11009 PRC

PIC18F Peripherals

UNIVERSITY OF MICROCHIP UOFM MASTERS 2007

Objectives

- When you walk out of this class, you will know
  - PIC18F Architecture
  - PIC18F Basic Peripherals and Interrupts
  - How to “Develop applications using PIC18F”
  - PIC18F Extended Instruction set advantage
  - PIC18F Special Features

University of Microchip UofM MASTERs 2007

Agenda

- **Architectural Overview**
  - **Programmers' Model**
    - Program Memory
    - Stack
    - Data Memory
  - **Instruction Set Overview**
- **Interrupt Handling and latency**
- **Basic Peripherals Discussion and LABS**
  - **I/O Ports**
  - **Development Tools**
    - LAB 1: Development tools usage, Initialization,
    - read and write of I/O ports
  - **Analog: Comparators & Voltage Reference & ADC**
    - LAB 2: Initialization and ADC conversion

University of Microchip UofM MASTERs 2007

Agenda (Continued..)

- **Basic Peripherals Discussion and LABS *(continued…)***
  - **Timers**
    - LAB 3: Counters
  - **CCP (Capture, Compare & PWM)**
    - LAB 4 and 5: Generating PWM pulses and Measuring Frequency/Duty Cycle
  - **MSSP (I^2C™ & SPI / Microwire)**
    - LAB 6: I^2C Initialization and communication with temperature sensor
  - **Table Read and Write Operations**
  - **USART**
    - LAB 7: Initialization, communication with host and Table read Write operations

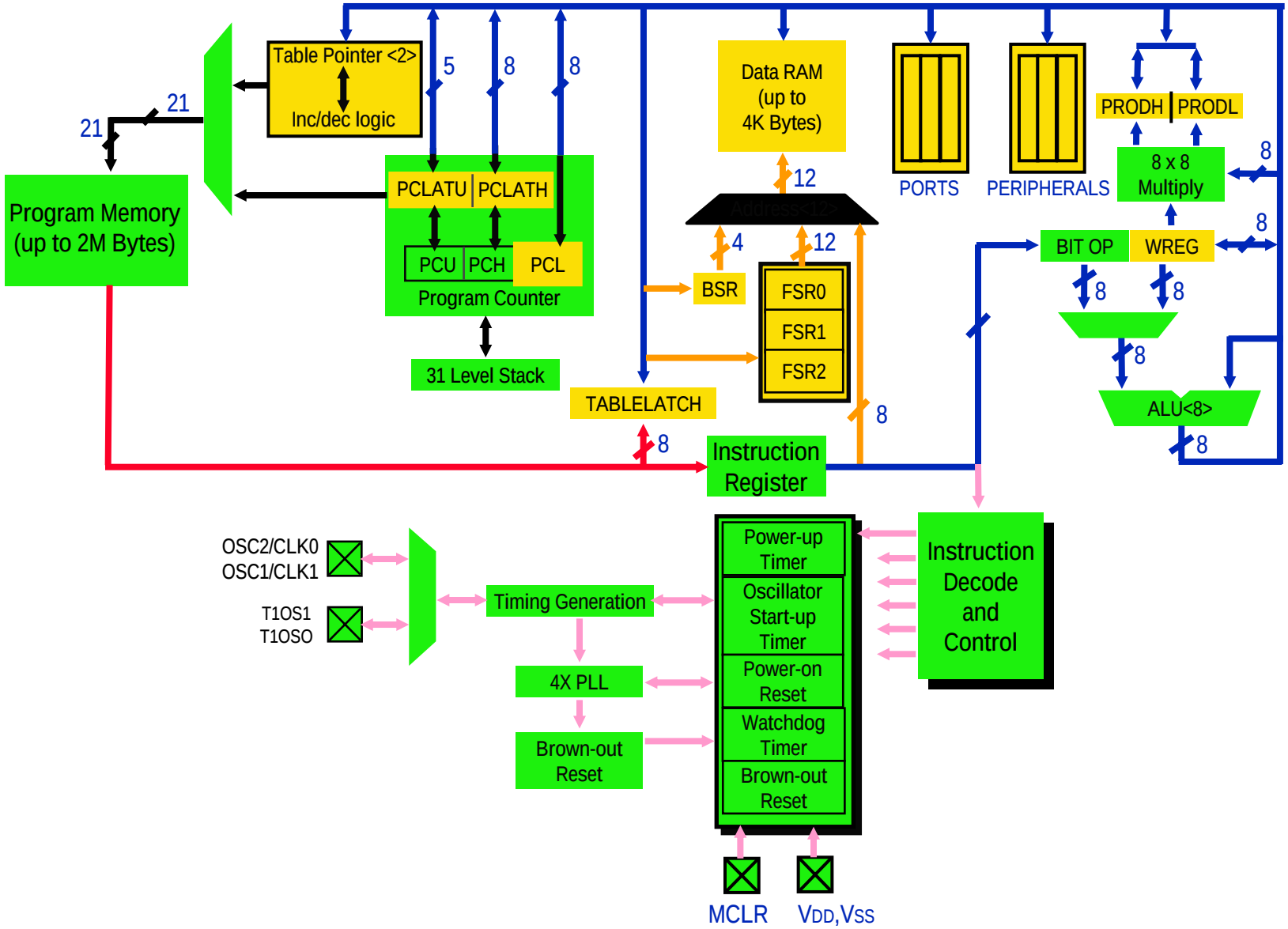
UNIVERSITY OF MICROCHIP UOFM MASTERS 2007

Agenda (Continued..)

- **Extended Architecture advantages**
  - LAB 8
    - Extended Instruction set advantage
- **Oscillators and Power-Saving modes**
- **Special features**
  - **PLVD**
  - **PBOR**
  - **ICSP™ programming capability**
  - **WDT**
  - **Resets**
    - LAB 9
      - Example: How to determine the cause of reset in real application

PIC18F
Basic Architecture

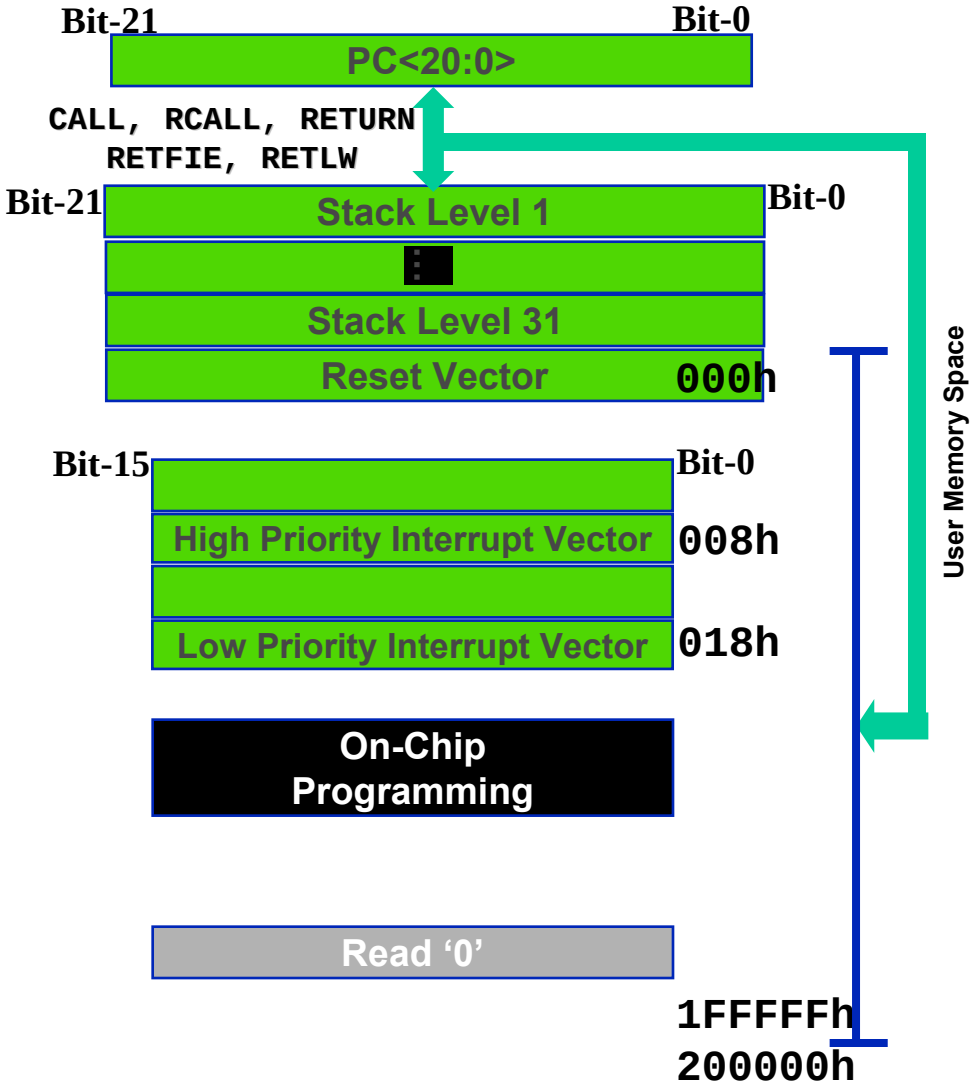
PIC18F Block Diagram



PIC18F Program Memory and Stack

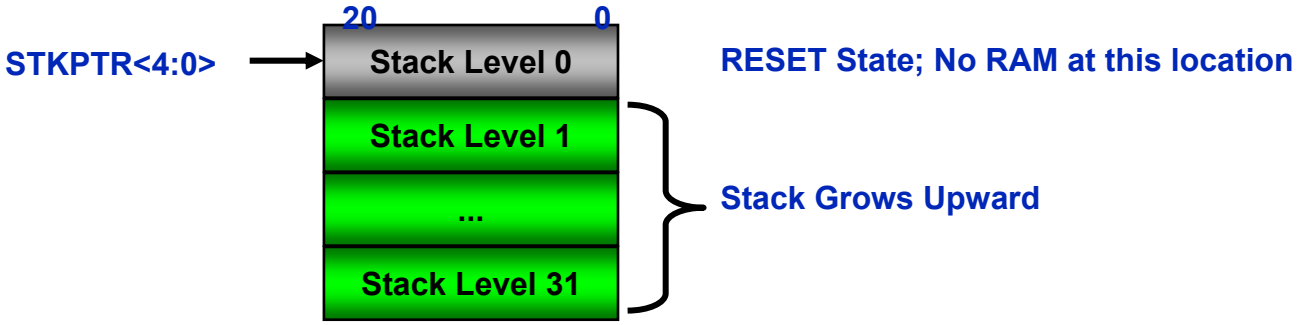
PIC18F MCU Architecture Program Memory

- Maximum 2M Bytes of program memory space
  - 21-bit PC (PCU PCH PCL)
- Reset Vector at 0000h
- Interrupt Vector at 0008h and 0018h
- PC increments in words LSB always = 0
- PC<0> cannot be changed



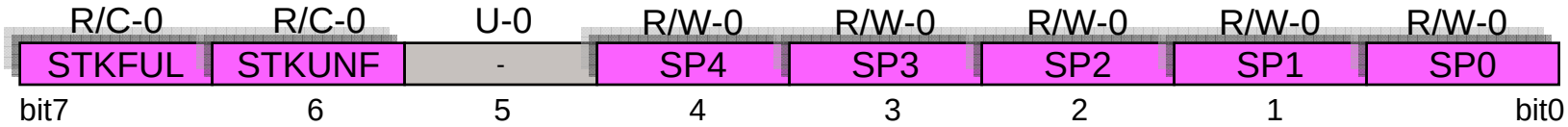
PIC18F Architecture Stack Memory

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



- 5-bit Stack pointer addresses 31 levels
  - Stack pointer does not roll over to 0 at end of stack
- Top-of-Stack = TOSU:TOSH:TOSL
  - Readable and Writeable

STKPTR - Stack Pointer Register



bit 7: **STKFUL:** Stack Full Flag bit

1 = Stack full or overflow occurred

0 = Reset or cleared by user software

bit 6: **STKUNF:** Stack Underflow Flag bit

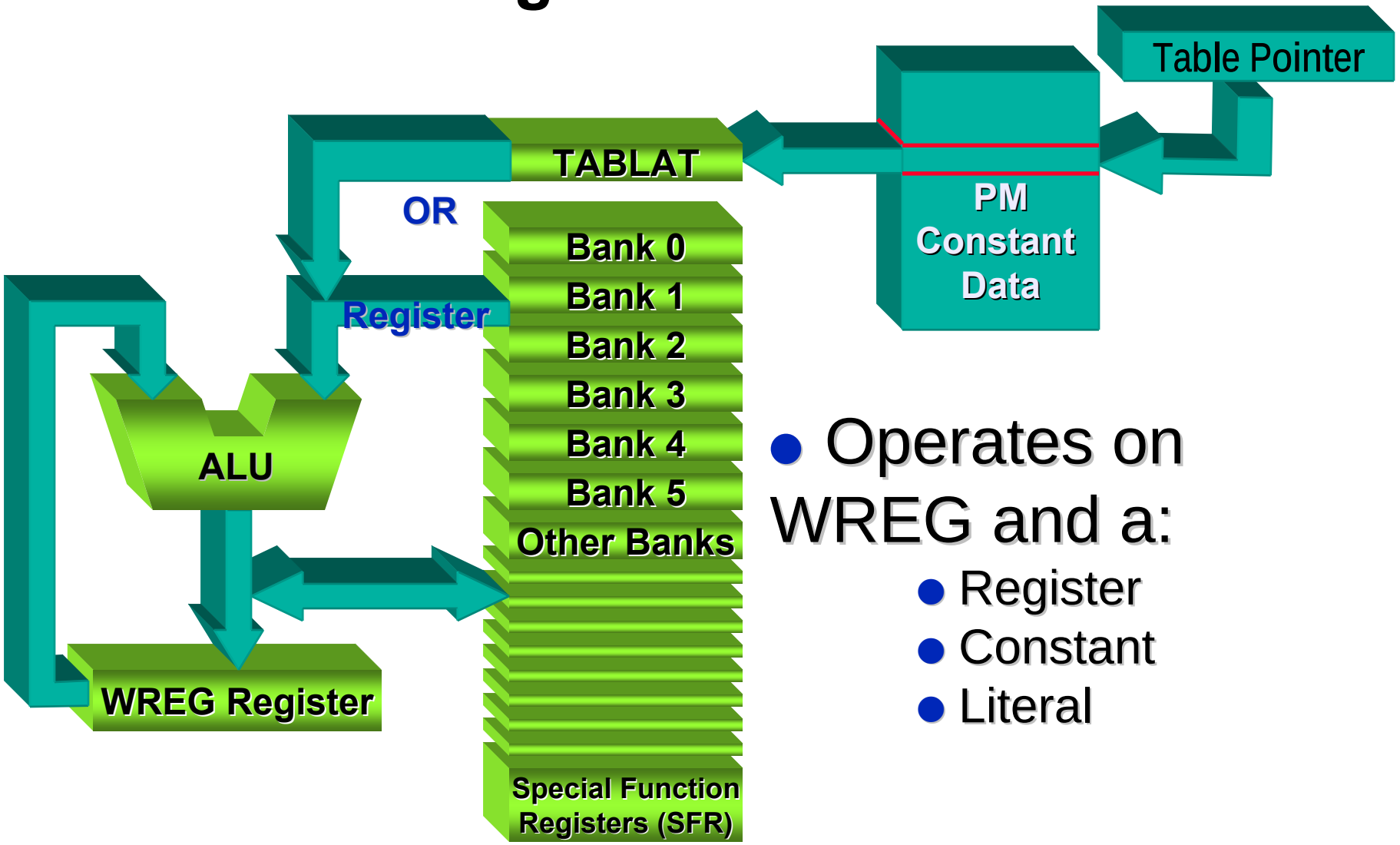
1 = Stack underflow occurred

0 = Reset or cleared by user software

bit 4-0: **SP4:SP0:** Stack Pointer bits

PIC18F
Data Memory

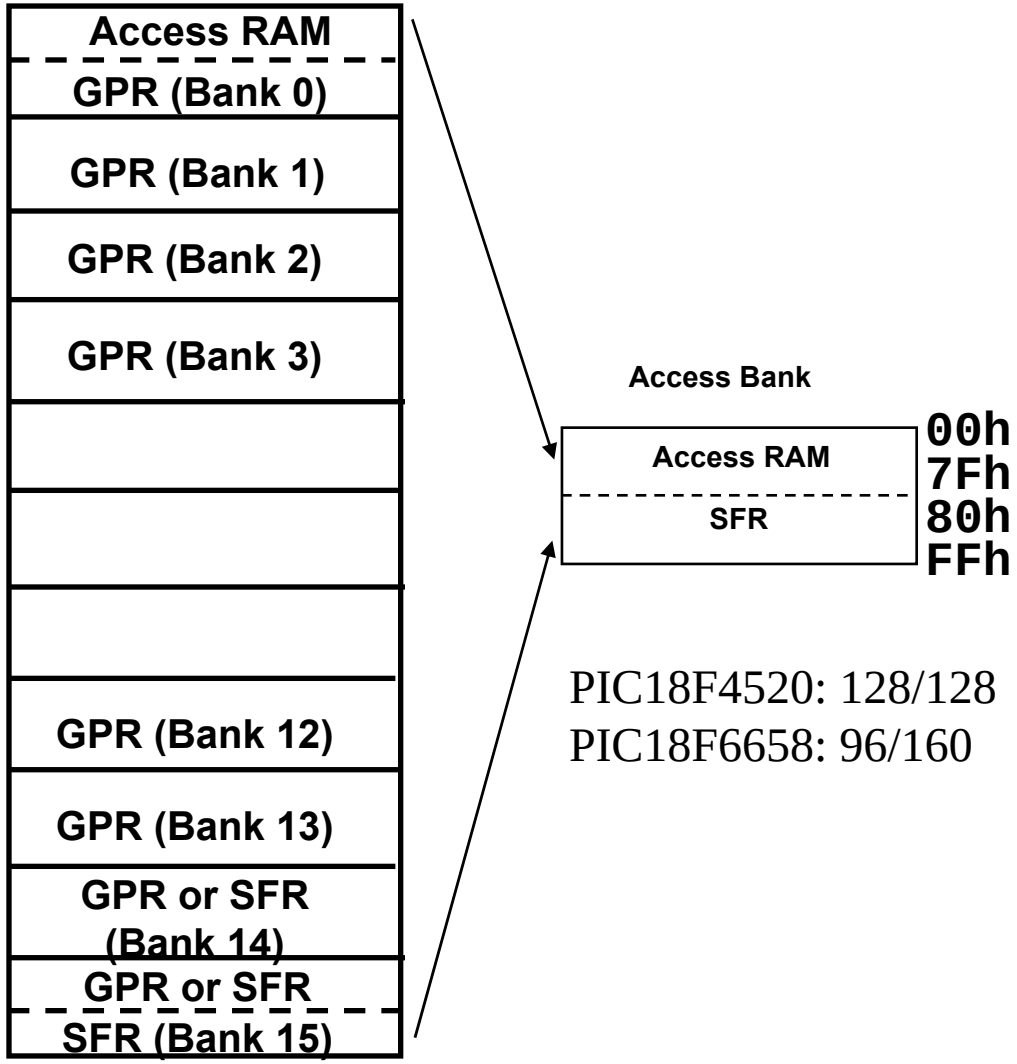
PIC18F Architecture
Programmer's Model



- Operates on WREG and a:
  - Register
  - Constant
  - Literal

PIC18F Architecture Data Memory

- Access Bank
  - Part of Bank 0
  - Part of Bank 15
- Accessible from any bank regardless of the state of BSR
- Minimizes bank switching
- Used for global vars, context save/restore, etc.
- Size of Access bank depends on device



