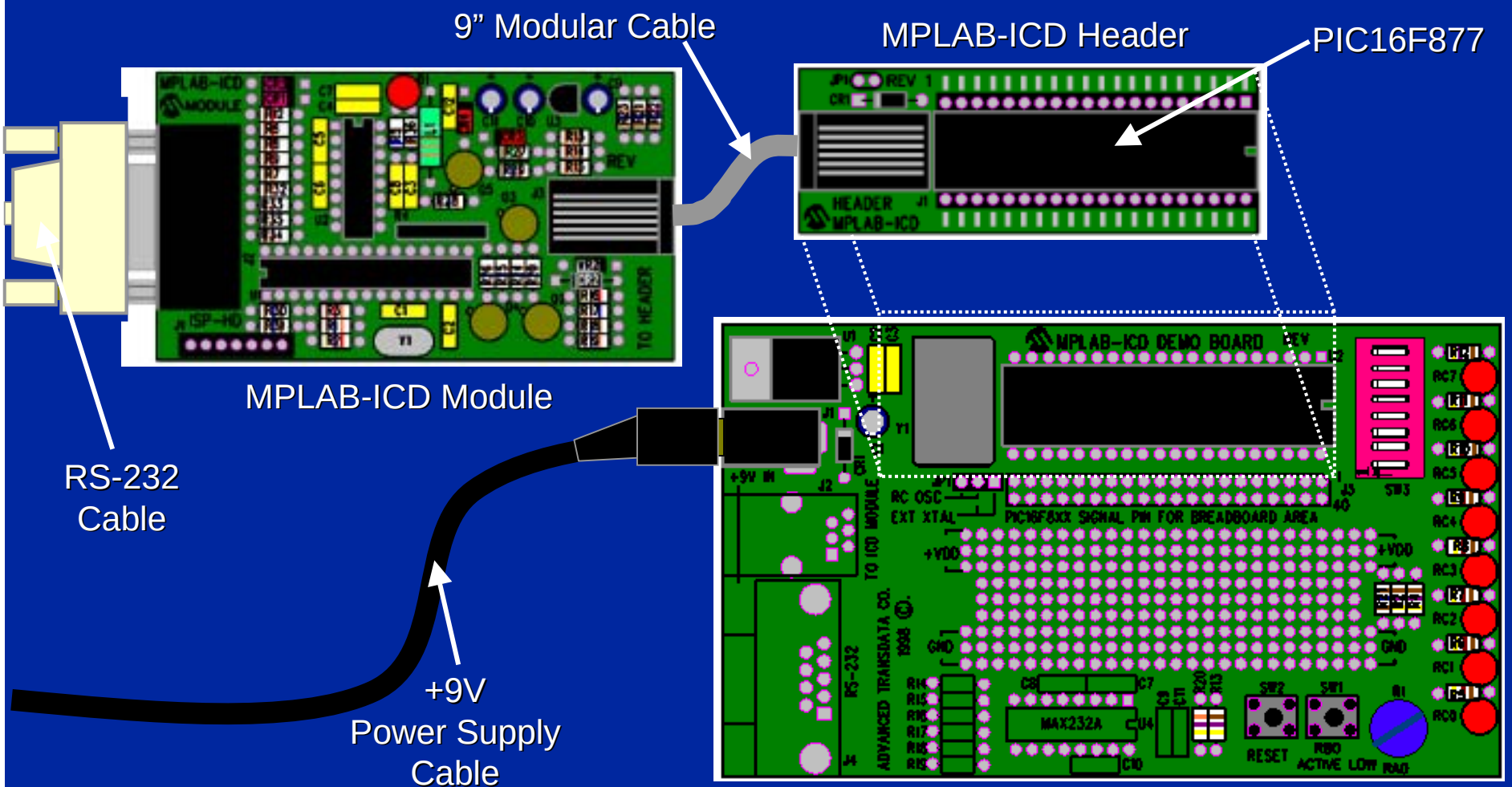
MICROCHIP

MPLAB-ICD Workshop Demo Board

MICROCHIP

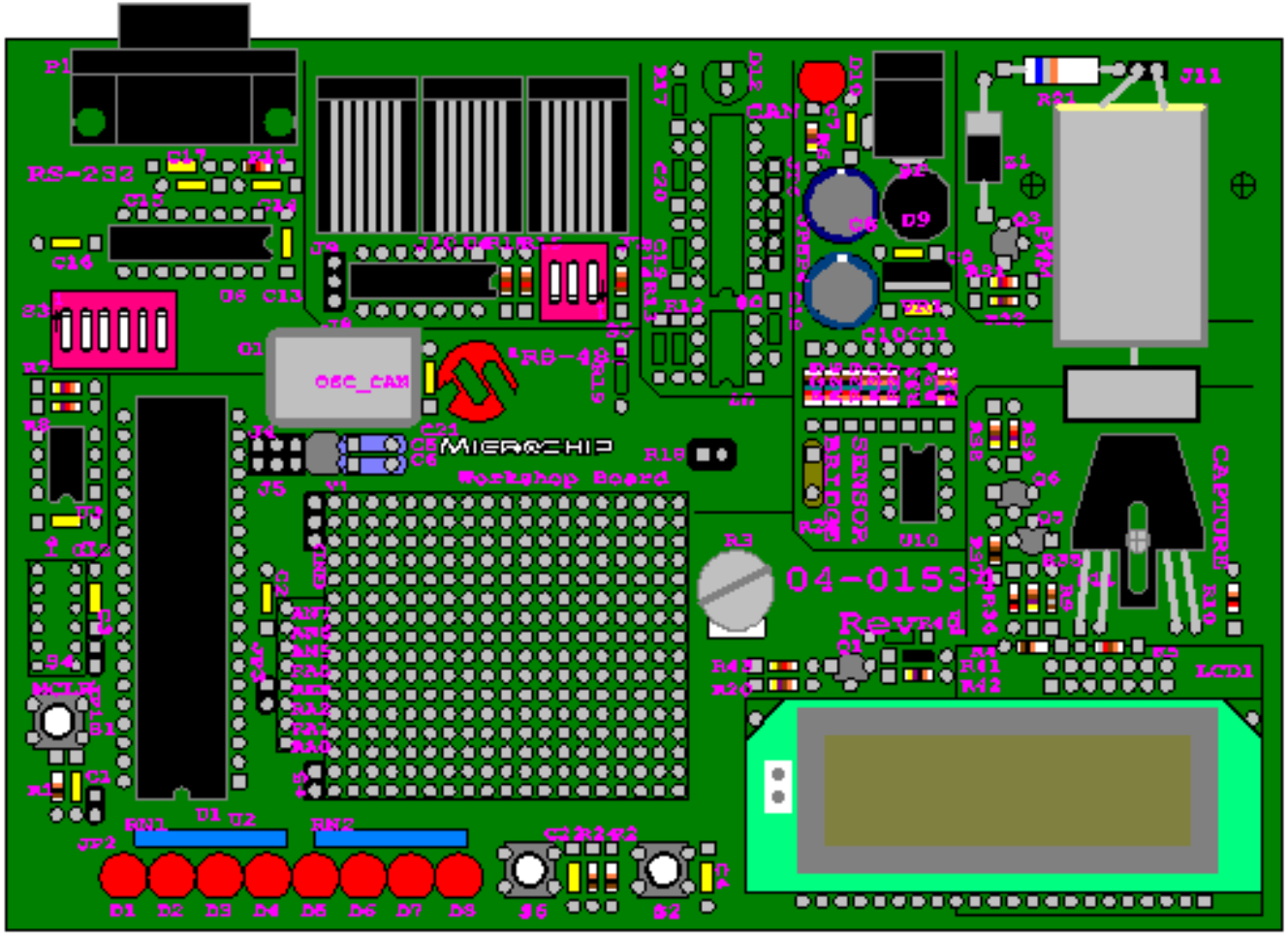
In-Circuit Debugger Block Diagram



In-Circuit Debugger

- When enabled, you will lose
  - 1 Program Memory location-0000h must be a NOP
  - 1 Level of Stack
  - Last 256 Words of Program Memory, 1F00h to 1FFFh
  - 6 bytes of Data Memory (70h, 1EBh to 1EFh)
  - I/O pins RB6 and RB7

