXXX FLASH MCUs: Upwardly Compatible with PIC17C7XX/PIC16CXXX/PIC18C5X/PIC12CXXX, 77 Instructions, C-compiler Efficient Instruction Set, Software Stack Capability Table Read/Write, Switchable Oscillator Sources, 4xPLL, 25mA Source/Sink per I/O, 10-12 MIPS																		
PIC18F2220	4096 (FLASH)	2048x16 (FLASH)	—	256	512	23	28P, 28SO	10 (10-bit)	2	2	3-16 bit, 1-8 bit, 1-WDT	AUSART/ MI^{2}C/SPI	40	✓	✓P	✓	2	Self-Programming, **Low Power Modes**, 8MHz Internal RC, ICD
PIC18F2320	8192 (FLASH)	4096x16 (FLASH)	—	256	512	23	28SP, 28SO	10 (10-bit)	2	2	3-16 bit, 1-8 bit, 1-WDT	AUSART/ MI^{2}C/SPI	40	✓	✓P	✓	2	Self-Programming, **Low Power Modes**, 8MHz Internal RC, ICD
PIC18F2331	8192 (FLASH)	4096x16 (FLASH)	—	128	512	22	28SP, 28SO	5 (10-bit)	—	2-10 bit 1-3g	1-8 bit, 3-16 bit, 1-WDT	AUSART/ MI^{2}C/SPI	40	✓	✓P	✓	2	Internal Oscillator, Self-Programming, 3-ch, 12-bit Motor PWM, 2-ch Quadrature Encoder, ICD
PIC18F2431	16384 (FLASH)	8192x16 (FLASH)	—	256	768	22	28SP, 28SO	5 (10-bit)	—	2-10 bit 1-3g	1-8 bit, 3-16 bit, 1-WDT	AUSART/ MI^{2}C/SPI	40	✓	✓P	✓	2	Internal Oscillator, Self-Programming, 3-ch, 12-bit Motor PWM, 2-ch Quadrature Encoder, ICD
PIC18F4220	4096 (FLASH)	2048x16 (FLASH)	—	256	512	34	40P, 44PT	13 (10-bit)	2	2	3-16 bit, 1-8 bit, 1-WDT	AUSART/ MI^{2}C/SPI	40	✓	✓P	✓	1/1	Self-Programming, PSP, **Low Power Modes**, 8 MHZ Internal RC, ICD
PIC18F4320	8192 (FLASH)	4096x16 (FLASH)	—	256	512	34	40P, 44PT	13 (10-bit)	2	2	3-16 bit, 1-8 bit, 1-WDT	AUSART/ MI^{2}C/SPI	40	✓	✓P	✓	1/1	Self-Programming, PSP, **Low Power Modes**, 8 MHZ Internal RC, ICD
PIC18F4331	8192 (FLASH)	4096x16 (FLASH)	—	128	512	34	40P, 44PT	9 (10-bit)	—	2-10 bit 1-4g	1-8 bit, 3-16 bit, 1-WDT	AUSART/ MI^{2}C/SPI	40	✓	✓P	✓	2	Internal Oscillator, Self-Programming, 4-ch, 12-bit Motor PWM, 2-ch Quadrature Encoder, ICD
PIC18F4431	16384 (FLASH)	8192x16 (FLASH)	—	256	768	34	40P, 44PT	9 (10-bit)	—	2-10 bit 1-4g	1-8 bit, 3-16 bit, 1-WDT	AUSART/ MI^{2}C/SPI	40	✓	✓P	✓	2	Internal Oscillator, Self-Programming, 4-ch, 12-bit Motor PWM, 2-ch Quadrature Encoder, ICD