UNIVERSITY OF MICROCHIP
UOFM
MASTERS 2007

PIC18F Instruction Set Overview

PIC18F: Instruction Set Overview

- Data move instructions
  - Data memory to data memory
  - Data memory to program memory (vice versa)
  - Literal
- Arithmetic, Logic, Shift instructions
- Single cycle 8 x 8 multiply (100ns)
- Powerful bit manipulation
  - Single cycle bit set, clear or toggle
  - Operate directly on all registers including I/O
- Branching / Conditional Branching

PIC18F: Instruction Set Overview

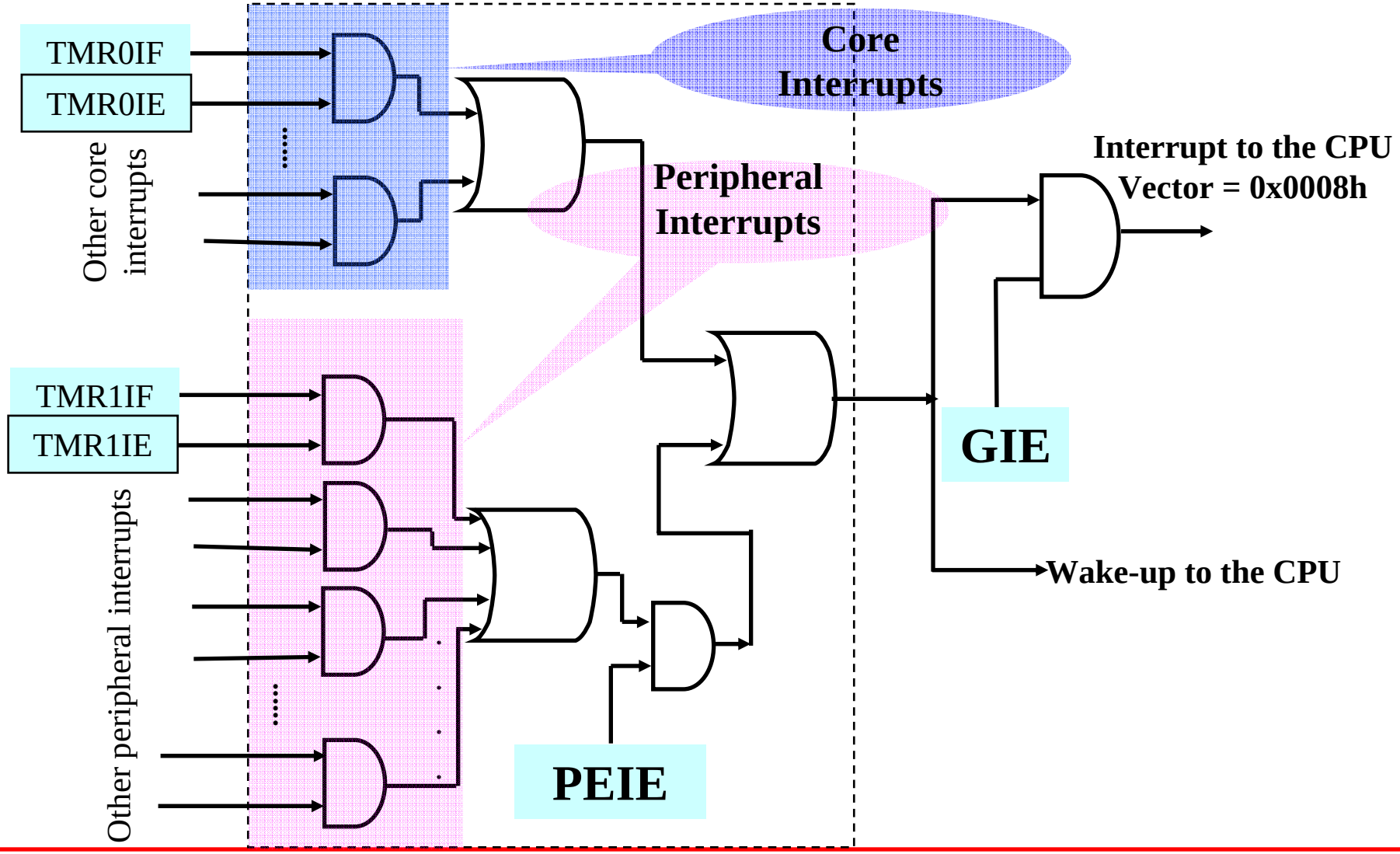
- 77 standard instructions
  - 73 are single word
  - 4 two-word `MOVFF`, `MOVLB`, `CALL`, `GOTO`
- Most of the instructions are single cycle
  - 4 two word instructions are two cycle
  - 10 conditional, unconditional branches, calls are 1 or 2 cycle
  - 8 table operation instructions are two cycle
  - 3 returns are 2 cycle
  - 10 conditional `SKIP` instructions are 1 or 2 cycle
- TABLE Read/Write
- 8 extended instructions

PIC18F
Interrupt Structure

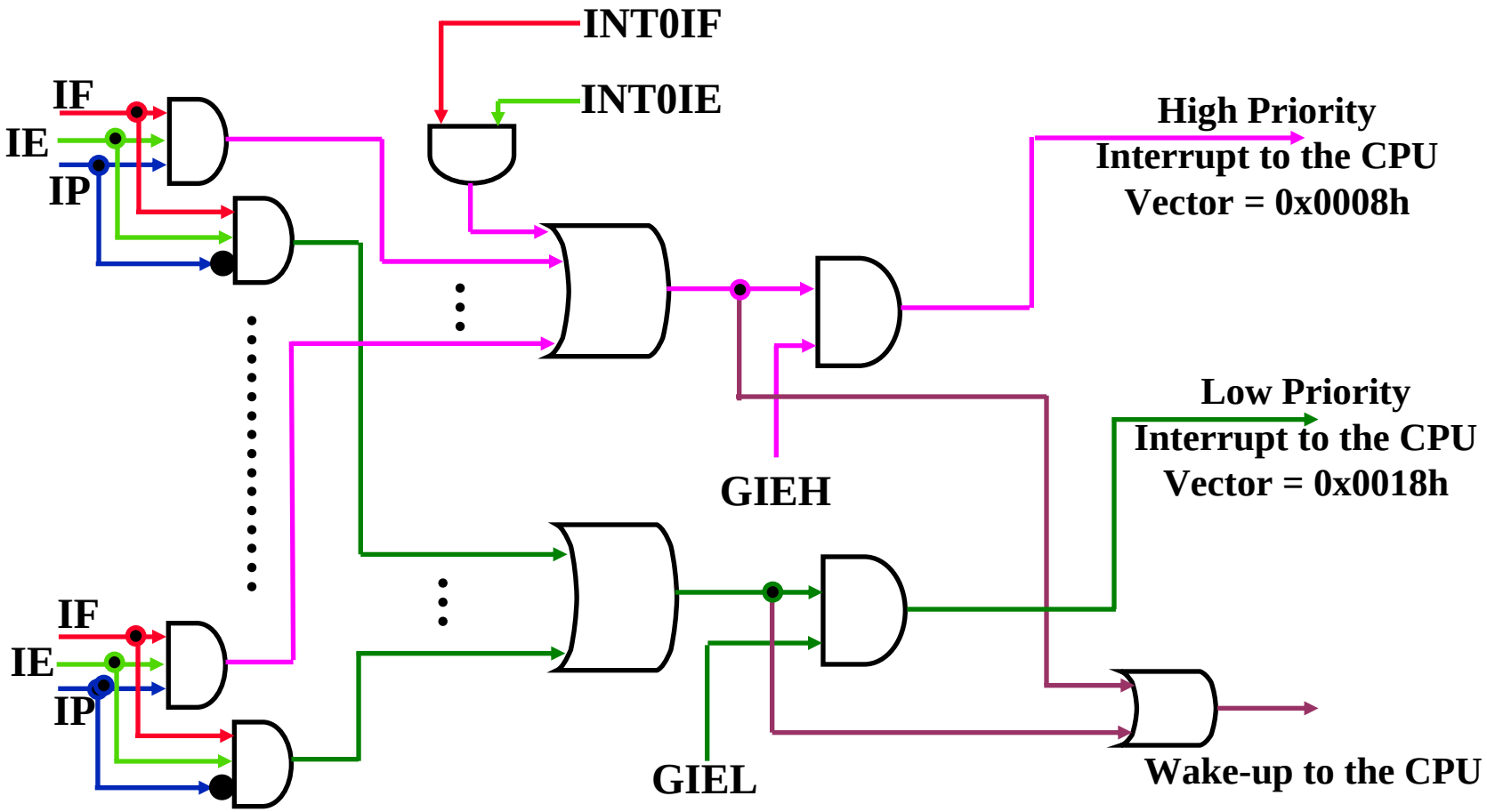
PIC18F Interrupt Overview

- Multiple internal and external interrupt sources
- Each interrupt has an individual interrupt flag and mask bit
- All interrupts can be masked by GIE bit
- Has two modes
  - Legacy mode (By clearing the bit IPEN in RCON)
  - Priority mode (By setting the bit IPEN in RCON)
- Each peripheral interrupt can be configured for High/Low hardware priority in priority mode
  - High Priority Vector at 000008h (Default)
  - Low Priority Vector at 000018h
- Fast context save on Interrupt
- Most interrupts wake processor from sleep
- Interrupt latency Three/Four instruction cycles

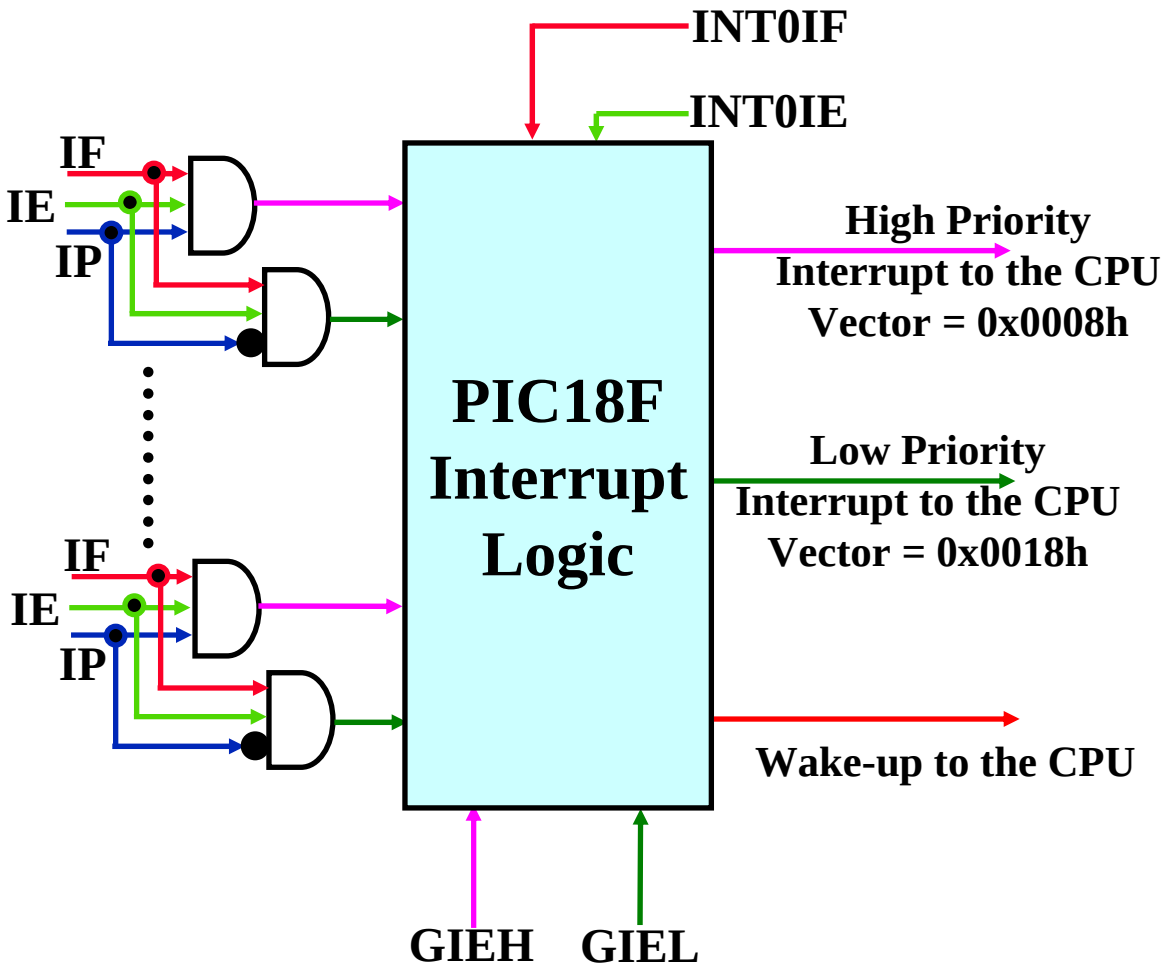
PIC18F
Interrupt Configuration (Legacy mode)



PIC18F
Interrupt Configuration (Priority Mode)



PIC18F
Interrupt Configuration (Priority Mode)



PIC18F Interrupt Overview

- **Interrupts**
  - **Multiple priority levels for interrupts**
    - High and Low priority
  - **Vectors corresponding to priority levels**
  - **Four external interrupt pins INT0, INT1, INT2, INT3**
  - **Each interrupt has an enable bit, flag bit and priority bit**
    - Except INT0, which is always high priority

PIC18F Interrupt Overview

- Shadow Registers: Fast Call / Return
  - Calls or Returns can choose to push / pop key registers to shadow registers
  - 1 level deep for W, STATUS, BSR registers
  - Selected by “s” bit in Call / Return instructions
    `CALL FAST`
- Shadow Registers: Interrupts
  - Any interrupt will push the shadow registers
  - If using `RETFIE` to pop shadow registers, the ISR must not be interrupted (low priority)
  - Shadow registers only useful for high priority

Shadow Registers

- One level deep
- The “s” bit in the CALL instruction enables a ‘Fast’ CALL/RETURN sequence
- If s = 1, WREG, STATUS, BSR registers are saved in the shadow registers
- Instructions with s = 1 will restore shadows back to WREG, STATUS, BSR registers
  - RETURN FAST
  - RETFIE FAST

Peripherals

PIC18F Peripherals

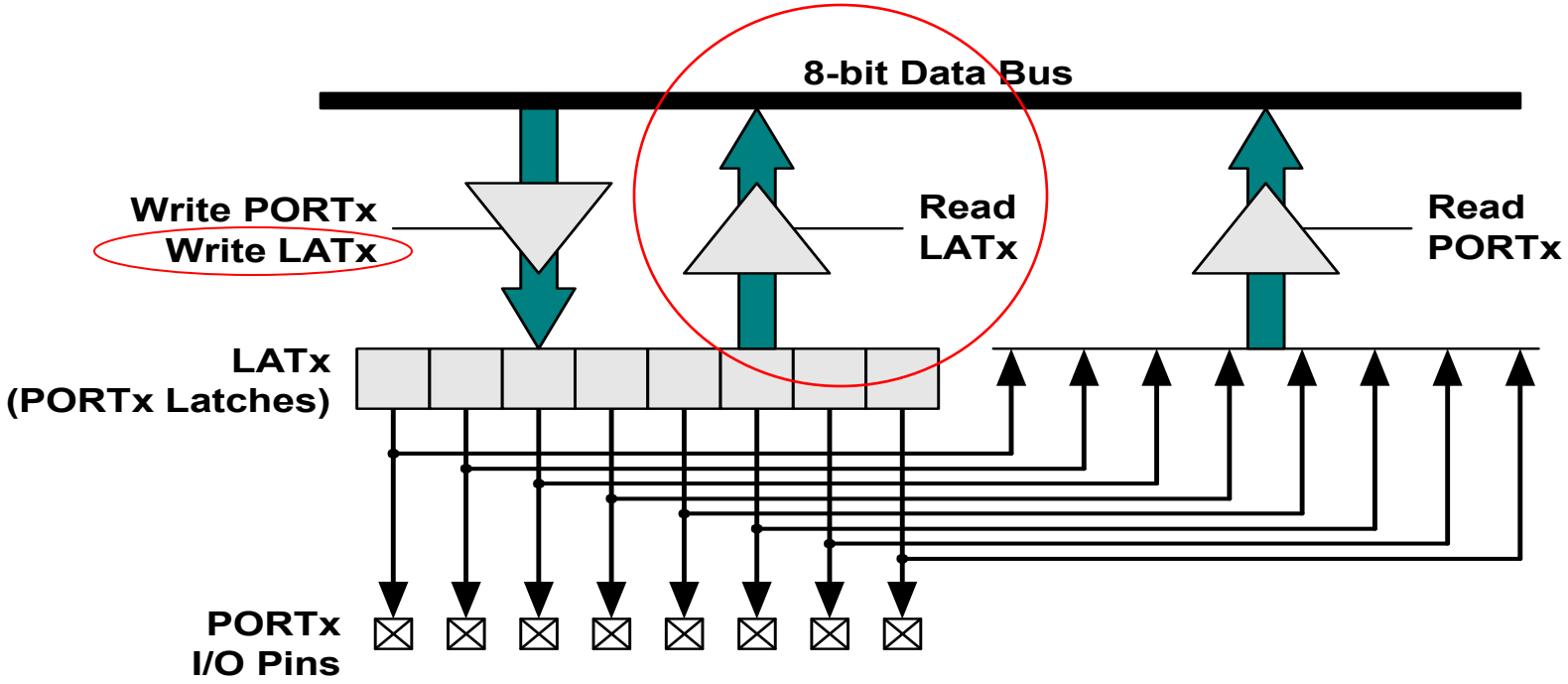
- Digital I/O Ports
- Analog Comparator
- Analog-To-Digital Converter
- Timer0, 1, 2, 3
- Compare/Capture/PWM (CCP)
- Addressable USART (AUSART)
- Master Synchronous Serial Port (MSSP)
- Parallel Slave Port (PSP)

PIC18F Peripherals
Digital I/O Ports

- Up to 66 bidirectional I/O pins, some multiplexed with peripheral functions
- High drive capability:
  - Max sink current per pin = 25 mA
  - Max source current per pin = 25 mA
- Can directly drive LEDs
- Direct, single cycle bit manipulation
- Each pin has individual direction control under software
- All pins have ESD protection diodes
  - 4 kV based on human body model
- Pin RA4 is usually open drain – check the device
- All I/O pins default to inputs on startup (high impedance state)
- Pins multiplexed with analog functions default to analog inputs