MCU201 Workshop Demo Board



MICROCHIP

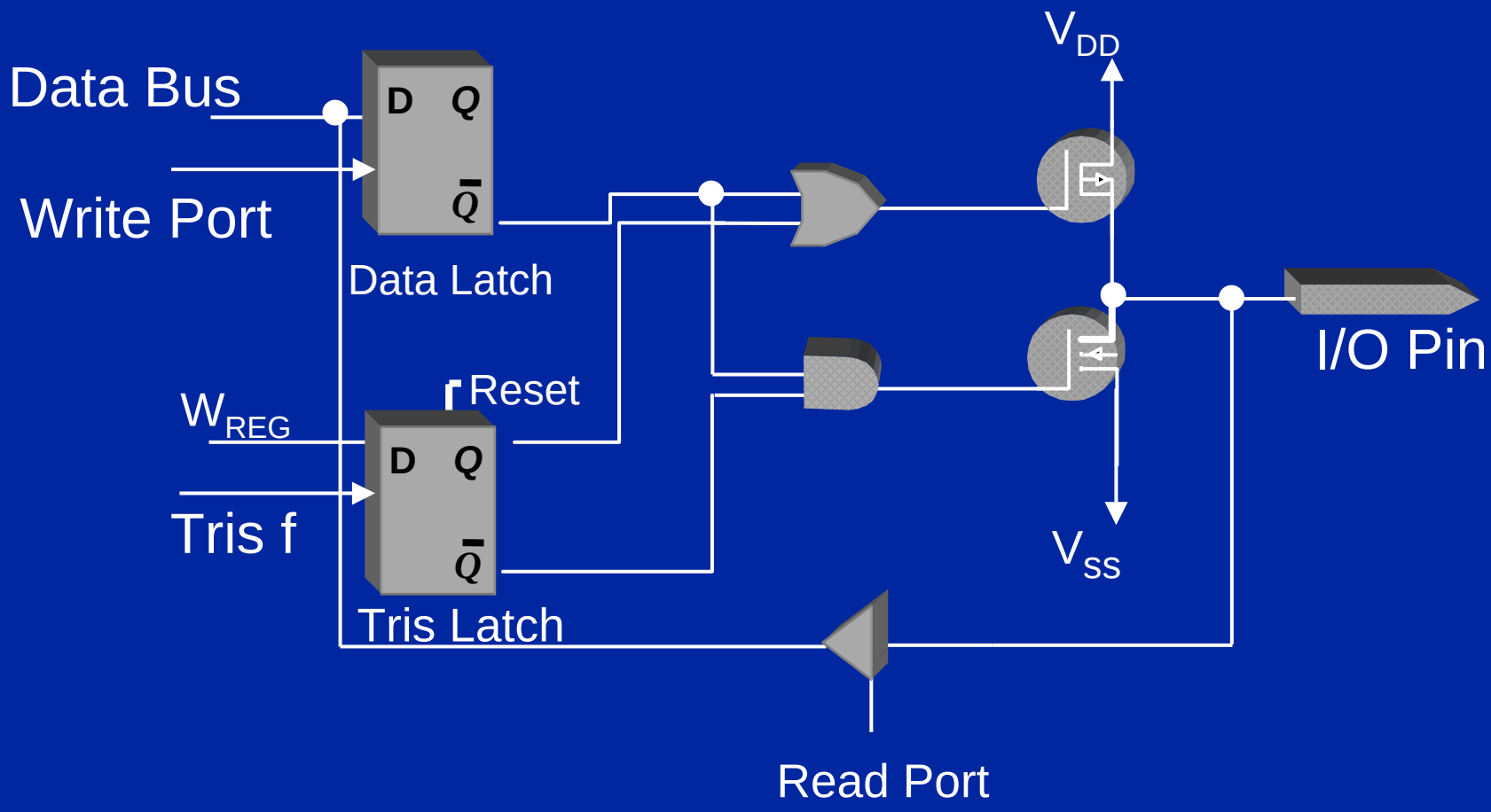
Peripherals

MICROCHIP

Peripherals
Digital I/O Ports

- Up to 33 bi-directional I/O pins
- Some are multiplexed with peripheral functions
- High drive capability (up to 25mA sink/source)
- Can directly drive LEDs
- Direct bit (pin) manipulation (single-cycle)
- Each port pin has individual direction control (software-controlled)
- All pins have ESD protection