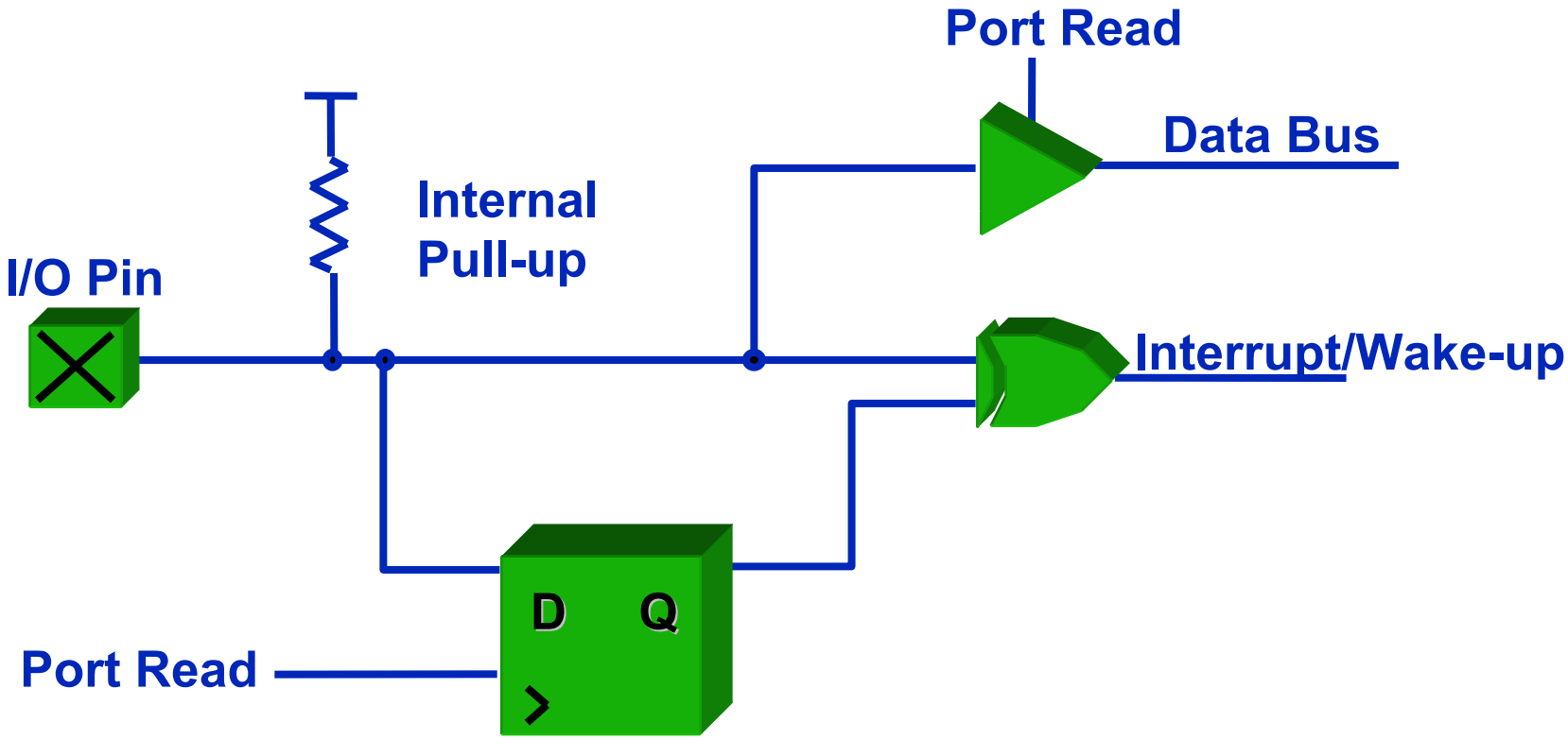
PIC18F Peripherals
Digital I/O Ports: Use LATx Register!

- Latch registers added to PIC18 memory map to eliminate RMW situations.
  - MOVF PORTX,w ; Reads PORTX *Pins* (actual state of pin)
  - MOVF LATX,w ; Reads PORTX *Output Latch Register* (desired state of pin)
  - MOVWF PORTX ; Writes to PORTX/LATX
  - MOVWF LATX ; Writes to PORTX/LATX



PIC18F Peripherals
PORTB : Interrupt on Change

- **Internal Pull-ups and Wake-up/Interrupt On Change feature**



PIC18F Peripherals

PORTB : Configuring the Interrupt on Change

INTCON2 : Interrupt Control Register 2

R/W-0	R/W-0	R/W-0	R/W-0		R/W-0		R/W-x
RBPU	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP
Bit - 7	6	5	4	3	2	1	

Enables the Internal Pull-ups.

Enable if pull-ups are not connected externally

Interrupt Priority bit

1: High Priority

0: Low Priority

Enables Interrupt On state change

INTCON : Interrupt Control Register

Interrupt status bit

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
Bit - 7	6	5	4	3	2	1	

LAB 1

University of Microchip UofM MASTERs 2007

PIC18F: Development Tools

First –

a review of basic usage of Microchip development tools…

PIC18F: Development Tools MPLAB® IDE

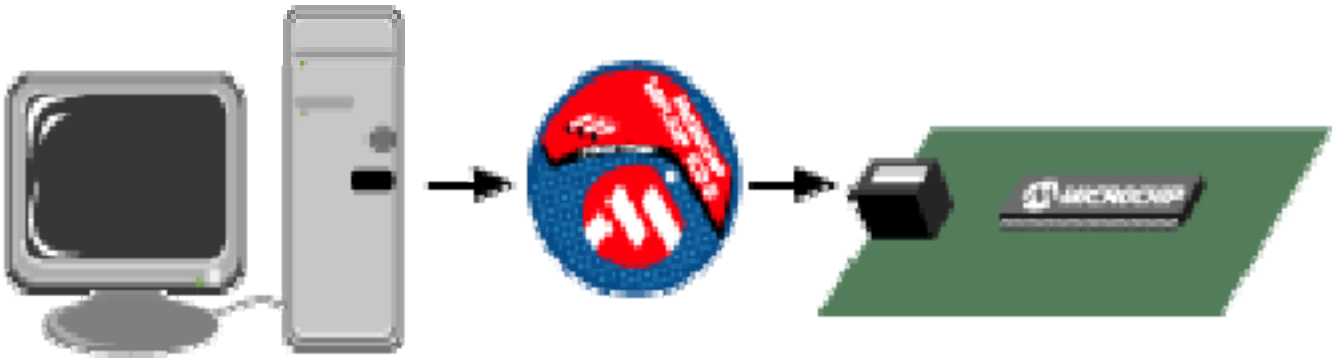
- **MPLAB IDE: Integrated Development Environment**
- **It integrates different Microchip and third party tools**
  - **Code Editor**
  - **Cross Compilers**
  - **Assemblers**
  - **Simulators, In-Circuit Debuggers, Emulators**
  - **Programmers**

MPLAB® ICD 2 (In Circuit Debugger)

- MPLAB ICD 2 is a low-cost, real-time debugger and programmer offering the following features:
  - Real-time background debugging
  - Reading/Writing memory space and EEDATA areas of target microcontroller
  - Programs Configuration bits
  - Erase of program memory space with verification

PIC18F: Development Tools MPLAB® ICD 2 (In-Circuit Debugger)

Connecting the MPLAB ICD 2



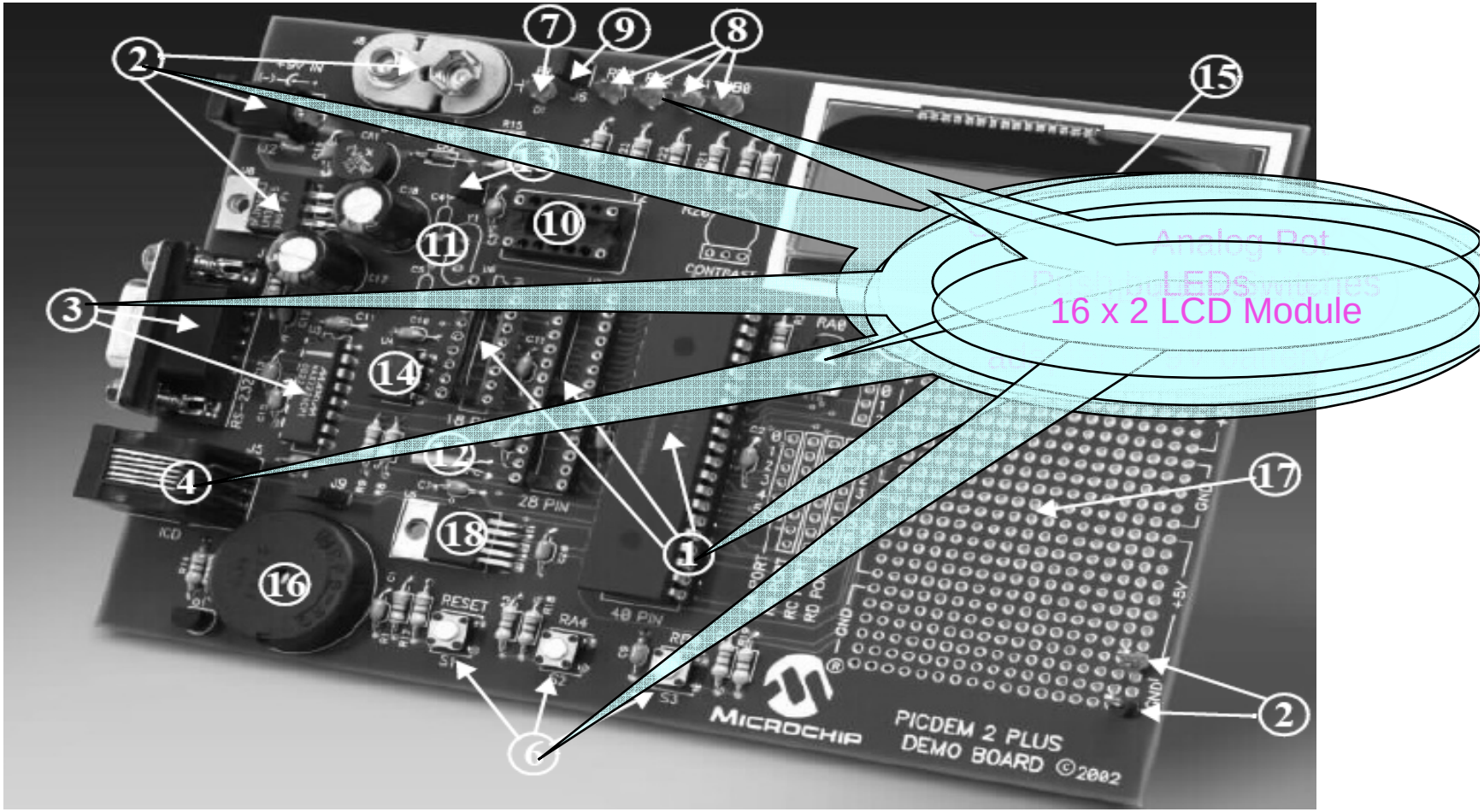
For all the hands on Labs we will be using MPLAB ICD 2 debugger along with PICDEM™ 2 Plus board

PIC18F: Development Tools
PICDEM™ 2 Plus Board

- **The PICDEM 2 Plus board**
  - **2 x 16 LCD display**
  - **Buzzer**
  - **Active RS 232 port**
  - **On board temperature Sensor**
  - **4 LEDs**
  - **2 push button switches and master reset**
  - **18/28/40 pin controller sockets**
  - **MPLAB® ICD 2 connector**
  - **Generous prototyping area**
  - **Works with a 9V battery or DC power pack**

PIC18F: Development Tools
PICDEM™ 2 Plus Board

PICDEM 2 Plus Board Hardware details



General Guidelines to carry out the Labs

- All Labs are Placed in the folder C:\Masters\11009_PRC\LabN
  - N is the Lab number associated with the lab.

General Guidelines to carry out the Labs

- Look for the key phrase:

  ### Your Code Here ###

  in every project and follow the instructions provided to carry out the lab
- Use the hand-outs provided for register definitions

Lab 1: Initialization, read and write of I/O ports

- Goals:
  - Understand PORT configuration
  - Understand Reading and Writing to a PORT or PORT pin
  - Understand significance of LAT registers
  - Understand usage of PORT interrupt on change
  - Understand External Interrupts

Lab 1: Initialization, read and write of I/O ports

- **Code for Lab 1:**
  - **Open the project Lab1.mcp**
  - **Write C source code in the file Lab1_main.c to configure:**
    - PORTB<7:1>pins as output
    - PORTB<0> as input
    - enable interrupt on change (for switch SW3 on PORTB<0>)

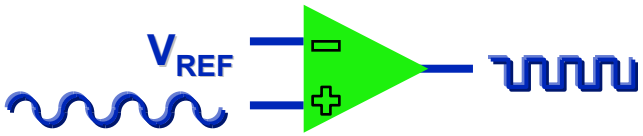
    TRISB = 0x01, INTCON2.RBPU = 1

    INTCON2.RBIP = 1, INTCON.GIE/GIEH = 1
  - **In PORTB state change ISR**
    - read PORTB<0> and increment the count if pin is zero
  - **In function main**
    - write a function to display the count on onboard LCD

Lab 1: Initialization, read and write of I/O ports

- **Expected Result:**
  - **Once you complete the lab you can expect:**
    - The LCD will display a number
    - The number will increment every time SW3 is pressed
    - Number will increment sequentially without skipping any digit

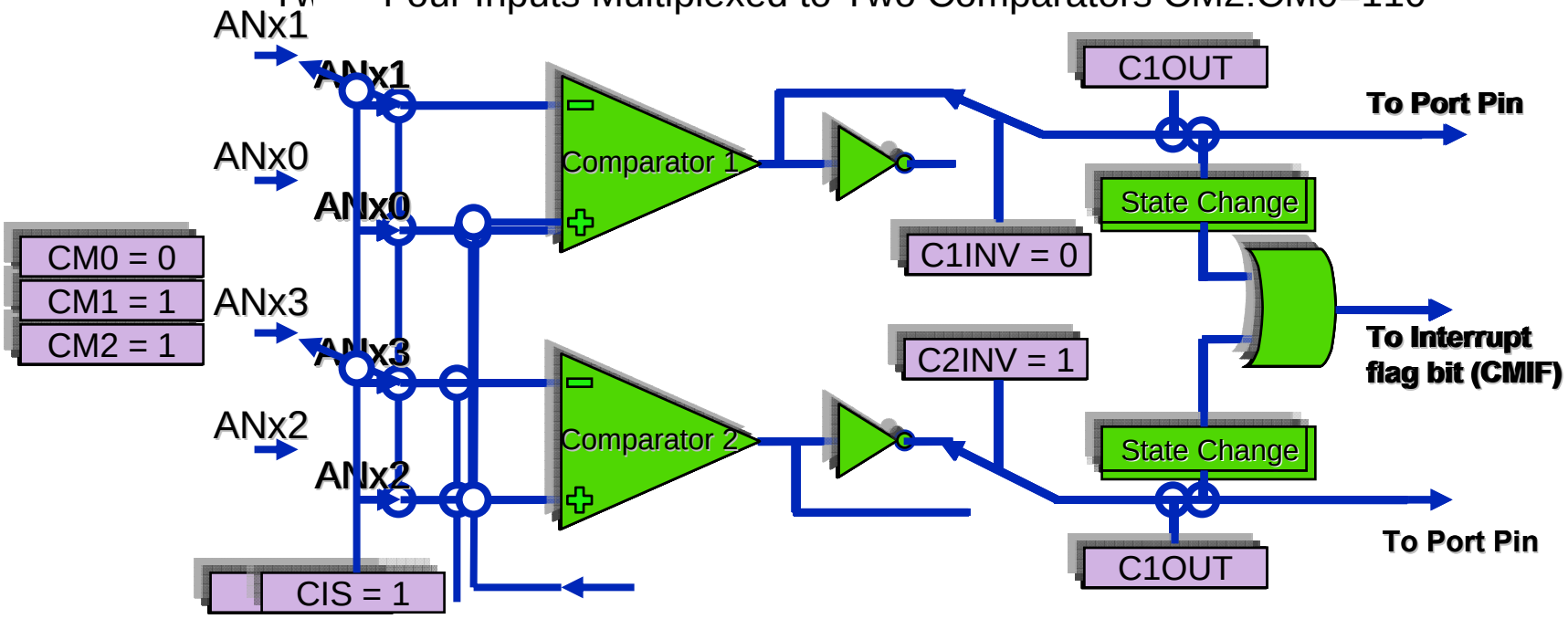
PIC18F Peripherals
Analog Comparator Module



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

PIC18F Peripherals
Analog Comparator Module Configuration

Tw Four Inputs Multiplexed to Two Comparators CM2:CM0=110



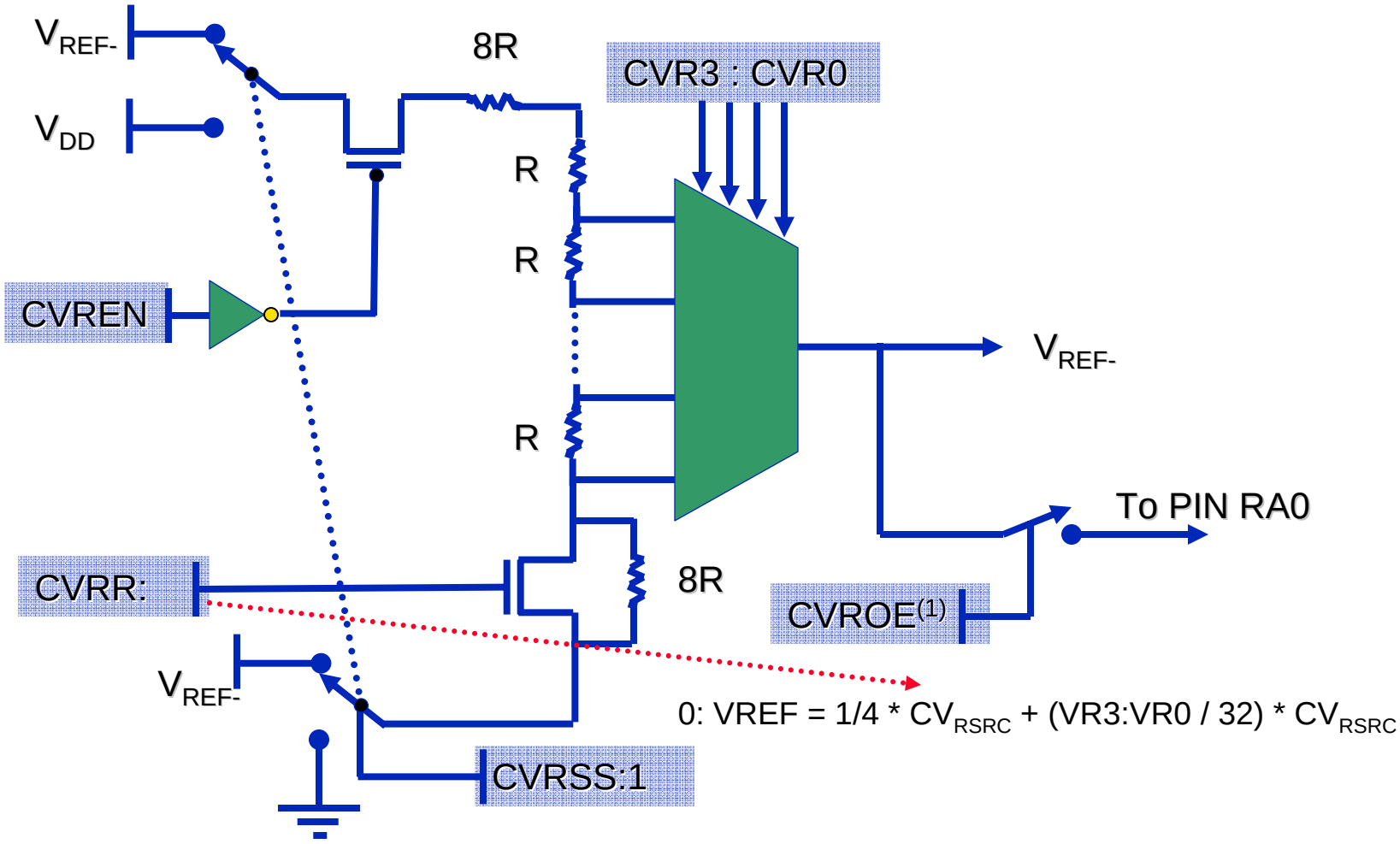
CMCON: COMPARATOR CONTROL REGISTER

R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0
bit7	6	5	4	3	2	1	0

PIC18F Peripherals
Internal VREF: Block Diagram

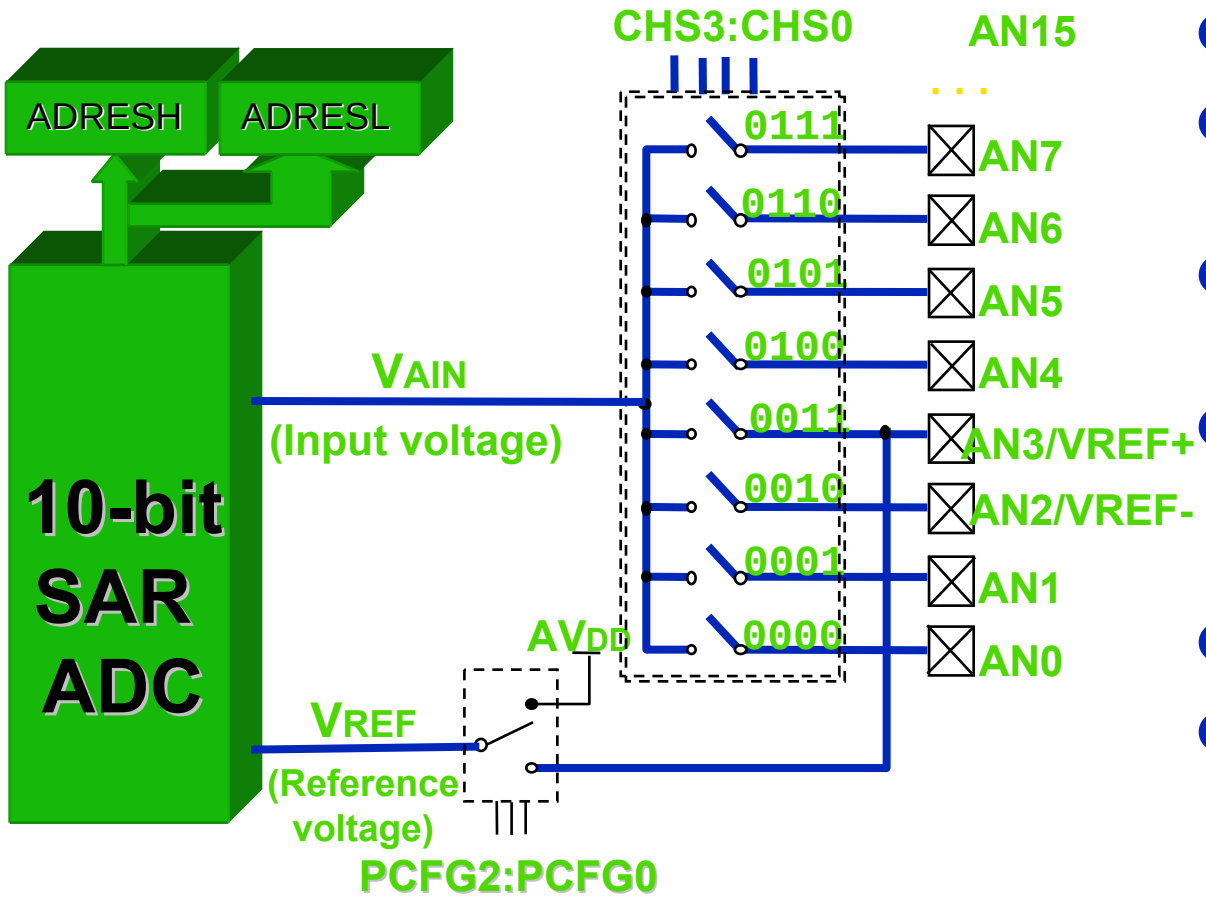
- Provides On-Chip voltage reference
- 16-tap resistor ladder network; provides programmable reference voltage
- 24 or 32 step sizes
- Internal or External Voltage source for reference voltage generation
- Can be used as a D/A converter
- VREF can be directed to an output pin

PIC18F Peripherals
Internal VREF: Configuration



Note: 1. CVROE overrides the TRISA<0> bit setting.

PIC18F Peripherals
10-bit ADC - Block Diagram



- Up to 16 ch.
- 10-bit ± 1 LSb
- Conversion during Sleep
- Conversion on Input Capture
- Internal Or External Reference
- 20 to 100 ksps*
- Easy to Configure

Note: See the device data sheet for the conversion speed

PIC18F Peripherals
ADC Module Configuration

ADCON0 : A/D CONTROL REGISTER 0

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CHS3	CHS3	CHS3	CHS3	GO/DONE	ADON
Bit7	6	5	4	3	2	1	0

This Configuration register will help to select the required channel and controls the conversion

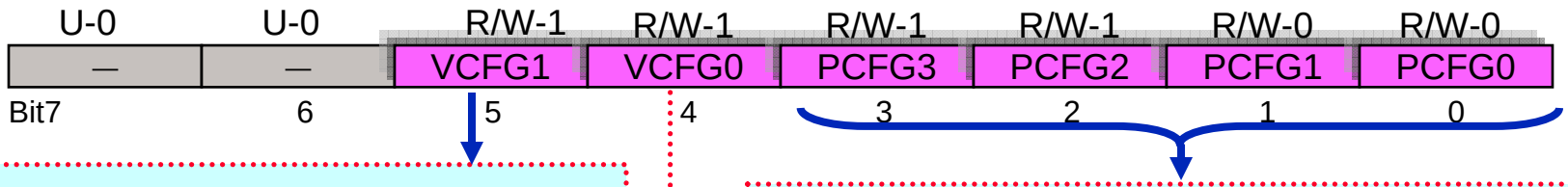
3	2	1	0	Selected
0	0	0	0	Ch
0	0	0	1	
0	0	1	0	CH – 2
0	0	1	1	CH – 3
0	1	0	0	CH – 4
;				;
;				;
1	1	1	1	CH – 15

Start Conversion

Enable ADC

PIC18F Peripherals
ADC Module Configuration

ADCON1 : A/D CONTROL REGISTER 1



Selects the reference voltage for V_{REF-} Input of ADC
0: V_{REF-} is AV_{SS}
1: V_{REF-} is AN2

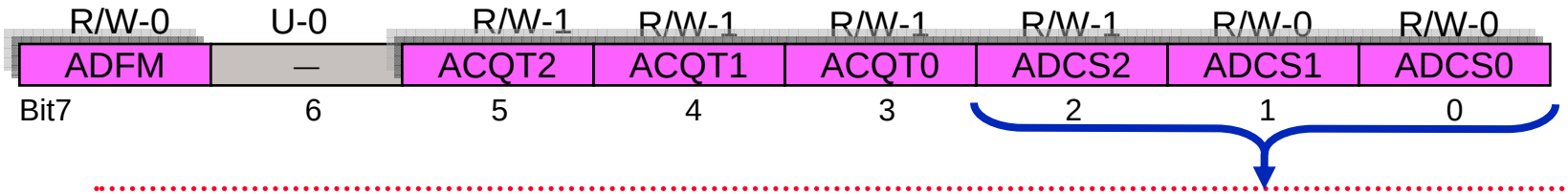
Selects the reference voltage for VREF + Input of ADC
0: VREF+ is AVDD
1: VREF+ is AN3

Defines number of Analog capable Port pins as analog inputs
0000 – No Analog Channels
0001 – 1 Analog Channel (AN0)
“
“
“
0111 – 7 Analog inputs (AN0 to AN6)
“
1111 – 16 Analog inputs (AN0 to AN15)

V reference selection helps to get the Full range of ADC values when input Sensors use voltage sources of different level.

PIC18F Peripherals
ADC Module Configuration

ADCON2 : A/D CONTROL REGISTER 2

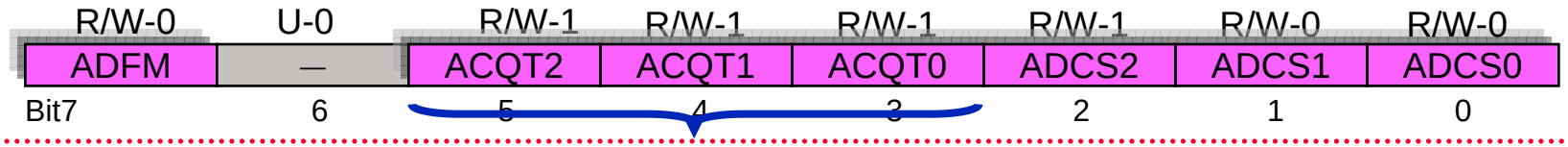


The SAR requires 11 clocks to for 10 bit conversion and duration of each clock should be greater than 2 µS(approx.)
These bits will help to achieve this from the range of input clock frequencies

ADCS2:ADCS0 – 000 → T_{AD} = 2 T_{OSC}
ADCS2:ADCS0 – 100 → T_{AD} = 4 T_{OSC}
ADCS2:ADCS0 – 001 → T_{AD} = 8 T_{OSC}
ADCS2:ADCS0 – 101 → T_{AD} = 16 T_{OSC}
ADCS2:ADCS0 – 010 → T_{AD} = 32 T_{OSC}
ADCS2:ADCS0 – 110 → T_{AD} = 64 T_{OSC}
ADCS2:ADCS0 – x11 → T_{AD} = RC**
** T_{AD} varies from 4 to 6 ms depending on V_{DD}

PIC18F Peripherals
ADC Module Configuration

ADCON2 : A/D CONTROL REGISTER 2



The Sample and hold CAPACITOR must be charged to the analog input voltage level in order to get the good accuracy in conversion

Following are the delays involved in Charging the capacitor

- Amplifier Settling Time (TAMP = 5 µS approx)
- Holding Capacitor Charging Time (TC = 9.61 µS approx approx.)
- Temperature Coefficient (TCOFF = 1.25 µS approx)

So the total Acquisition tine required is TACQ = 12.86 µS

ACQT2:ACQT0 will provide this delay before starting the conversion (After setting GO bit)

Acquisition delay = $(2^n * T_{AD})$**

Where n = ACQT2:ACQT0

** Refer the device data sheet for accurate delay cycles

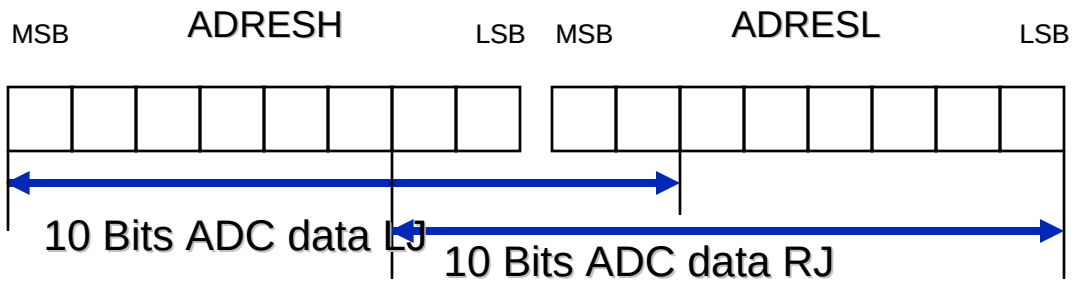
PIC18F Peripherals
ADC Module Configuration

ADCON2 : A/D CONTROL REGISTER 2

R/W-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-0	R/W-0
ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0
Bit7	6	5	4	3	2	1	0

This bit will allows the ADC result to be stored in Left/Right aligned in ADRES register

If ADFM = 0: Result is Left Justified; this will help to get the Low resolution data by reading only ADRESH register
If ADFM = 1: Result is Right Justified; this will help to get the High resolution data by reading only ADRESL register



Lab 2: Initialization and ADC conversion

- Goals:
  - Understand ADC configuration
  - Understand Reading ADC channels
  - Understand selecting ADC channels
  - Understand switching ADC channels
  - Understand scaling ADC value

Lab 2: Initialization and ADC conversion

- **Code for Lab 2:**
  - **Open the project Lab2.mcp**
  - **Write C source code in the file Lab2_main.c to configure**
    - PORTA<0> pin as analog input
    - Conversion clock = FOSC/8
    - Acquisition time = $12T_{AD}$
    - Result format = Right aligned
    - Select internal reference voltage for the ADC

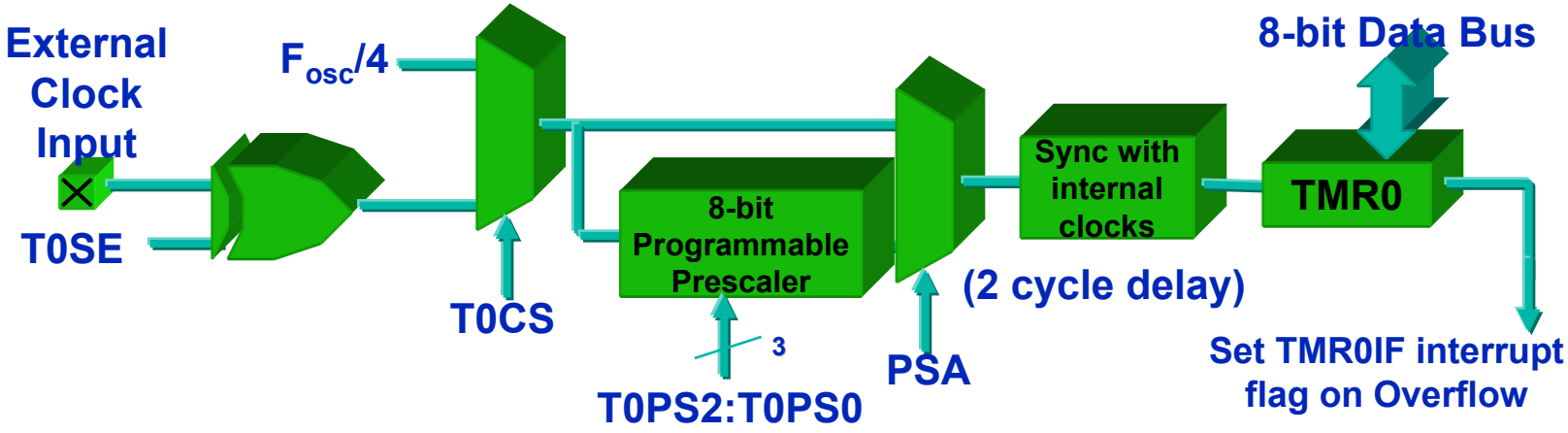
    ADCON0 = 0x01, ADCON1 = 0x0E, ADCON2 = 0x29
  - **Select ADC channel 0 for conversion**
  - **Start the ADC conversion and wait for conversion complete**
  - **Read the ADC result register and display**

Lab 2: Initialization and ADC conversion

- **Expected Result:**
  - **Once you complete the lab you can expect:**
    - LCD will displaying voltage at potentiometer
    - Minimum voltage should be 0V (Pot at minimum position)
    - Maximum voltage should be 5V (Pot at maximum position)

PIC18F 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

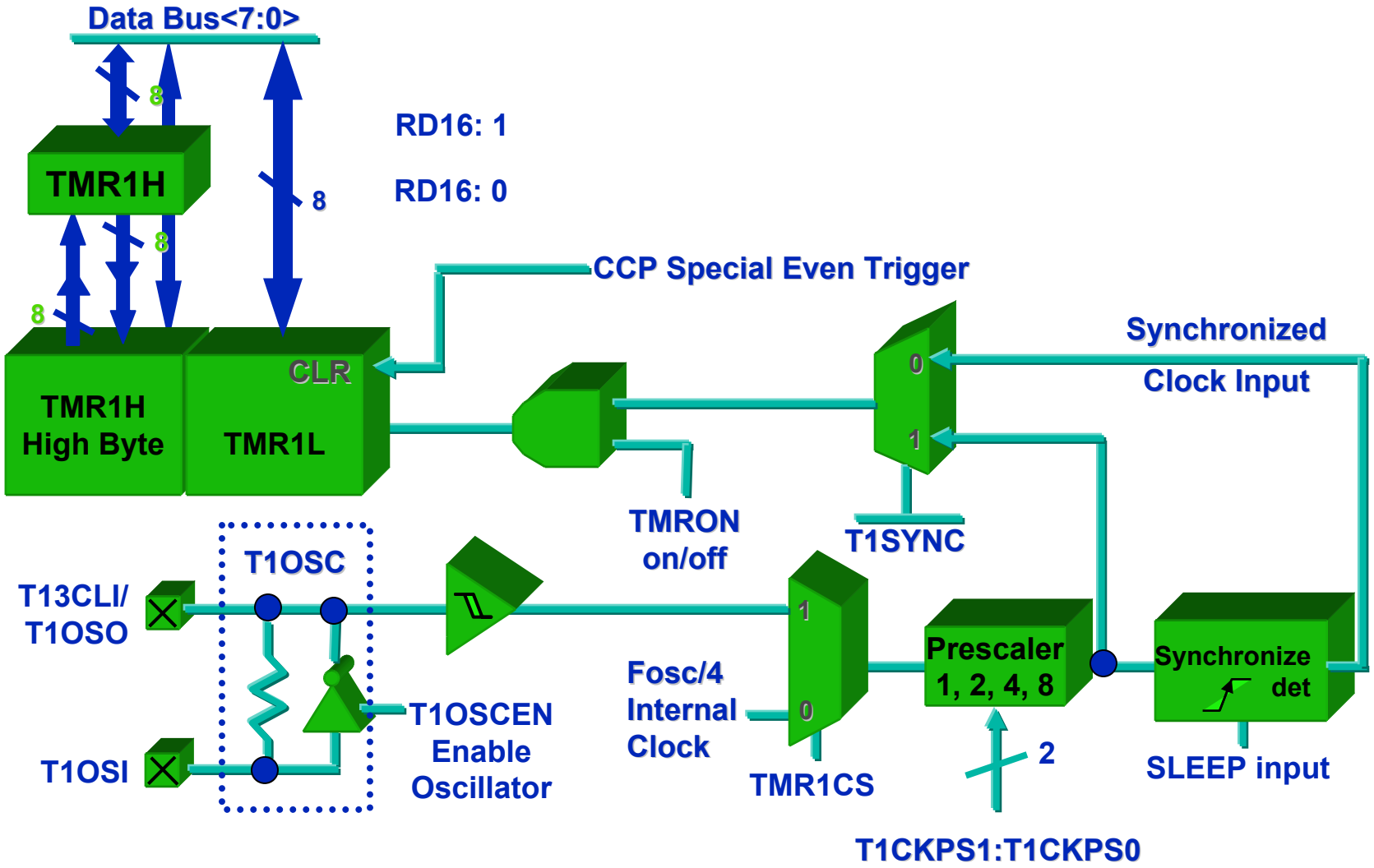


University of Microchip UofM MASTERs 2007

PIC18F Peripherals
Timer1 and Timer3

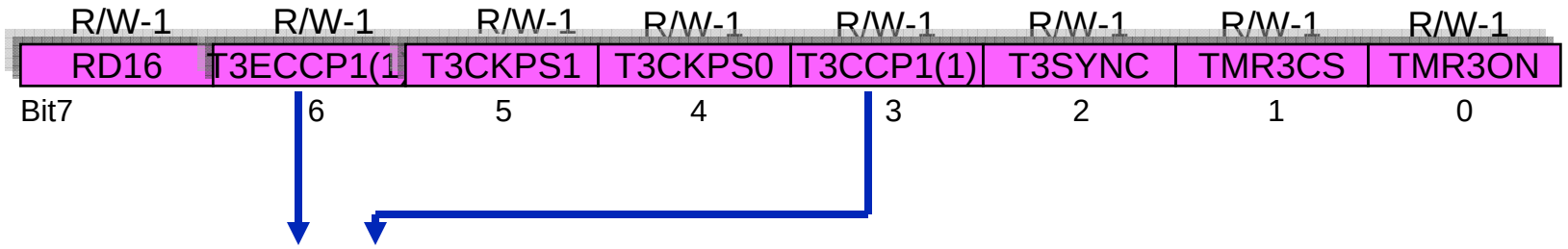
- 16-bit Timer / Counter
- Timer, Synchronous Counter or Asynchronous Counter
- Built in oscillator allows operation from external crystal
- Consists of two readable and writeable 8-bit registers
- ÷1, ÷2, ÷4, or ÷8 Prescaler
- Interrupt on overflow from FFFFh to 0000h