Peripherals
Digital I/O Ports

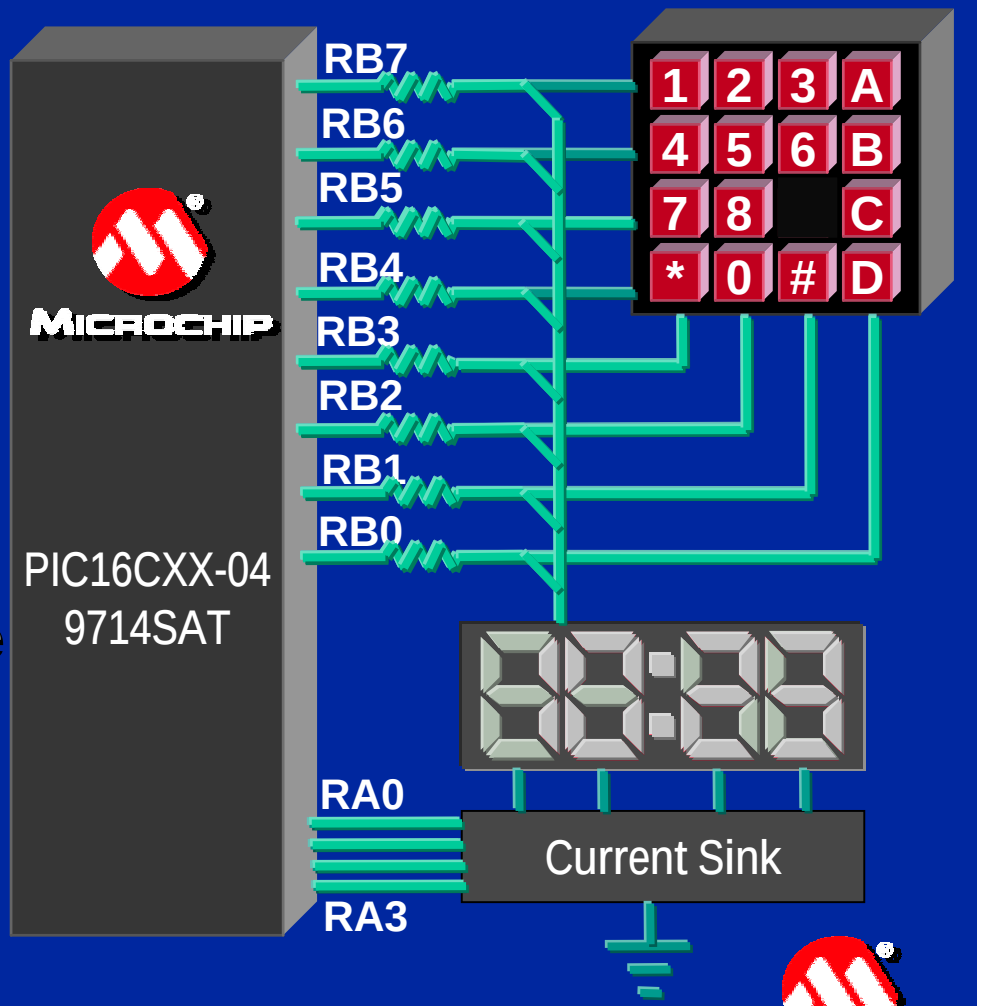


I/O pins have ESD protection diodes

PIC16CXX Peripherals
Digital I/O Ports Application

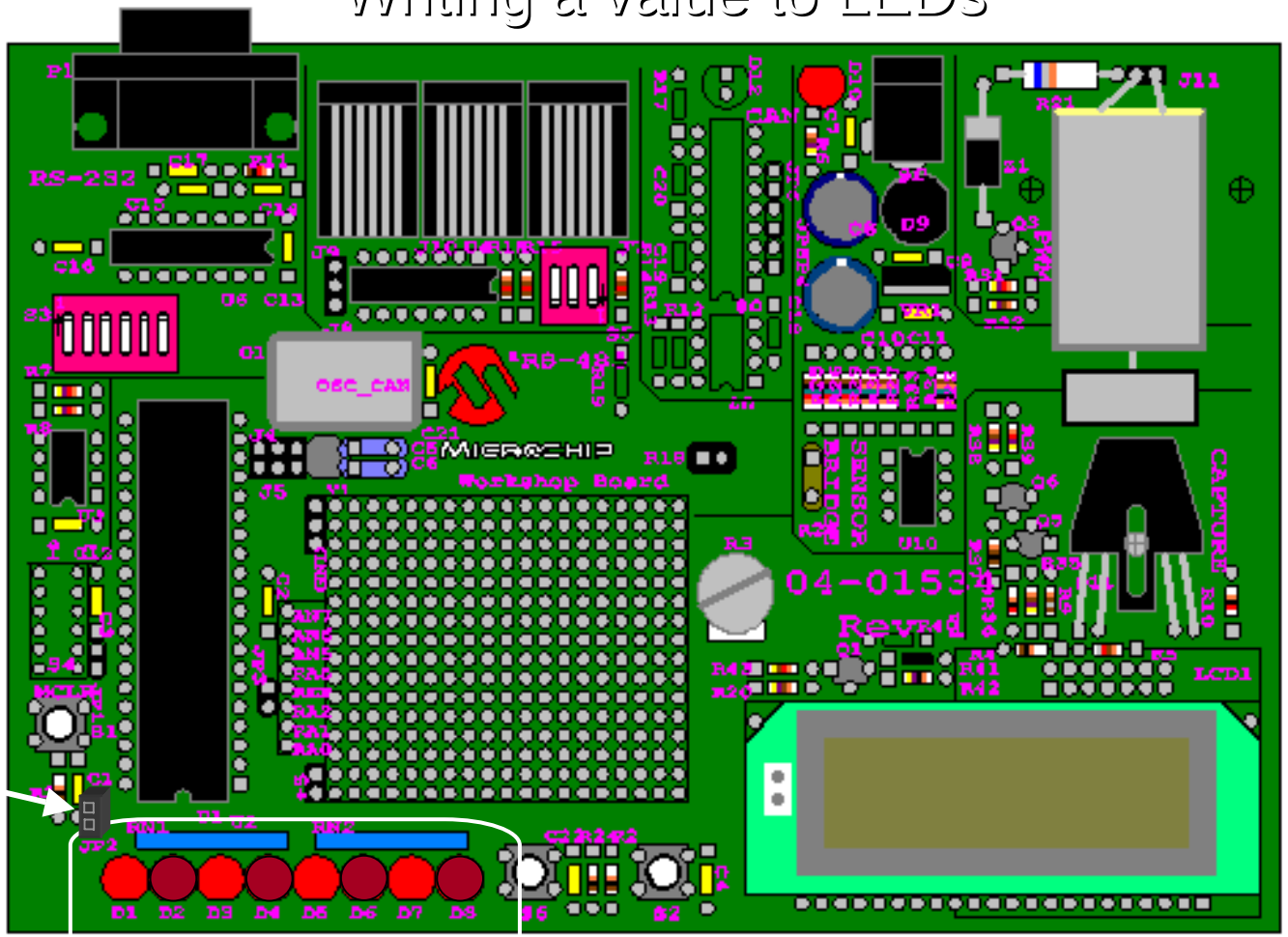
Clock with Keypad

- 7 Segment LEDs MUXed with keypad with minimal software overhead
- LEDs strobed every 20ms for 5ms
- Keys sampled every 20ms for 100µs, w/LEDs OFF
- Most pins have two or more function capability.
- Efficient use of I/O lines allows multiple functions with reduced pin count



Lab 1: Digital I/O Ports

Writing a value to LEDs



Install Jumper

LEDs on PORTD

Make PORTD all Outputs, = 00h

Delay, then Increment PORTD

Lab 1: Digital Ports

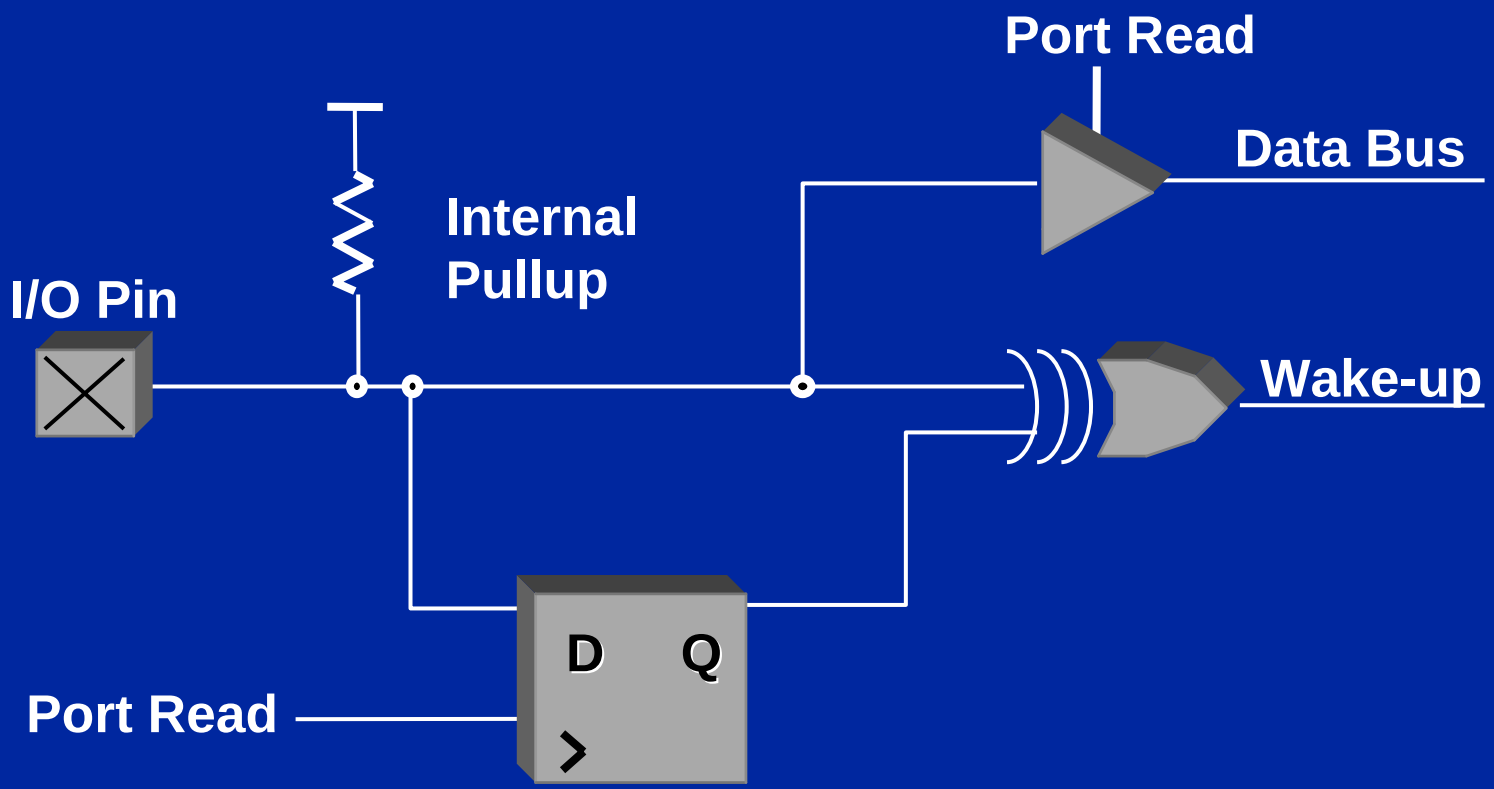
- Create a new project lab1.pjt
- Create a new source code file
- Save it as lab1.asm
- Add lab1.asm to project
- Add assembly to:
  - Make PORTD all outputs, value of 0xFF
- Loop
  - Delay
  - Increment PORTD

Lab 1: Goals

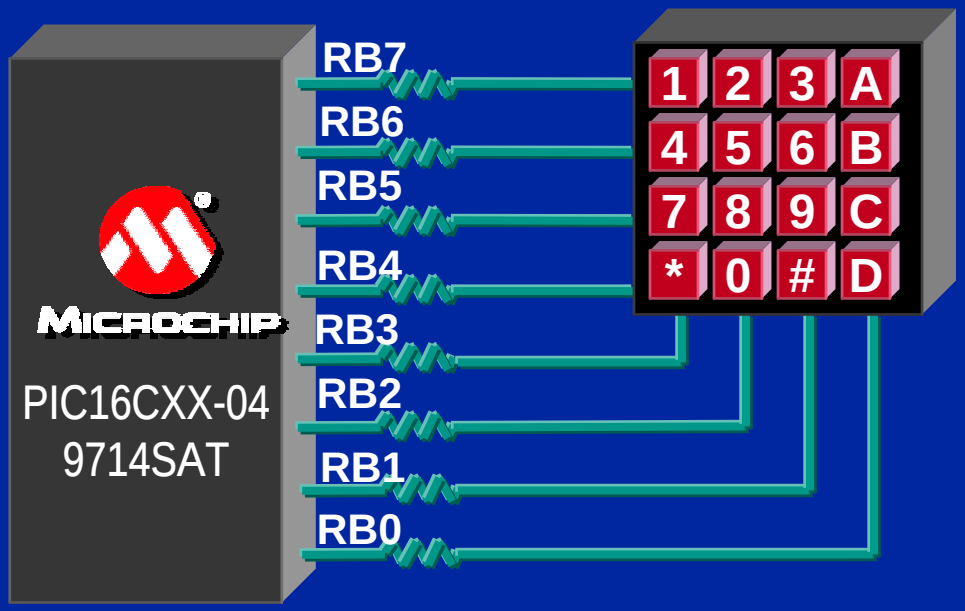
- Using MPLAB/MPASM
  - Creating a project
  - Adding files to a project
  - Writing instructions & assembling a file
- Using the MPLAB-ICD
  - Downloading firmware, Setting Debug mode
  - Run/Stop/Single Step/Breakpoint
- Configuring I/O Ports
- Switching banks
- Writing Delay routines

Peripherals
Digital I/O Ports: PORTB Interrupt on Change

- Internal Pull-Ups and Wakeup/Interrupt On Change feature (up to 8 pins)