PIC18F Peripherals
Timer1 and Timer3 (Continued)



PIC18F Peripherals
Timer 3 Module Configuration

T3CON: TIMER0 CONTROL REGISTER



1 x: Timer3 is the capture/compare clock source for both CCP and ECCP modules

0 1: Timer3 is the capture/compare clock source for ECCP1; Timer1 is the capture/compare clock source for CCP1

0 0: Timer1 is the capture/compare clock source for both CCP and ECCP modules

CCP and ECPP Clock Source selection bits

Note: 1 These bits are available only in PIC18F4X80 devices only

Lab 3: Working with Counters

- Goals:
  - Understand TIMER configuration in counter mode
  - Understand Frequency measurement techniques

Lab 3: Working with Counters

- **Code for Lab 3:**
  - **Open the project Lab3.mcp**
  - **Write C source code in the file Lab3_main.c to configure:**
    - Timer1 for 16 bit synchronous counter mode of operation
    - Select Timer1 Prescaler 1:1
    - Preload the Timer1 counter with 8000h (d32768) to provide 1 second time interval
    - Enable Timer1 overflow interrupt

    T1CON = 0x0B, TMR1H = 0x80, TMR1L = 00
    - Write Timer1 ISR to implement a software RTCC

Lab 3: Working with Counters

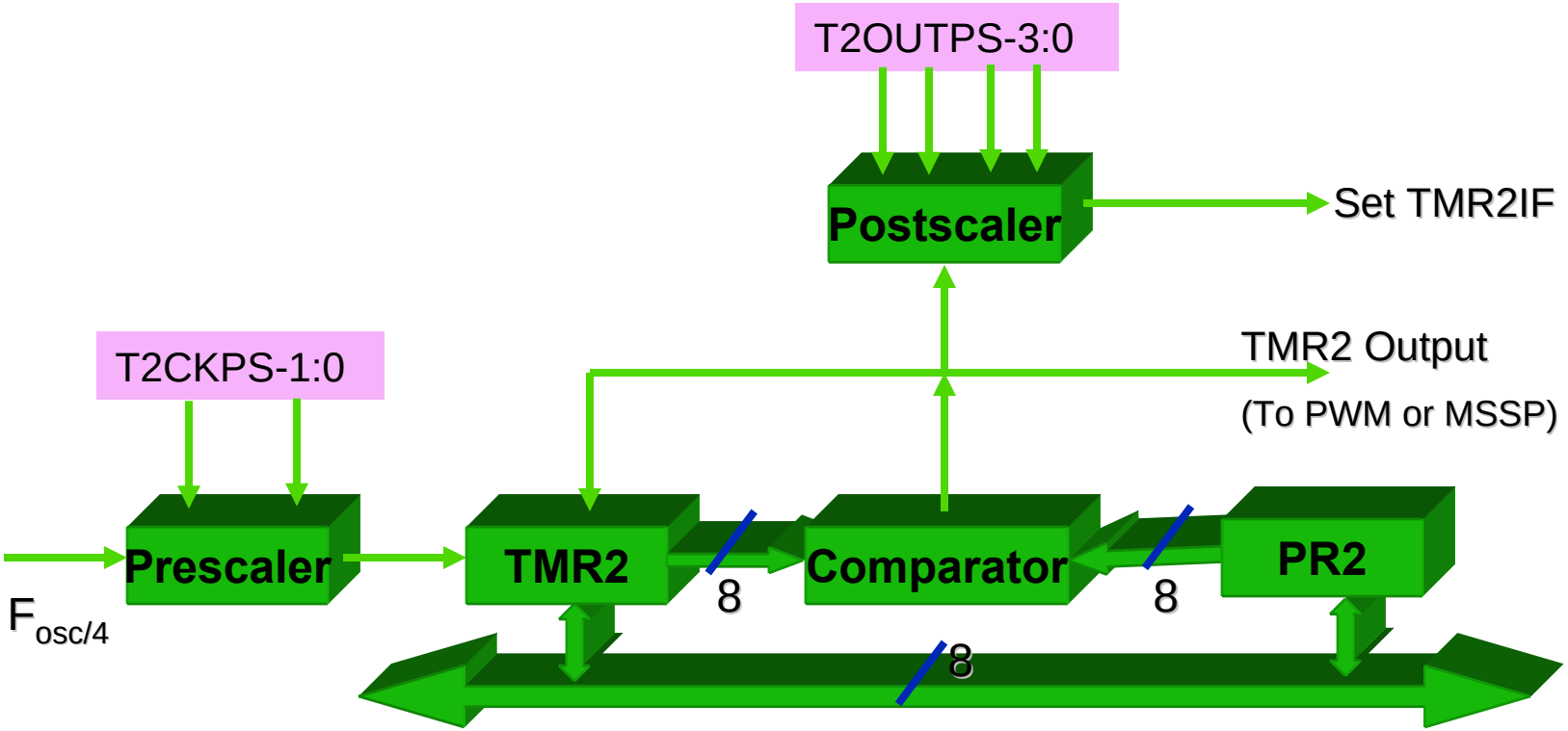
- **Expected Result:**
  - **Once you complete the lab you can expect:**
    - LCD displaying Time in HH:MM:SS format

UNIVERSITY OF MICROCHIP UOFM MASTERS 2007

PIC18F Peripherals Timer2

- 8-bit Timer with prescaler and postscaler
- Used as time base for PWM mode of CCP module
- TMR2 is readable & writable
- TMR2 increments until it matches period PR2, then resets to 00h
- TMR2 match with PR2 generates an interrupt through postscaler
- Used as baud clock for MSSP (SPI)

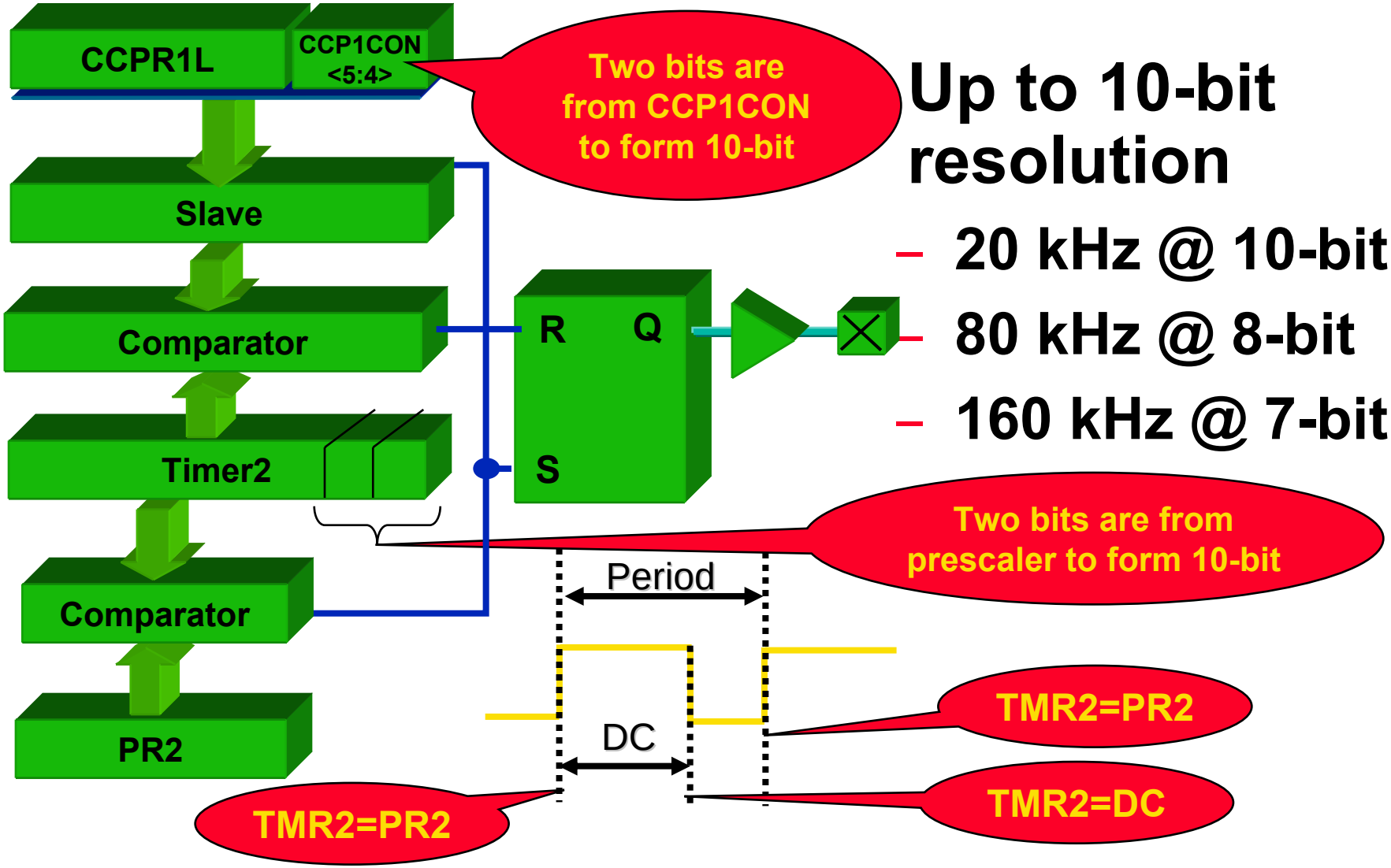
PIC18F Peripherals
Timer 2 Module Configuration



T2CON: TIMER0 CONTROL REGISTER

U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0
Bit7	6	5	4	3	2	1	0

PIC18F Peripherals
CCP Module: PWM Mode



- Up to 10-bit resolution
  - 20 kHz @ 10-bit
  - 80 kHz @ 8-bit
  - 160 kHz @ 7-bit

Lab 4: Working with PWM

- **Goals:**
  - Understand TIMER configuration for PWM
  - Understand TIMER2 PWM mode of operation
  - Learn how to calculate period value from system frequency

Lab 4: Working with PWM

- **Code for Lab 4:**
  - **Open the project Lab4.mcp**
  - **Write C source code in the file Lab4_main.c to configure**
    - Timer2 for Single output PWM mode of operation
    - Set Prescaler to 1:4
    - Set default frequency to 4.9 KHz
    - Set default duty cycle to 50%

    CCP1CON = 0Ch, T2CON = 05h, PR2 = 80h, CCP1CON<5:4> = 3
  - **Toggle PWM on/off when SW2 or SW3 pressed**
  - **Modify the period proportional to the Analog Input**

Lab 4: Working with PWM

- **Code for Lab 4:**
  - **Use the following formula to calculate the PWM frequency (1/Period)**

$$\text{PWM Period} = (\text{PR2}) + 1] \cdot 4 \cdot \text{Tosc} \cdot (\text{TMR2 Prescale Value})$$

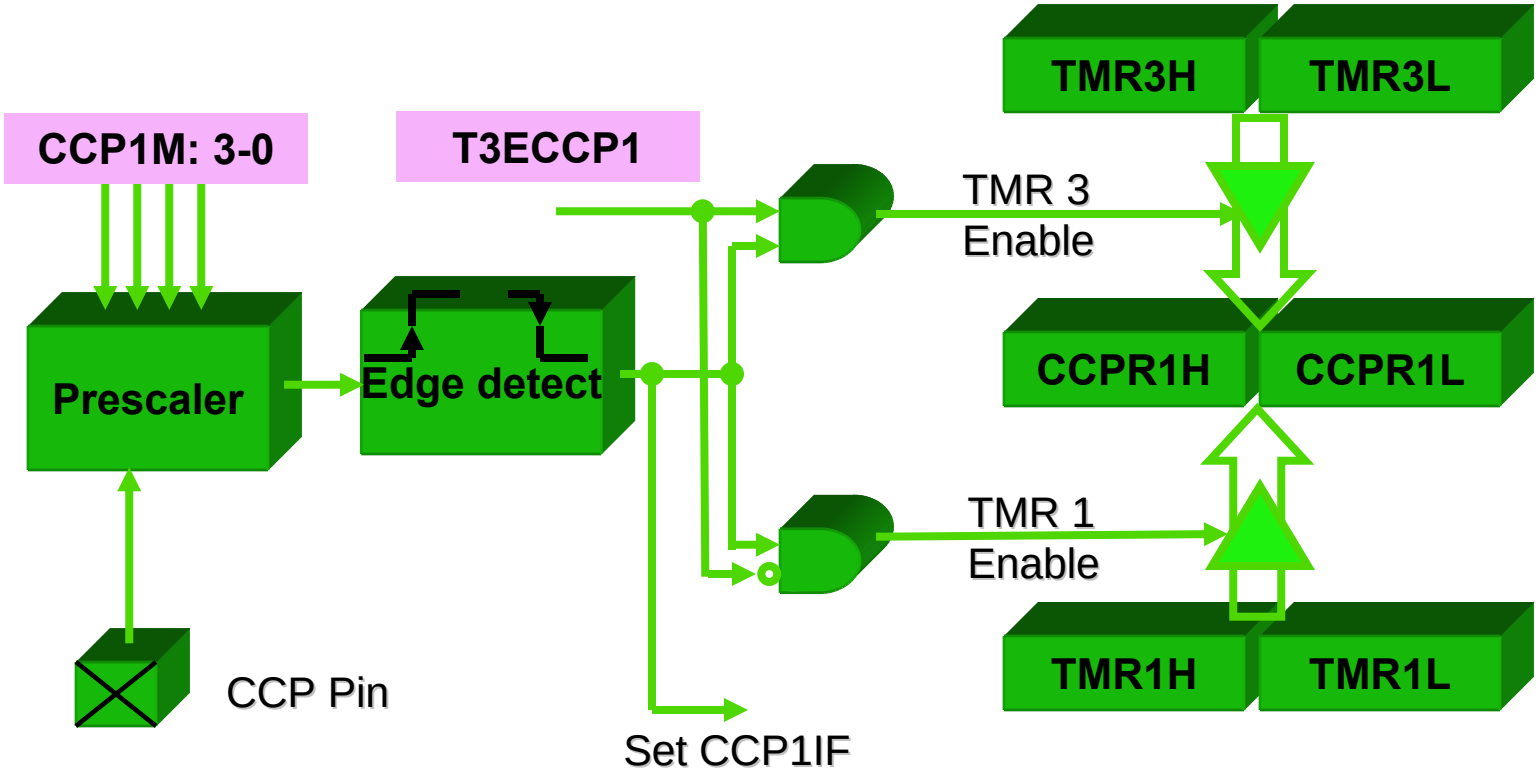
Lab 4: Working with PWM

- **Expected Result:**
  - **Once you complete the lab you can expect:**
    - Sound on the buzzer
      - The PWM will start or stop by pressing switches SW2 or SW3
      - The frequency of sound will vary when you turn the pot

PIC18F Peripherals
CCP Module: Input Capture Mode

- Captures 16-bit Timer1/Timer3 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

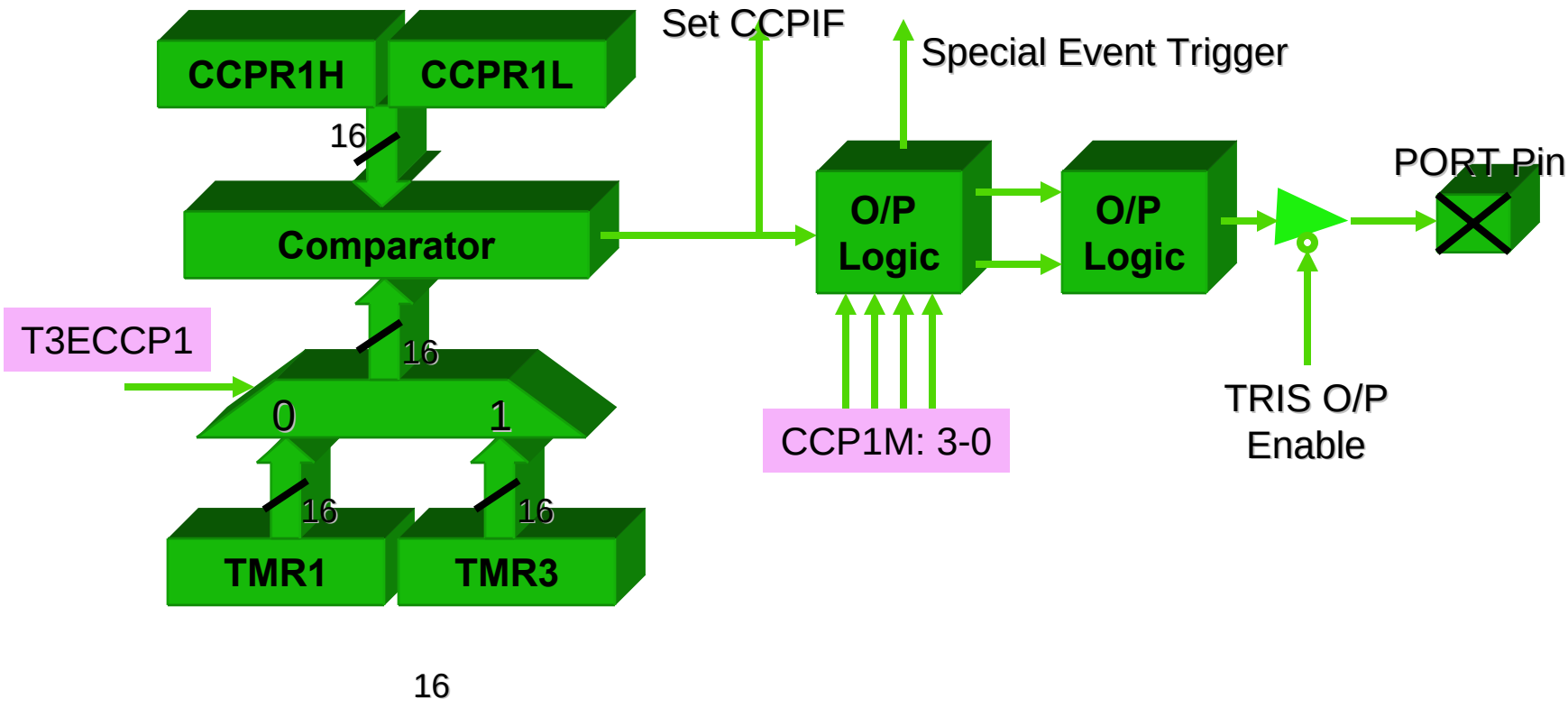
PIC18F Peripherals
CCP Module: Input Capture Mode