Peripherals
Digital I/O Ports: PORTB Interrupt on Change

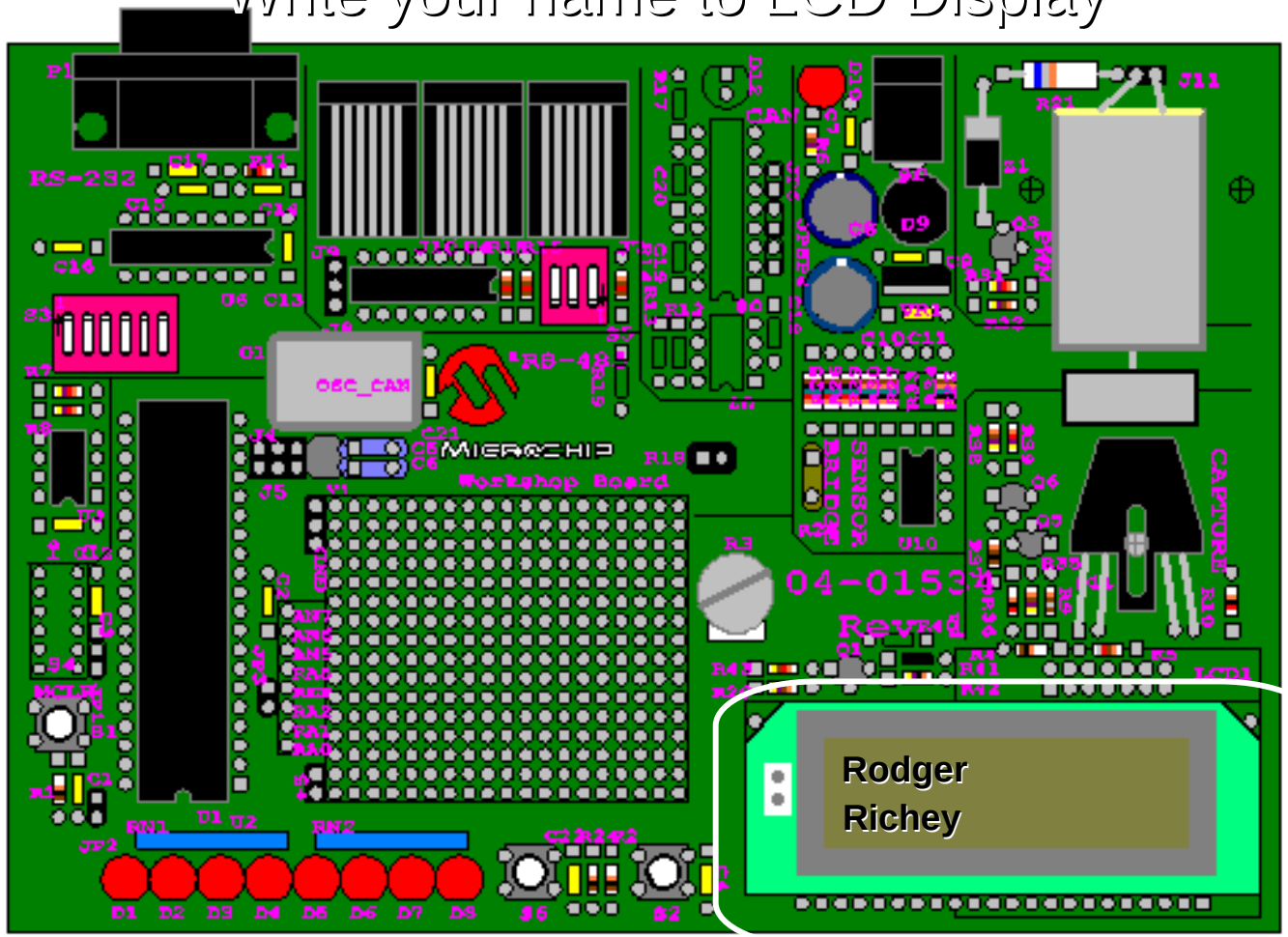


* Optional resistors for ESD protection

- Internal pull-ups (software option) keep RB4-RB7 pins high
- RB0-RB3 pins drive 0's
- Any key pressed pulls down an RB4-RB7 pin and generates an interrupt
- The interrupt wakes processor from sleep
- Saves timer resources

Lab 2: External LCD Display

Write your name to LCD Display

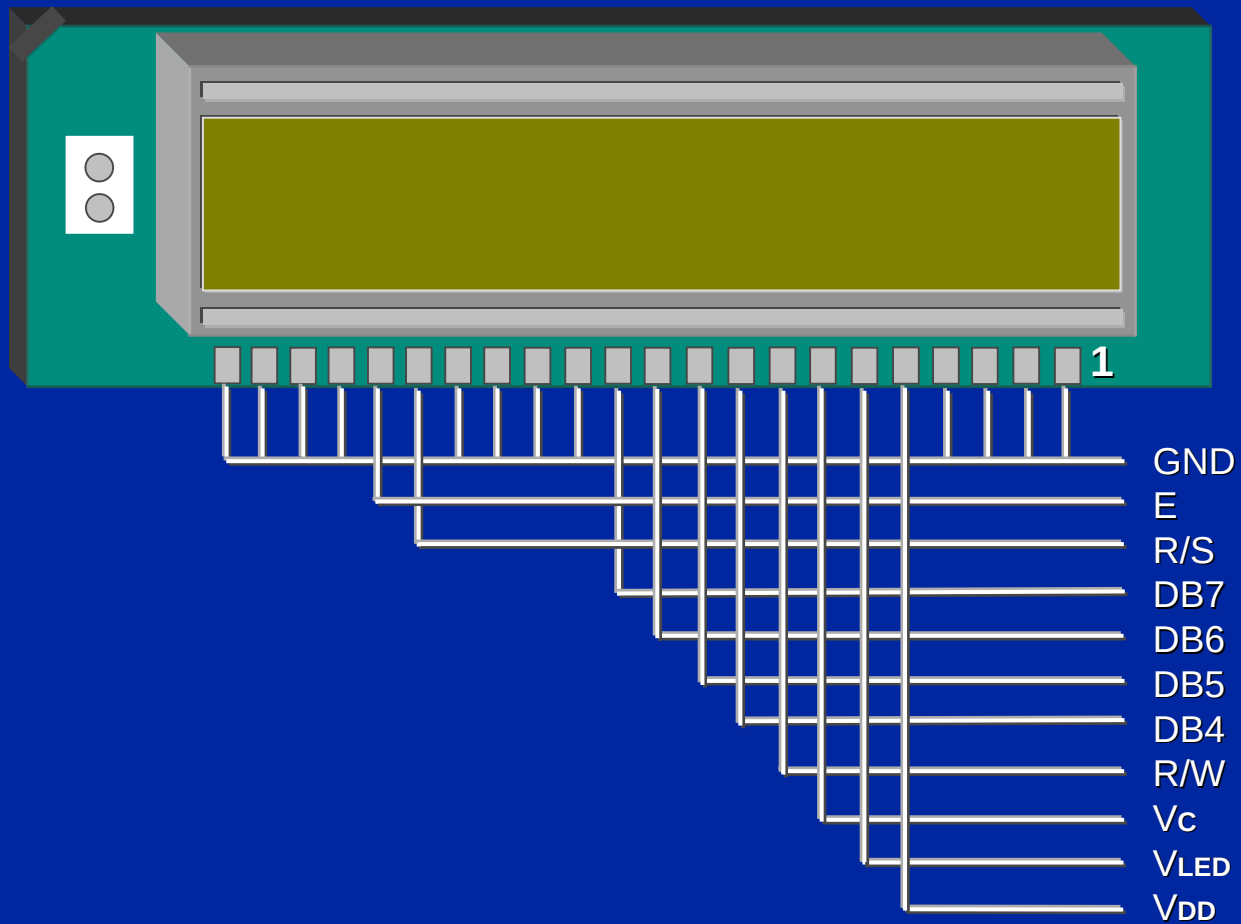


Write your first name to Line1.

Write your last name to Line 2.

LCD Display connected to PORTB

LCD Pinout



External LCD Display Wiring

- 4-bit LCD display, HD44780 compatible
- Control lines:
  - RB4 -> E (clock)
  - RB5 -> RS (register select)
  - R/W (read/write) grounded for write only
  - RB6,7 have to be left unconnected for ICD
- Data lines: RB0-RB3
- LED Backlight control connected to RE0

LCD Subroutines

- InitLCD — Initializes the LCD Display
- putcLCD — Writes a character to LCD
Character in WREG
- clrLCD — Clears LCD
- L1homeLCD — Set cursor to Line 1 home
- L2homeLCD — Set cursor to Line 2 home

Lab 2: External LCD Display

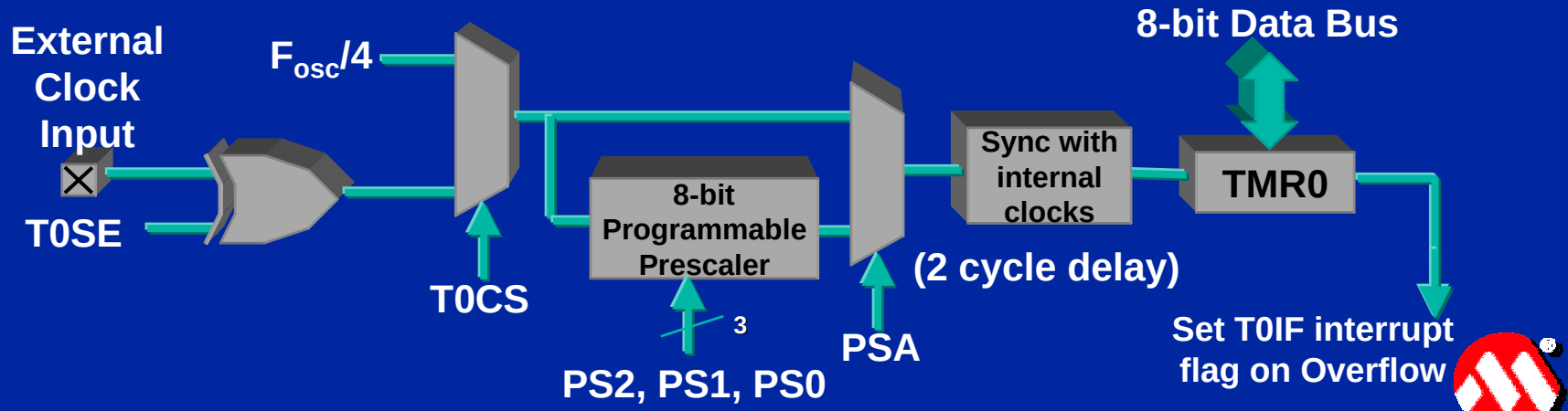
- Open project lab2.pjt
- Initialize the LCD display
- Set cursor to home position on Line 1
- Write your first name
- Set cursor to home position on Line 2
- Write your last name

Lab 2: Goals

- Using the external 4-bit LCD display
  - Initializing the display
  - Setting cursor position
  - Writing data
- Calling subroutines using MPASM
- Writing relocatable assembly source code
- Using MPLINK to link in source code modules
  - LCD routines located in lcd.o

Peripherals
Timer0

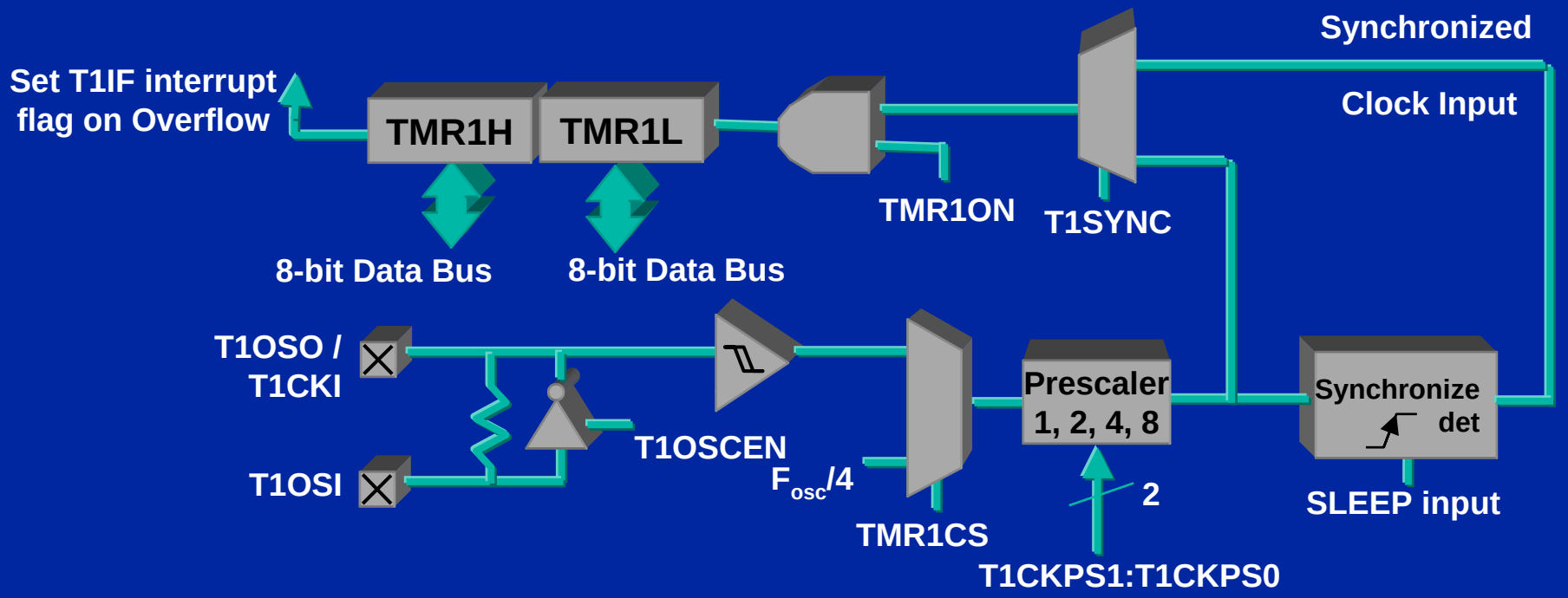
- 8-bit Timer / Counter
- Readable and writeable
- 8-bit software programmable prescaler
- Internal or external clock select
- Interrupt on overflow from FFh to 00h
- Edge select for external clock



Peripherals
Timer1

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

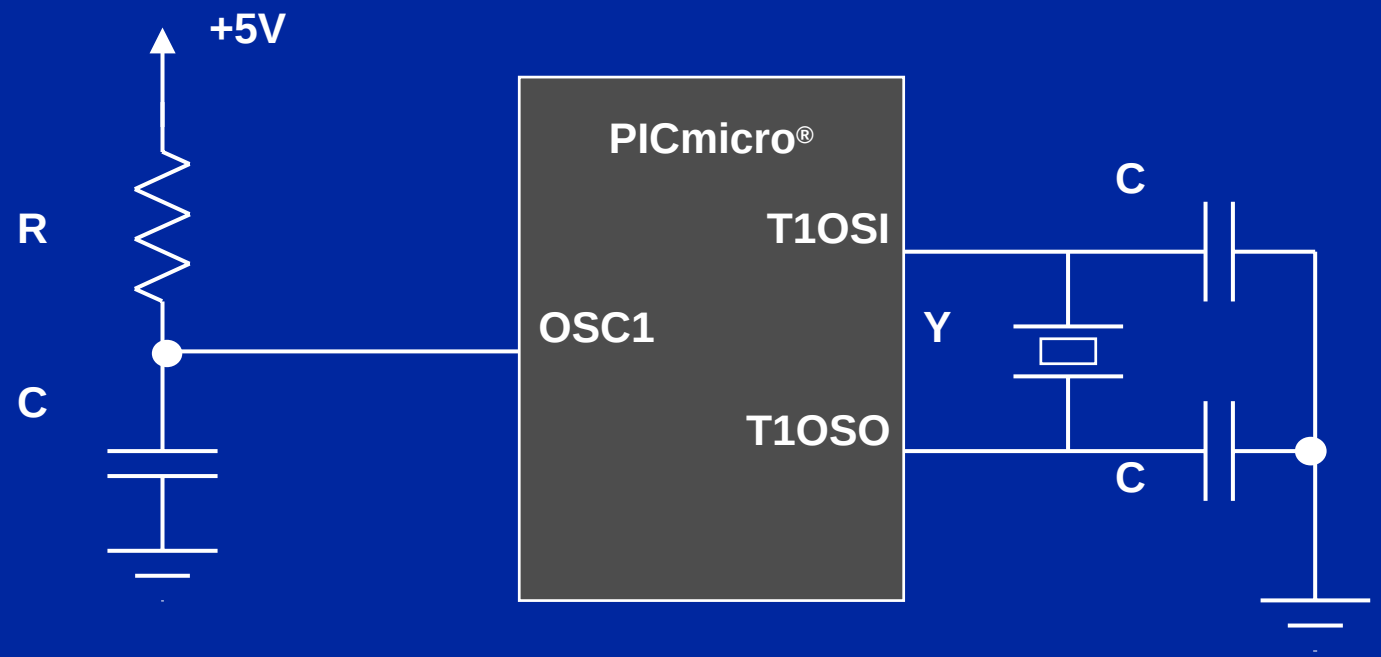
Peripherals
Timer1 (cont)