PIC18F Peripherals
CCP Module: Output Compare Mode

- 16-bit CCPRn register value is compared to TMR1/3, and on match the CCPn pin is
  - Driven High/Low
  - Toggled
  - Unchanged
- Compare match generates interrupt
- Compare match can trigger a Special event
- Can start A/D conversion

PIC18F Peripherals
CCP Module: Output Compare Mode



Lab 5: Working with CCP

- Goals:
  - Understand TIMER configuration for input capture
  - Understand TIMER1 Capture mode of operation
  - Learn how to calculate period from captured counts

Lab 5: Working with CCP

- **Code for Lab 5:**
  - **Open the project Lab5.mcp**
  - **Write C source code in the file Lab5_main.c to configure:**
    - Timer3 for CCP clock source input
    - Set Timer3 to run on internal clock source (FOSC/4)
    - Configure and Enable capture interrupt for both rising and falling edges

    T3CON = 0x49, CCP2CON = 0x05 (For rising edge)

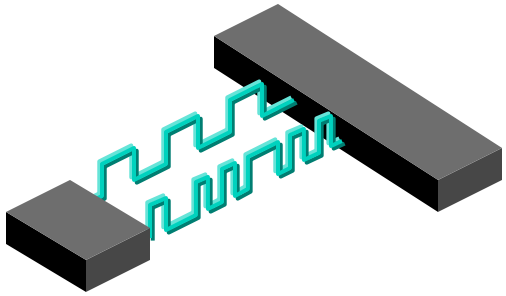
    CCP2CON = 0x04 (For Falling edge)
  - **Connect PWM port pin to Capture port pin**
  - **Write Capture ISR to**
    - Read Capture register and PORT pin status
    - Calculate Period and Duty cycle

Lab 5: Working with CCP

- **Expected Result:**
  - **Once you complete the lab you can expect:**
    - LCD displaying Period/Frequency and Duty Cycle
      - The PWM will start or stop by pressing switches SW2 and SW3
      - The period will vary if you turn the pot

PIC18F Peripherals
Master Synchronous Serial Port

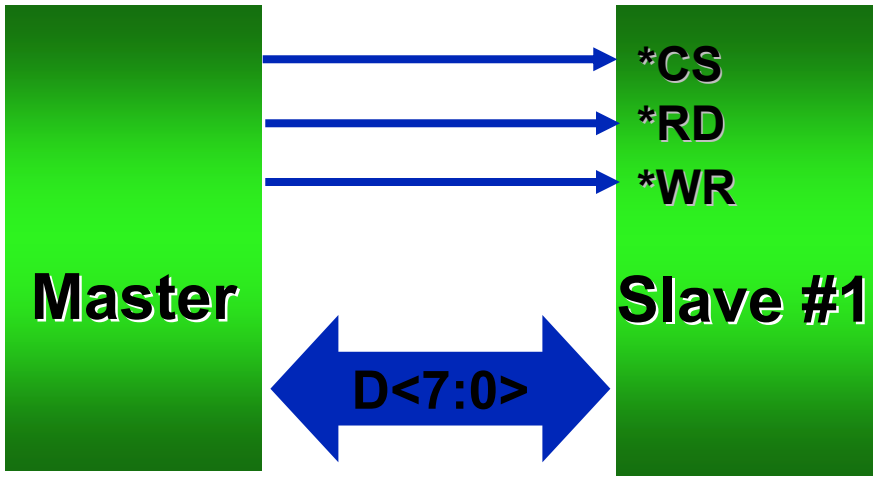
- **Operates in either SPI or I²C™ mode**
- **SPI Mode**
  - **Programmable baud rate**
  - **Maximum baud rates (@ 40MHz)**
    - Master: 10 Mbps
    - Slave: 2.5 Mbps Single Byte Tx



- **I²C Mode**
  - **Supports standard (100 kHz), fast (400 kHz), and Microchip's 1 MHz I²C standards**
  - **Hardware Master/Slave implementation**
  - **Glitch filter and Slew rate control on SCL and SDA pins**

PIC18F Peripherals
Parallel Slave Port

- 8-bit Multi-MCU bus
  - One master - many slaves
  - Master can Read/Write from slave(s)
- Asynchronous communication
- High operating speed



Lab 6: Working with I²C™

- Goals:
  - Understand MSSP configuration
  - Understand I²C Master mode of operation
  - Understand steps required to communicate with I²C slave device

Lab 6: Working with I²C™

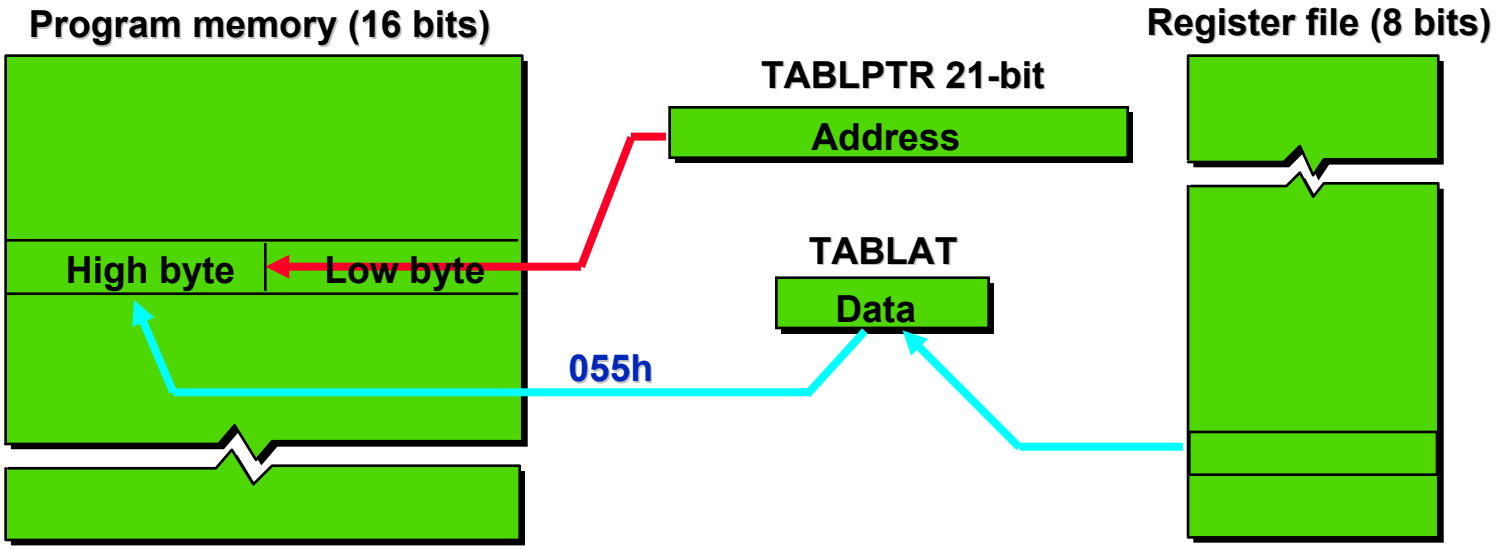
- **Code for Lab 6:**
  - **Open the project Lab6.mcp**
  - **Write C source code in the file Lab6_main.c to configure:**
    - CLK, and SDA as inputs TRISC (TRISC<3:4> = 11).
    - Configure as Master.
      - SSPCON2 = 0x28;
    - Clock for master mode = (FOSC /(4 x (SSPADD + 1))
      - SSPADD = 0x63; (Note: Choose any desired baud rate)
  - **Send Temperature read request command to Temperature sensor using I2C_read() function**
  - **Convert received Temperature data to BCD (use Bin2BCD() function)**
  - **Display the Received temperature on on-board LCD using LCD write function**

Lab 6: Working with I²C™

- **Expected Result:**
  - **Once you complete the lab you can expect:**
    - LCD displaying Temperature data

PIC18F Table Operations

```
MOVLW   055h
MOVF    TABLAT
TBLWT*
```



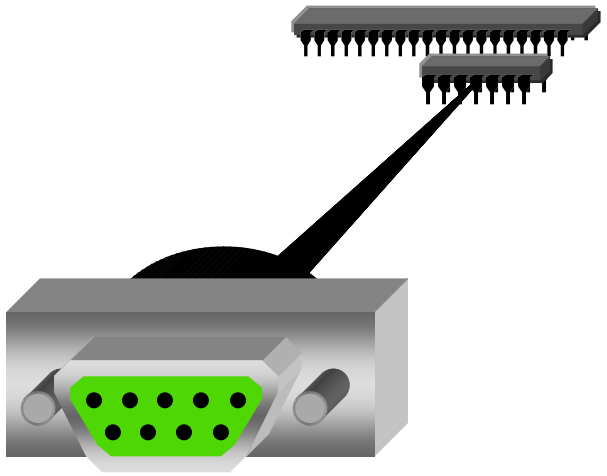
- TBLPTR: Register holds program memory address
- TABLAT: Register holds data to be written or read
- Read transfers data from program memory to register file
- Write transfer data from register file to program memory
- Each transfer takes 4 cycles

PIC18F Table Pointer Operations

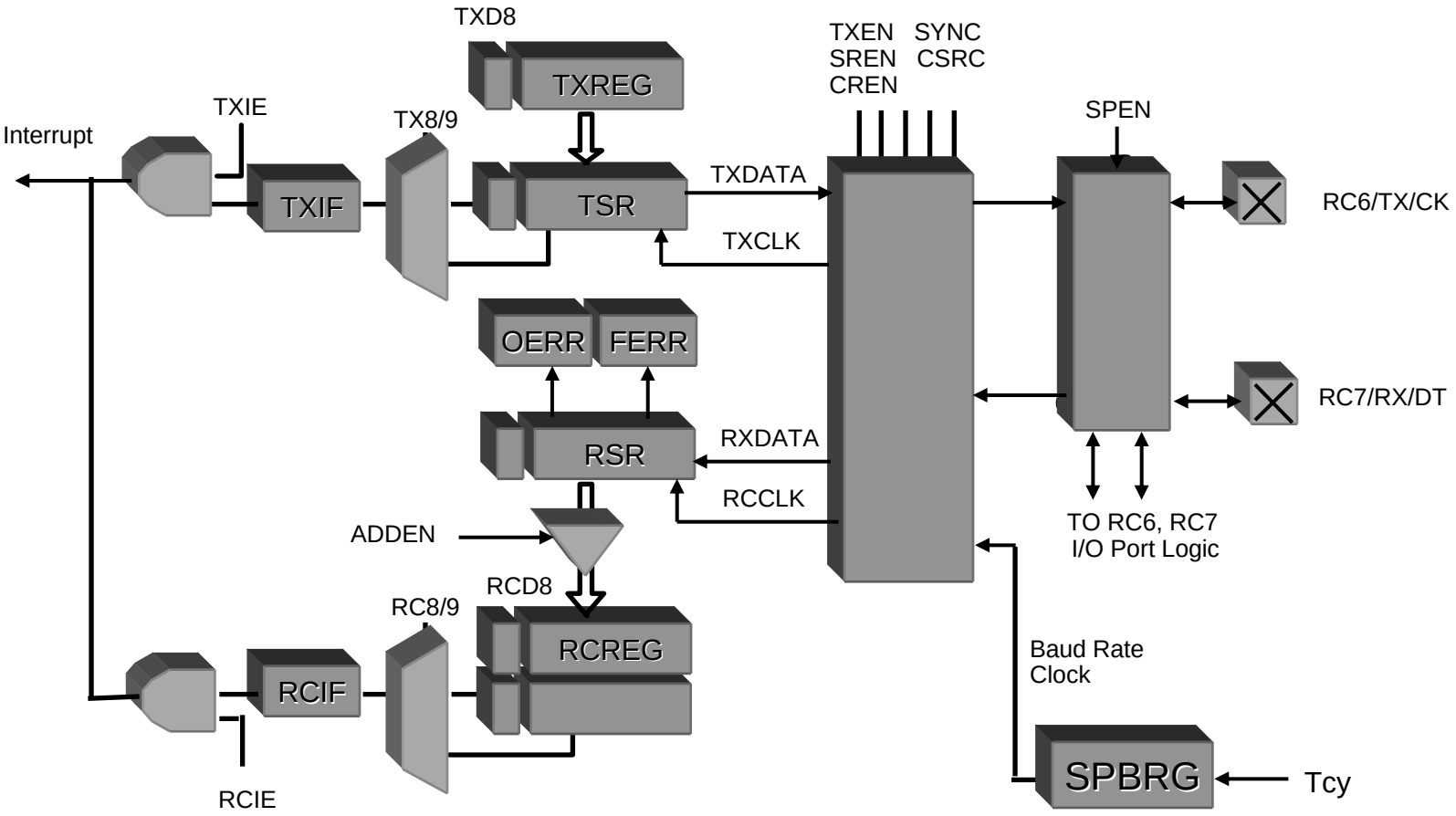
- TBLRD * — no change to table pointer
- TBLRD *+ — auto post-increment of table pointer
- TBLRD *- — auto post-decrement of table pointer
- TBLRD +* — auto pre-increment of table pointer

PIC18F Peripherals
Serial Communications Interface (SCI/USART)

- Full-duplex asynchronous -or- half-duplex synchronous
- 8- or 9-bit data
- Double-buffered transmit and receive buffers
- Separate transmit and receive interrupts
- LSB first transmission and reception
- Dedicated 16-bit baud rate generator
- Max baud rates @ 40 MHz
  - Synchronous: 10 Mbaud
  - Asynchronous: 625 Kbaud/2.5 Mbaud
- 9-bit addressable mode
- Auto wake-up on character reception
- Auto-Baud calibration
- 12-bit Break character transmission



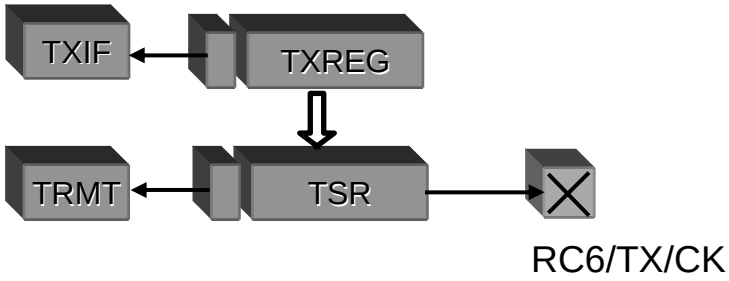
PIC18F Peripherals
Serial Communications Interface
USART Block Diagram



PIC18F Peripherals
Serial Communications Interface
TXIF and TRMT Operation

- TXREG empty, will set TXIF
- Loading TXREG resets TXIF

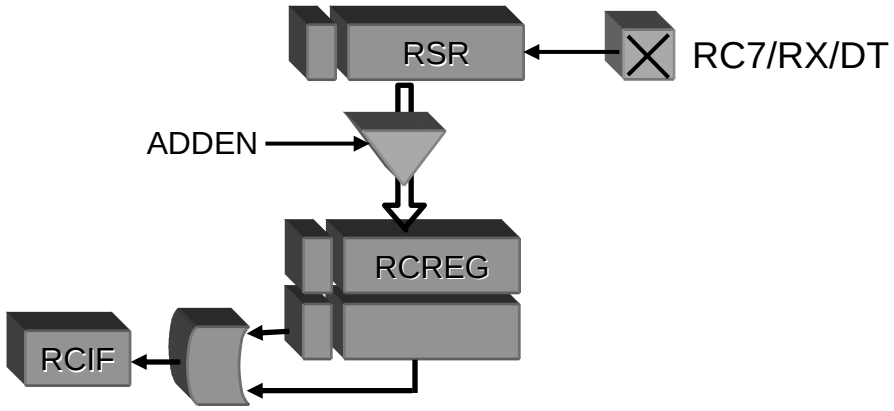
- TSR empty will set TRMT
- Loading TSR resets TRMT



- If TXREG is loaded and TRMT is set, then data is immediately loaded to TSR. Serial data shifting starts and TXIF will be set

PIC18F Peripherals
Serial Communications Interface
RCIF Operation

- RSR receives data with valid start/stop
- Data loaded into RCREG FIFO and RCIF = 1
- If 2nd byte is received before the 1st has been serviced, then new data will be placed in the 2nd location on the FIFO
- When servicing the receive interrupt, after reading the 1st byte, if another byte is still in the FIFO, then a second RCIF interrupt is generated



2 deep FIFO

University of Microchip UofM MASTERs 2007

Lab 7: Working with USART

- Goals:
  - Initializing SFRs for USART peripheral setup
  - Understand data Transmission and Reception using USART
  - Writing code to handle interrupt events
  - Understand the usage of Table read/write instructions

Lab 7: Working with USART

- **Code for Lab 7:**
  - **Open the project Lab_7.mcp**
  - **Write C source code in the file Lab7_main.c to configure USART**
    - 19,200 baud async. @ 10 MHz
    - Transmit enabled, Continuous Reception
    - Address Detection disabled
    - Configure as a high-priority interrupt source
  - **Add I²C™ file to the project (to read temperature data)**
  - **Add code in 'HighISR' to:**
    - Check for USART receive interrupt
    - Echo data back to PC terminal
    - When Key 'R' is received transmit the Temperature data
    - When Key 'S' is received transmit the Constant string
  - **Verify results with HyperTerminal**

Lab 7: Working with USART

- **Code for Lab 7:**
  - **Here is the formula to calculate the baud rate**

SYNC	BRGH = 0 (Low Speed)	BRGH = 1 (High Speed)
0	(Asynchronous) Baud Rate = Fosc/(64(X+1))	Baud Rate = Fosc/(16(X+1))
1	(Synchronous) Baud Rate = Fosc/(4(X+1))	NA

X = value in SPBRG (0 to 255)

  - **Click on the following link to open the Hyper terminal**
    - Link to Hyper Terminal

Lab 7: Working with USART

- **Expected Result:**
  - **Once you complete the lab you can expect**
    - You can see echoed data on the Hyper terminal for every key press
    - If you press Key R you will receive Temperature data
    - If you press Key S you will receive pre-stored constant string

PIC18F: Architecture Advantage

- **C Compiler Optimized**
  - **Architecture**
    - Linear Program Memory addressing to 2 MB
    - Linear Data Memory addressing to 4 KB
    - 3 Data Pointers with 5 addressing modes
  - **Extended Instruction Set**
    - Software Stack Pointer manipulation
    - Dynamic allocation and de-allocation of software stack
    - Manipulation of variables located in a software stack
    - Direct Function pointer invocation