Peripherals
Timer1 as a Real Time Clock

- **Use a 32.786 kHz crystal on Timer1's oscillator circuit:**
  - Timer1 increments on own oscillator.
  - System can use faster low cost RC oscillator.
  - Timer1 will increment when asleep.
  - When Timer1 overflows, interrupt will be generated.
  - RC oscillator can be calibrated using 32kHz crystal.

Peripherals
Timer1 as a Real Time Clock



Preload TMR1H register for faster overflows:

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

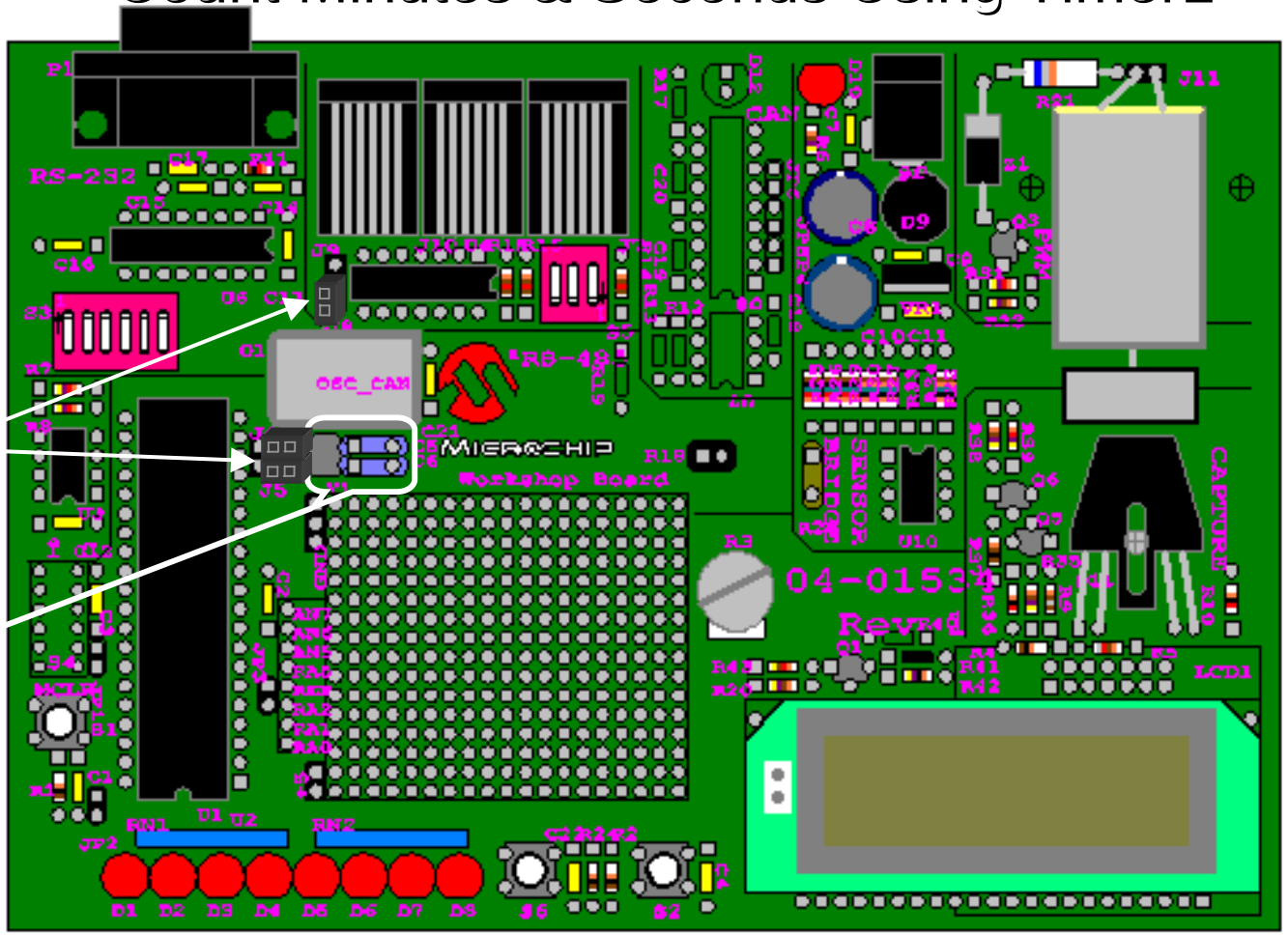
See Application Note AN580 for more info.

Lab 3: Real Time Clock

Count Minutes & Seconds Using Timer1

Install Jumpers

32.768KHz Crystal & Caps



Preload TMR1H with 0x80
Write 00:00 to LCD

When Timer1 Overflows, In the ISR:

Reload TMR1H with 0x80
Update Secs & Mins
Update LCD Display

Lab 3: Real Time Clock

- Open project lab3.pjt
- Configure Timer1:
  - 1:1 Prescale, Oscillator Enabled
  - NOT synchronized, External Clock Source
- Write 00:00 to LCD Display
- In an interrupt service routine whenTimer1 overflows
  - Load TMR1H with 0x80
  - Update Mins and Secs variables
  - Write time to screen w/format MM:SS