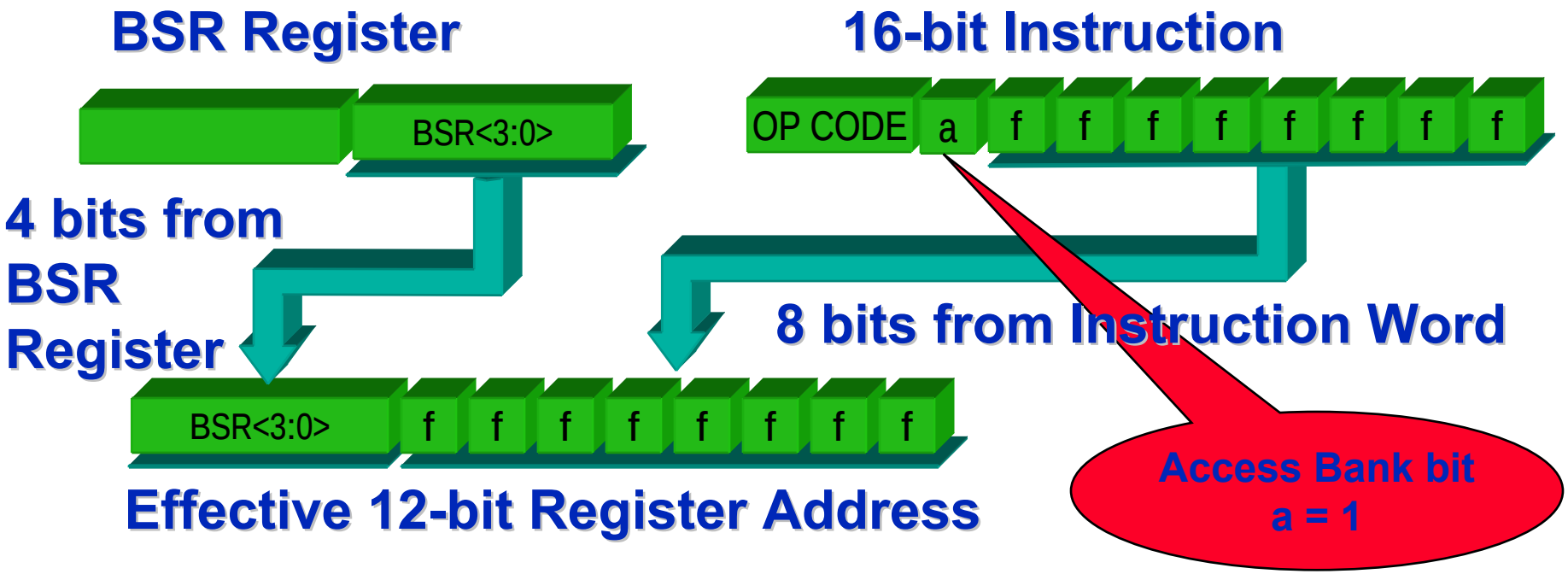
PIC18F: Extended Mode

- Enable/Disable by *XINST* configuration bit
- Indexed Literal Offset Addressing mode and…

Instruction		Description
ADDFSR	f, k	Add literal to FSR
ADDULNK	k	Add literal to FSR2 and return
CALLW		Call subroutine using WREG
MOVSF	Zs, fd	Move zs (source) to 1st word fd (destination) 2nd word
MOVSS	Zs, Zd	Move zs (source) to 1st word zd (destination) 2nd word
PUSHL	k	Store literal at FSR2, decrement FSR2
SUBFSR	f, k	Subtract literal from FSR
SUBULNK	k	Subtract literal from FSR2 and return

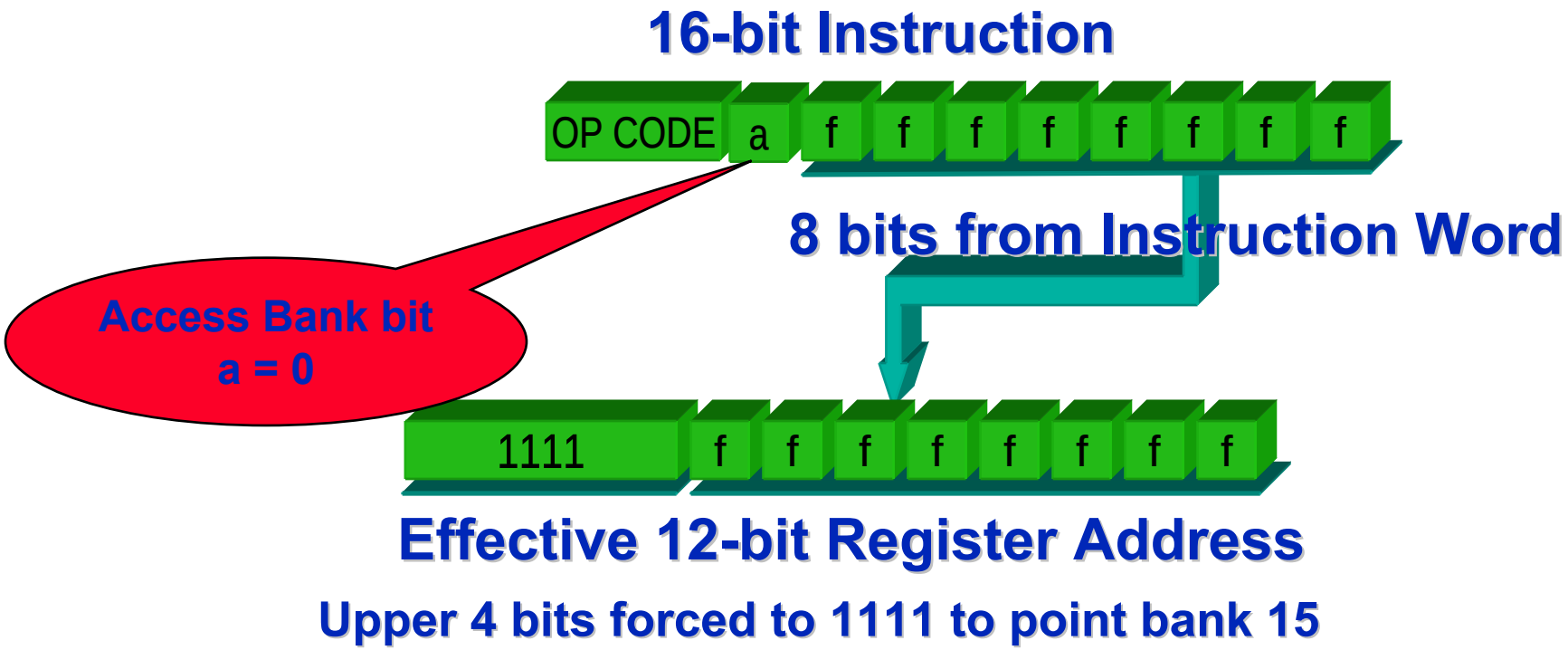
PIC18F Architecture
Accessing Data Memory (Extended Mode *XINST = 1*)

- Instruction format example when a bit (access bank bit) = 1



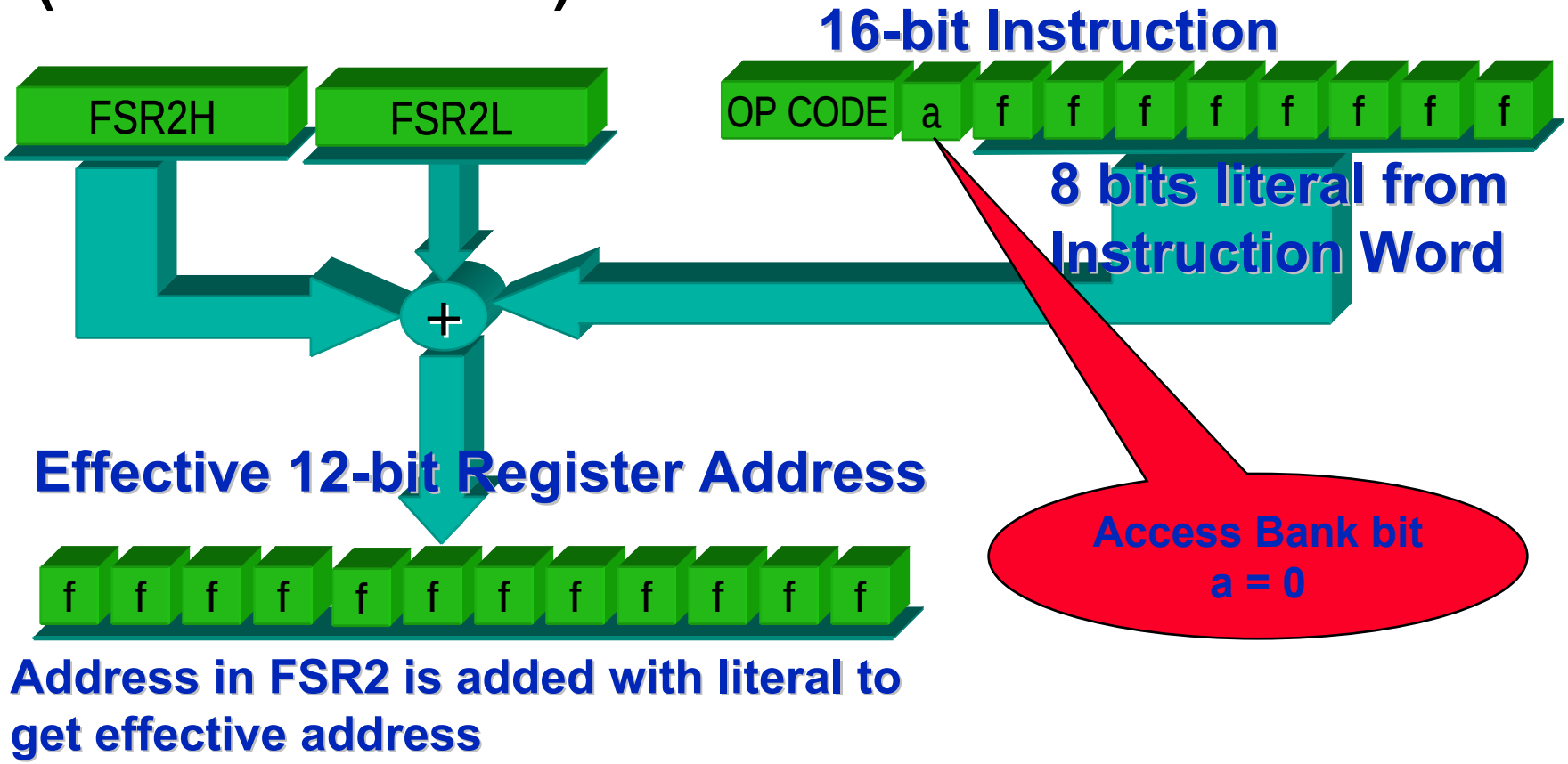
PIC18F Architecture
Accessing Data Memory (Extended Mode *XINST = 1*)

- **Instruction format example when *a* bit (access bank bit) = 0 and f ≥ 60h**



PIC18F Architecture Indexed Literal Offset Addressing (Extended Mode *XINST = 1*)

- **Instruction format example when *a* bit (access bank bit) = 0 and f ≤ 5Fh**



Lab 8: Extended Architecture Advantages demo

- **Goals:**
  - **PART-I: Using the MPLAB® IDE**
    - Setting options for Standard mode and Extended mode
  - **PART-II: Understand and evaluate the advantages of Extended architecture**

Lab 8: Extended Architecture Advantages demo

- **Instructions:**
  - **Open Lab8_Extended.mcp**
    - Disable Extended mode option in Configuration
    - Disable Extended mode in MCC18 compiler
    - Include standard mode linker file ’18f4520i_e.lnk’
    - Build the Project
  - **Open Lab8_Standard.mcp**
    - Enable Extended mode option bit in Configuration
    - Enable Extended mode bit in MCC18 compiler
    - Include extended mode linker file ‘18f4520i.lnk’
    - Build the Project
  - **Compare code size and execution time**

Lab 8: Extended Architecture Advantages demo

- **Matrices:**

	Standard mode	Extended mode
Code size		
Execution time		

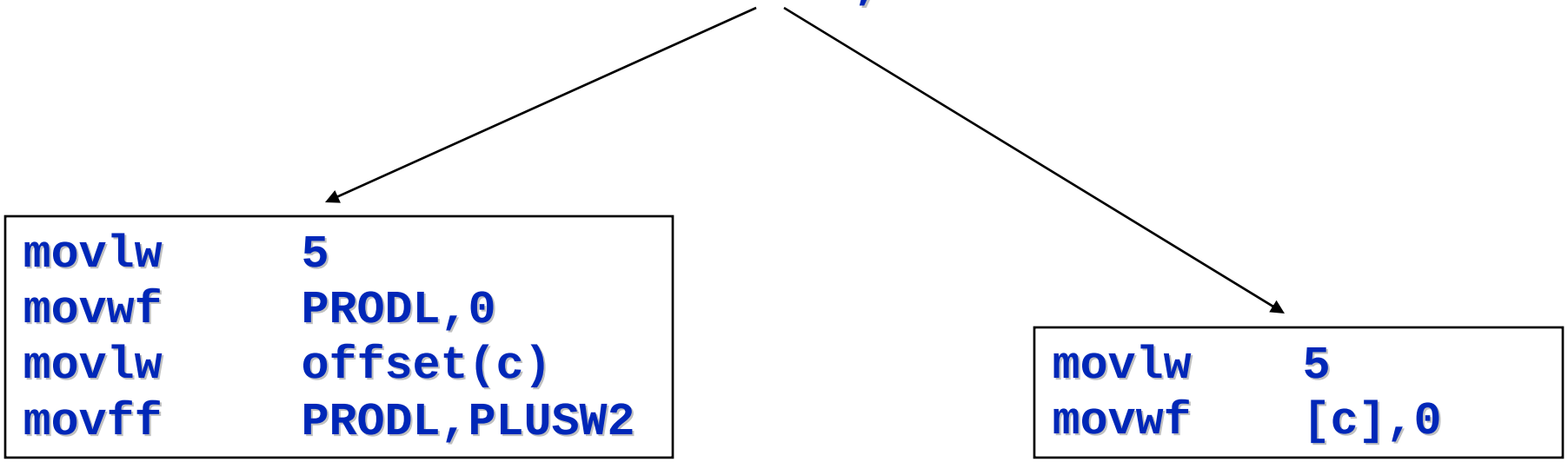
- **Conclusion:**
  - Extended mode reduces the code size
  - Extended mode reduces program execution time

LAB 8 Explanation

PIC18F: Extended Architecture Advantages

Example of improvement on simple assignment:

```
c = 5;
```



```
movlw     5
movwf     PRODL,0
movlw     offset(c)
movff     PRODL,PLUSW2
```

Traditional PIC18F

10 bytes

```
movlw     5
movwf     [c],0
```

Extended PIC18F

4 bytes

PIC18F: Extended Architecture Advantages

Example of improvement for function prologue:

Single bank stack:

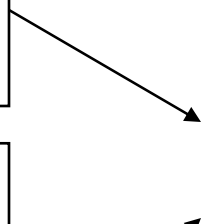
```
movlw     FrameSize
addwf     FSR1L,1,0
```

Multi-bank stack:

```
movlw     FrameSize
addwf     FSR1L,0,0
bnc       PC+6
setf      FSR1L,0
movf      POSTINC1,1,0
movwf     FSR1L,0
```

Traditional PIC18F

4-12 bytes



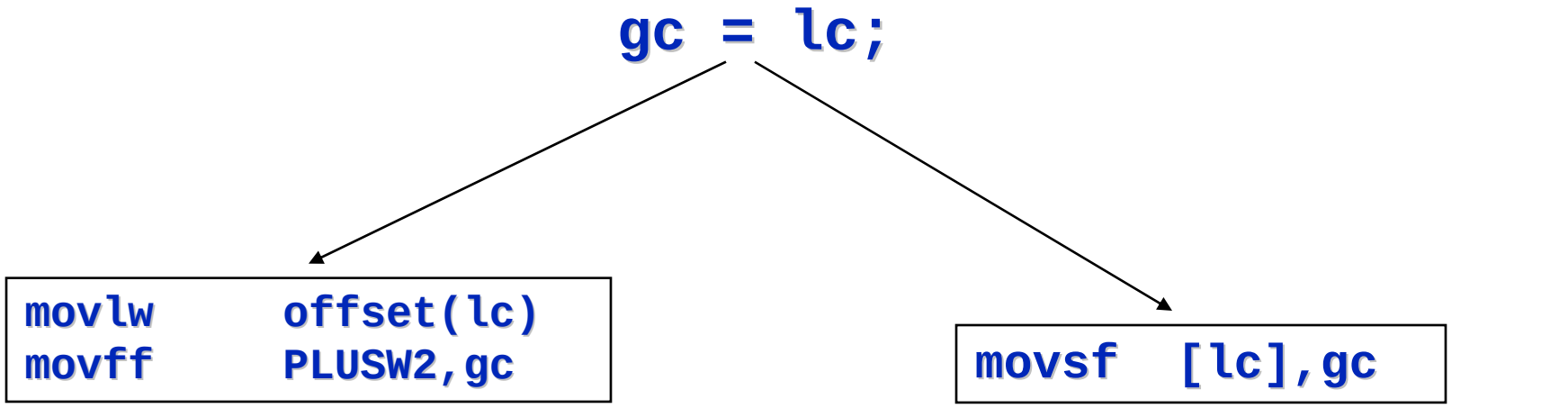
```
addfsr FrameSize,1
```

Extended PIC18F

2 bytes

PIC18F: Extended Architecture Advantages

Example of Improvement in Stack to Global move:



Traditional PIC18F

6 bytes, affects WREG

Extended PIC18F

4 bytes, WREG unchanged

PIC18F: Extended Architecture Advantages

Example of improvement in Stack to Stack move:

```
lc1 = lc2;
```

```
movlw     offset(lc2)
movf      PLUSW2,0,0
movwf     INDF1,0
movlw     offset(lc1)
movff     INDF1,PLUSW2
```

Traditional PIC18F

12 bytes, affects WREG

```
movss  [lc2],[lc1]
```

Extended PIC18F

4 bytes, WREG unchanged

PIC18F: Extended Architecture Advantages

Example of improvement in Function pointer invocation:

```
fn();
```

```
bra     PC+12
movff   fn+2,PCLATU
movff   fn+1,PCLATH
movlb   fn
movf    fn,0,1
movwf   PCL,0
rcall   PC-10
```

Traditional PIC18F

18 bytes, 2 branches

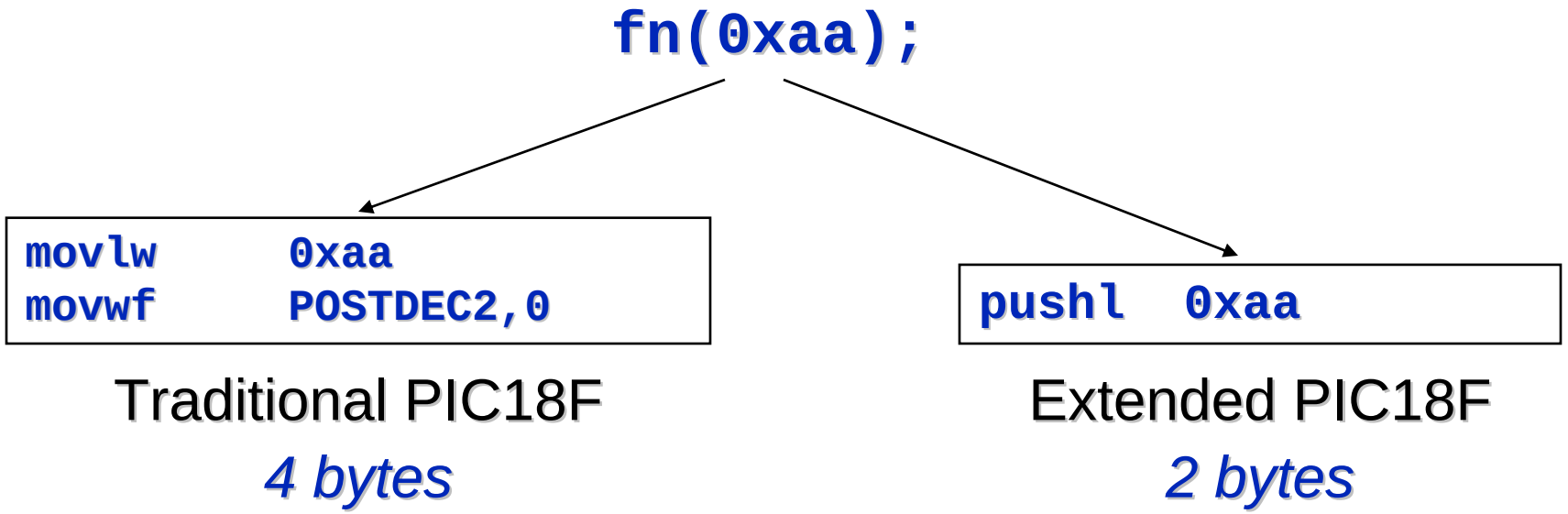
```
movff fn+2,PCLATU
movff fn+1,PCLATH
movlb fn
movf  fn,0,1
callw
```

Extended PIC18F

14 bytes, 0 branches

PIC18F: Extended Architecture Advantages

Example of improvement in passing a Literal Parameter:



UNIVERSITY OF MICROCHIP
UOFM
MASTERS 2007

Oscillators and Power-Managed Modes

PIC18F: Clock System

- **Clock Sources**
  - Primary
    - Fixed Selection
    - LP, XT, HS, RC, EC, Int RC Osc
  - Secondary
    - Timer1 Oscillator - fixed frequency
    - Required for Real-Time Clock time base
  - Internal RC Oscillator
    - INTOSC (8 MHz) source
      - 4, 2, 1 MHz, 500, 250, 125 and 31 kHz
    - INTRC (31 kHz) source

PIC18F: Clock System

- **New Internal RC Oscillator**
  - **2 separate RC sources**
    - 8 MHz (INTOSC)
    - 31 kHz (INTRC)
  - **2-31 kHz sources**
  - **INTOSC 8 & 4 MHz can be routed through PLL**
    - 16 or 32 MHz
  - **Modifying IRCF<2:0> bits immediately selects a different INTOSC postscaler tap**

University of Microchip UofM MASTERs 2007

PIC18F: Power Managed Modes

3 Categories

RUN - 3 clock sources

IDLE - 3 clock sources

SLEEP - no clocks

Total = 7 Modes

PIC18F: Power Managed Modes

- **PRI_RUN Mode**
  - **Config Word defines Primary Clock Source**
    - FOSC3:FOSC0 (_CONFIG1H<3:0>) 10 modes
      - Crystal Oscillator - LP, XT, HS, HSPLL
      - External Clock - EC, ECIO
      - External RC Oscillator - RC, RCIO
      - Internal RC Oscillator - INTIO1, INTIO2
- **SEC_RUN Mode**
  - **Clock switching mechanism in other PIC18 controllers**
  - **Timer1 source, Primary oscillator is disabled**
- **RC_RUN Mode**
  - **IRCF<2:0> selects clock speed**
  - **IOFS set after 1us (typ.) delay if Freq ≠ 31 kHz**

PIC18F
Special Features

PIC18F Special Features

- Wide operating voltage range: 2.0V to 5.5V
- 100,000 erase/write cycle Enhanced Flash program memory (typical)
- 1,000,000 erase/write cycle Data EEPROM memory (typical)
- Flash/Data EEPROM Retention: 100 years typical

PIC18F Special Features

- Extended Watchdog Timer (WDT):
  - Programmable period from 4 ms to 131s
- In-Circuit Serial Programming™ (ICSP™) programming capability
- Programmable High/Low-Voltage Detection (HLVD) module
  - Supports interrupt on High/Low-Voltage Detection
- Programmable Brown-out Reset
  - Enable during Configuration Or
  - Software enable option

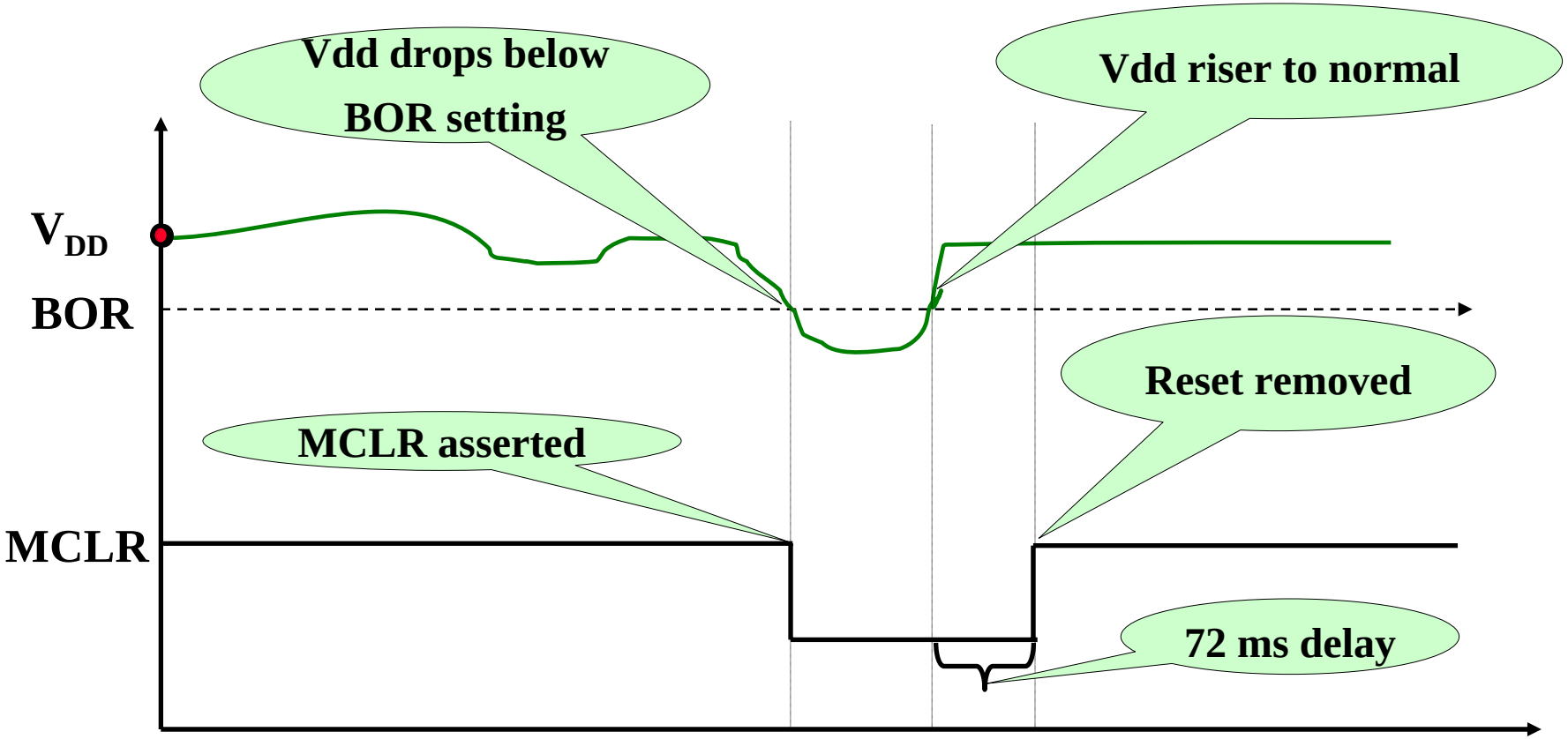
PIC18F Special Features
Programmable Low-Voltage Detect

- 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

PIC18F Special Features
Programmable Brown-Out Reset

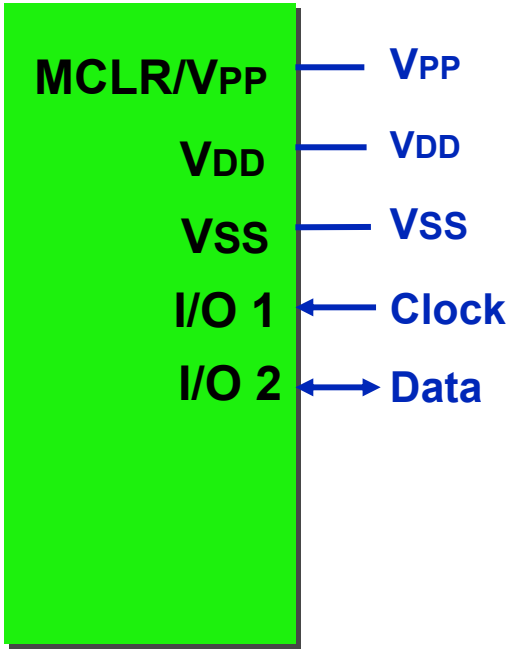
- Monitors operating voltage range
- Resets MCU during predefined range
  - Removes Reset after Vdd returns to normal (+ 72 ms)
- Programmable internal reference
  - Up to 4 references (2.0, 2.7, 4.2, 4.5)
- Enabled via Configuration register

PIC18F Special Features
Programmable Brown-Out Reset



PIC18F Special Features
In-Circuit Serial Programming™ programming capability

- In-System Programming Method
- Uses only two pins to send/receive data
- Non-intrusive to normal operation
- Advantages of ICSP™ programming capability
  - Reduce cost of field upgrades
  - Calibrate and Serialize Systems during manufacturing
  - Reduce handling



PIC18F Special Features
Watchdog Timer (WDT)

- Recovers from software malfunction
- Resets MCU if not attended on-time
  - Software must clear it periodically (CLRWDT)
- Programmable period
  - 4 ms to 131.0 s typical
- Configuration controlled postscaler
- Enabled via Configuration register or Software
- Wakes up CPU from Sleep/Idle mode

PIC18F Special Features RESETS

- PIC18 RESETS
  - Power-on Reset (POR)
  - MCLR Reset during normal operation
  - Programmable Brown-out Reset (BOR)
  - Watchdog Timer (WDT) Reset (during execution)
  - RESET Instruction
  - Stack Full Reset
  - Stack Underflow Reset
- For all resets PC vectors to address 0

PIC18F Special Features
Reset Registers

- **After Reset PC will have the address 0x000000**
- **Following bits will be affected after each Reset**
  - **POR = '0':** *Power-On Reset*
  - **BOR = '0' & POR = '1':** *BOR Reset*
  - **TO = '0':** *WDT Reset*
  - **RI = '0':** *Reset Instruction*
  - **STKFUL = '1':** *Stack over flow Reset*
  - **STKUNF = '1':** *Stack under flow Reset*
  - **POR, BOR, TO & RI = '1' and STKFUL & STKUNF = '0':** *MCLR Reset*

Lab 9: How to know the cause of Reset in real application

- **Goals:**
  - Understand PIC18 Reset Sources
  - Understand significance of each of the resets
  - Identifying the source of reset in application

- **The demo is comprised of two parts**
  - PART – 1: Writing software to handle reset types
  - PART – 2: Simulating the condition to generate the reset

Lab 9: How to know the cause of reset in real application

- **Code for Lab 9:**
  - **Open the project Lab_9.mcp**
  - **Write C source code in the file Lab_main.c to identify and handle each of the resets as follows**
    - POR Reset
      - Check for POR and BOR bits in RCON register, if only POR bit is not set it is a POR
      - Write inactive value to both RCON and STKPR registers
      - Call a function named “POR_Occured” which will display reset type on the on-board display

Lab 9: How to know the cause of reset in real application

- **Code for Lab 9: (Continued…)**
  - **BOR reset**
    - Check for POR and BOR bits in RCON register, if both bits are not set then it is a BOR
    - Write inactive value to both RCON and STKPR
    - Call a function named “BOR_Occured” which will display reset type on the on-board display
  - **MCLR Reset**
    - Check for POR and BOR bits in RCON and STKPTR register, if both register reset status bits are unchanged then it is MCLR reset
    - Write inactive value to both RCON and STKPR
    - Call a function named “MCLR_Occured” which will display reset type on the on-board display

Lab 9: How to know the cause of reset in real application

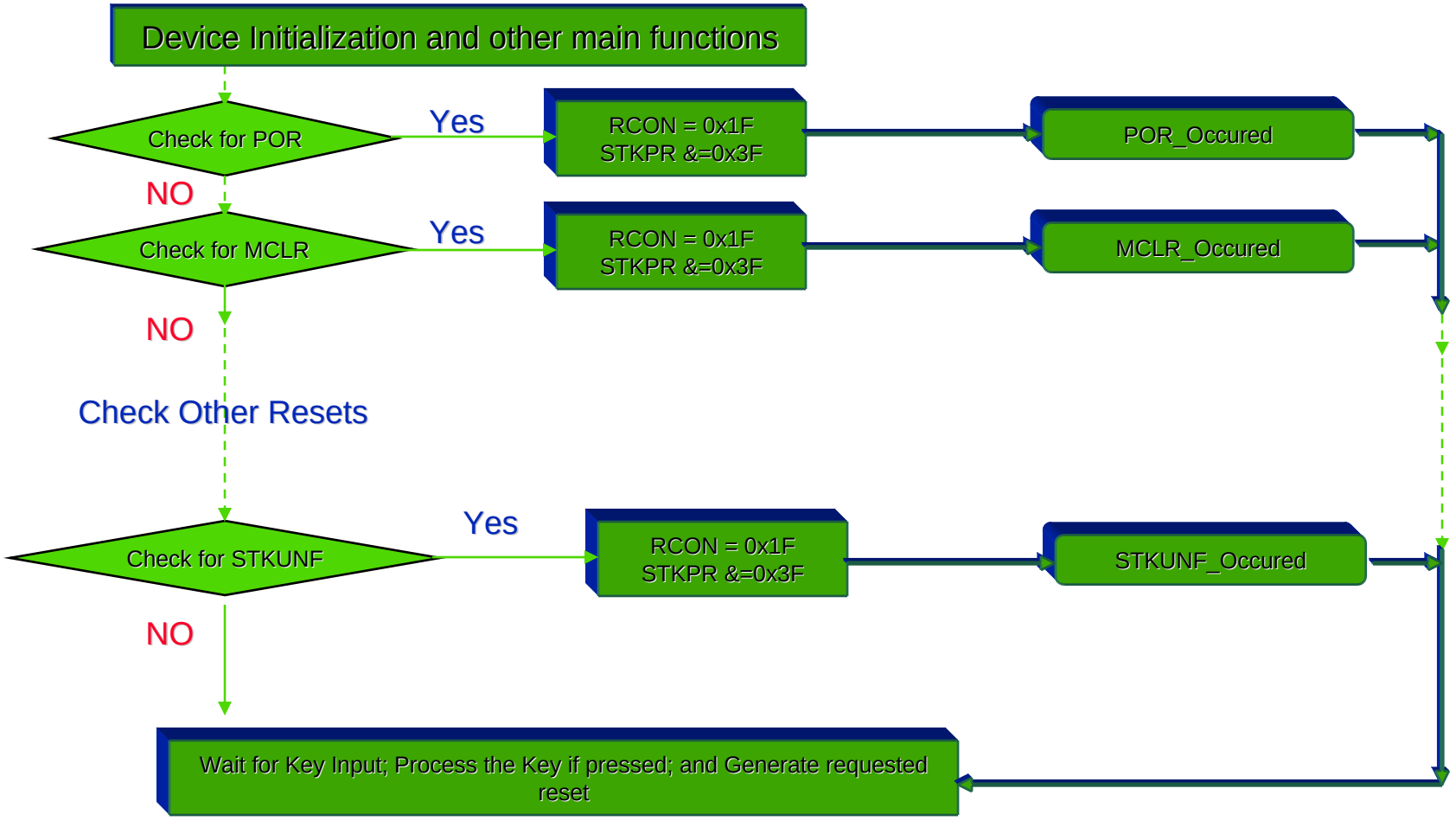
- **Code for Lab 9: (Continued…)**
  - **Watchdog Timer reset**
    - Check for TO bit in RCON register, if bit is not set it is a Watchdog reset
    - Write inactive value to both RCON and STKPR
    - Call a function named “WDT_Occured” which will display reset type on the on-board display
  - **Software reset**
    - Check for RI bit in RCON register, if bit is not set then the reset is due to execution of software reset instruction
    - Write inactive value to both RCON and STKPR
    - Call a function named “RI_Occured” which will display reset type on the on-board display

Lab 9: How to know the cause of reset in real application

- **Code for Lab 9: (Continued…)**
  - **Stack Full reset**
    - Check for STKFUL bit in STKPTR register, if bit is set, it is a Stack Overflow reset
    - Write inactive value to both RCON and STKPR
    - Call a function named “StkOvFlo_Occured” which will display reset type on the on-board display
  - **Stack Underflow reset**
    - Check for STKUNF bit in STKPTR register, if bit is set, it is a Stack Underflow reset
    - Write inactive value to both RCON and STKPR
    - Call a function named “StkUnFlo_Occured” which will display reset type on the on-board display

Lab 9: How to know the cause of reset in real application

Flow chart for Lab 9:



Lab 9: How to know the cause of reset in real application

- Lab 9: PART-2 Simulating the Reset Conditions
  - Build the Project
  - Program the Microcontroller using MPLAB® ICD 2
  - Turn OFF power to the Demo board
  - Remove the ICD cable from the Demo board

Lab 9: How to know the cause of reset in real application

- **Lab 9: PART-2 Simulating the Reset Conditions**

Now Turn ON the power supply to the

Demo board

It will display Power ON reset message on LCD

Lab 9: How to know the cause of reset in real application

- **Lab 9: PART-2 Simulating the Reset Conditions**

Now Press and release Reset button on the

Demo board

It will display MCLR reset message on LCD

Lab 9: How to know the cause of reset in real application

- **Lab 9: PART-2 Simulating the Reset Conditions**

Now Press Switch SW1 on the

Demo board

It will display Watchdog reset message on LCD

Lab 9: How to know the cause of reset in real application

- **Lab 9: PART-2 Simulating the Reset Conditions**

Now Press Switch SW1 on the

Demo board

It will display Software reset message on LCD

Lab 9: How to know the cause of reset in real application

- **Lab 9: PART-2 Simulating the Reset Conditions**

Now Press Switch SW1 on the

Demo board

It will display Stack Full reset message on LCD

Lab 9: How to know the cause of reset in real application

- **Lab 9: PART-2 Simulating the Reset Conditions**

Now Press Switch SW1 on the

Demo board

It will display Stack Under flow reset message on LCD

Summary

- We discussed Architecture overview
  - PIC18F Architecture
  - Programmer's Model
  - Instruction set Overview
  - Interrupt Handling and latency
- Reviewed Basic Peripherals
  - I/O Ports
  - Development Tools
  - Comparators, Voltage Reference
  - ADC

Summary

- Timers
- CCP (Capture, Compare & PWM)
- MSSP (I^2C™ & SPI / Microwire)
- Table Read and Write Operations
- USART

- **We also covered:**
  - Extended Architecture
  - Oscillator and power saving modes
  - Special features of PIC18F controllers

Summary

- **Carried out the Labs on**
  - IO Port initialization, reading and writing to the Ports
  - Initialization and ADC conversion
  - Time configuration
  - Generating PWM
  - Capturing the input signal
  - I^2C™ module configuration
  - Comparison of standard and extended instruction set
  - Reset identification

Thank You

UNIVERSITY OF MICROCHIP
UOFM
MASTERS 2007

Trademarks

The Microchip name and logo, the Microchip logo, Accuron, dsPIC, KeeLoq, KeeLoq logo, microID, MPLAB, PIC, PICmicro, PICSTART, PRO MATE, rfPIC and SmartShunt are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

AmpLab, FilterLab, Linear Active Thermistor, Migratable Memory, MXDEV, MXLAB, SEEVAL, SmartSensor and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, Application Maestro, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, ECAN, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, PICkit, PICDEM, PICDEM.net, PICLAB, PICtail, PowerCal, PowerInfo, PowerMate, PowerTool, REAL ICE, rfLAB, Select Mode, Smart Serial, SmartTel, Total Endurance, UNI/O, WiperLock and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.