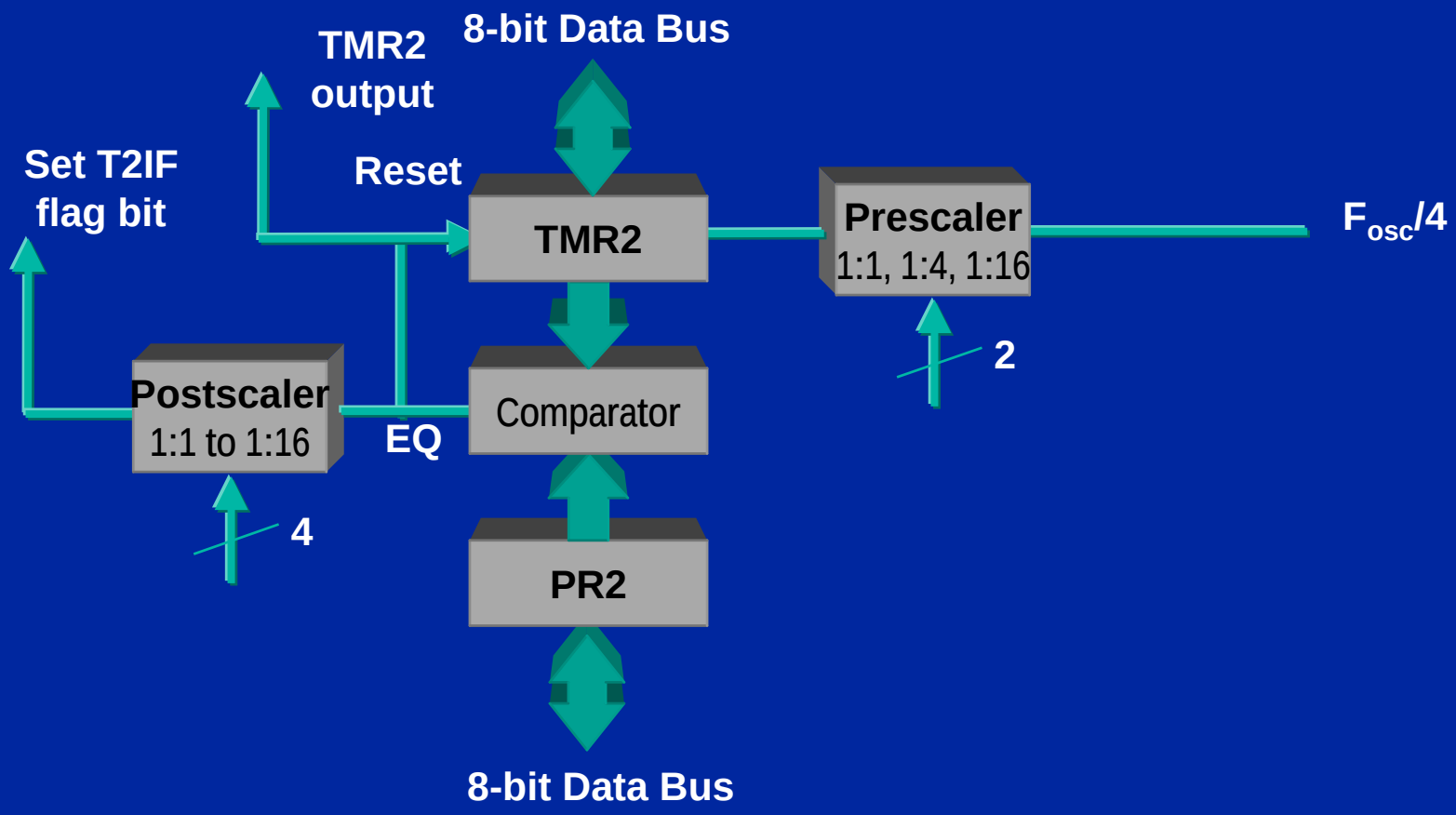
Lab 3: Goals

- Configuring Timer1
- Using Timer1 for as a RTC
- Using interrupt service routines
- Linking object modules
  - lcd.c

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 reg. PR2, then resets to 00h
- TMR2 match with PR2 generates an interrupt through postscaler
- Used as baud clock for SSP (SPI™)

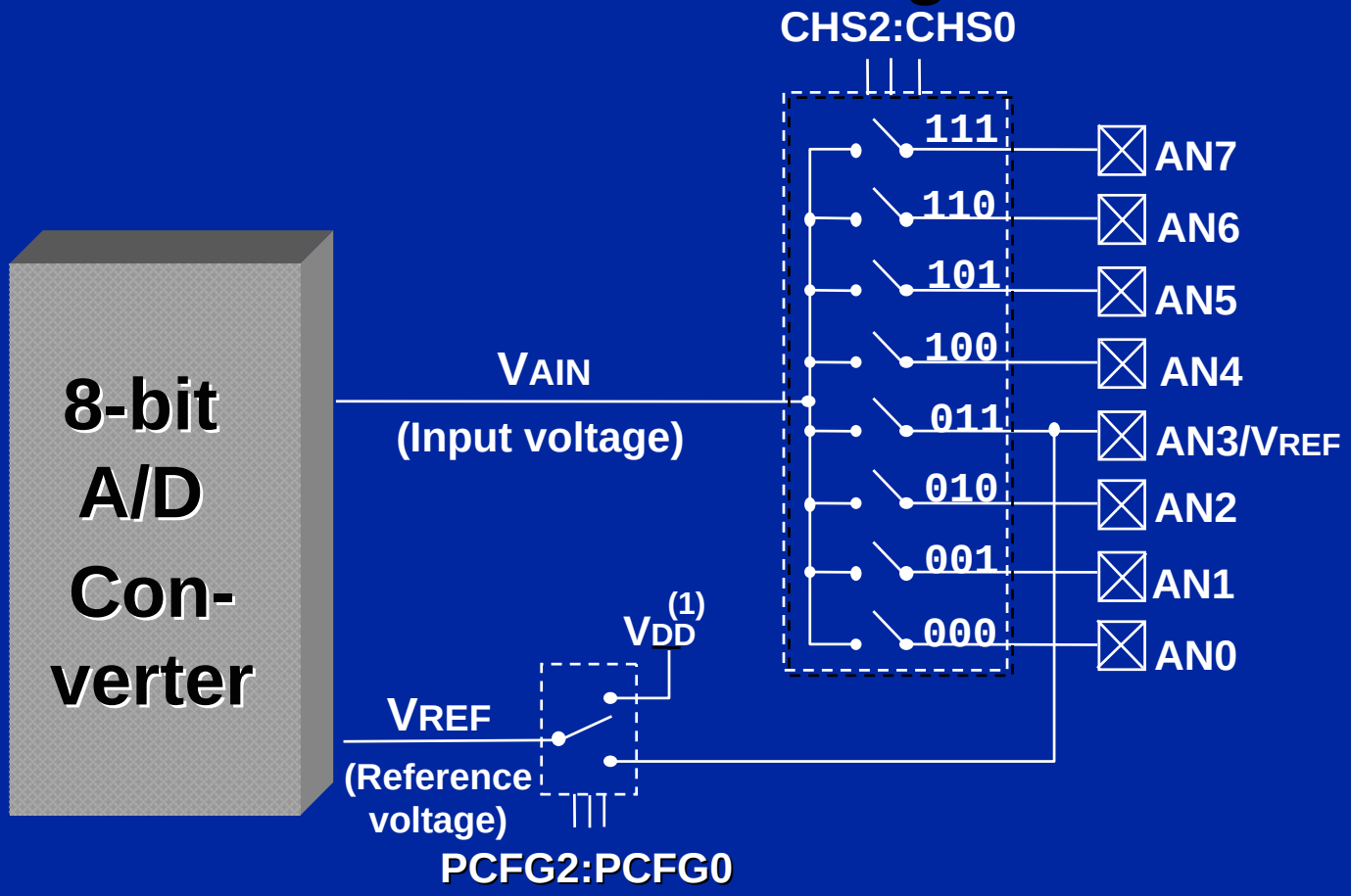
Peripherals
Timer2 (cont)



Peripherals
8-bit Analog-to-Digital (A/D) Converter

- 4 to 8 analog inputs multiplexed into one A/D converter
- 11 µs typical sampling time, as fast as 16 µs conversion time
- 8-bit resolution with ±1 LSb accuracy
- A/D conversion during SLEEP. A/D complete can wake processor
- External reference input, VREF (VREF <= VDD)
- Analog input range: VSS to VREF
- 27 µs for complete A/D result => A/D sampling at 27 kHz
  - 17 µs without channel change

Peripherals
8-Bit A/D Block Diagram

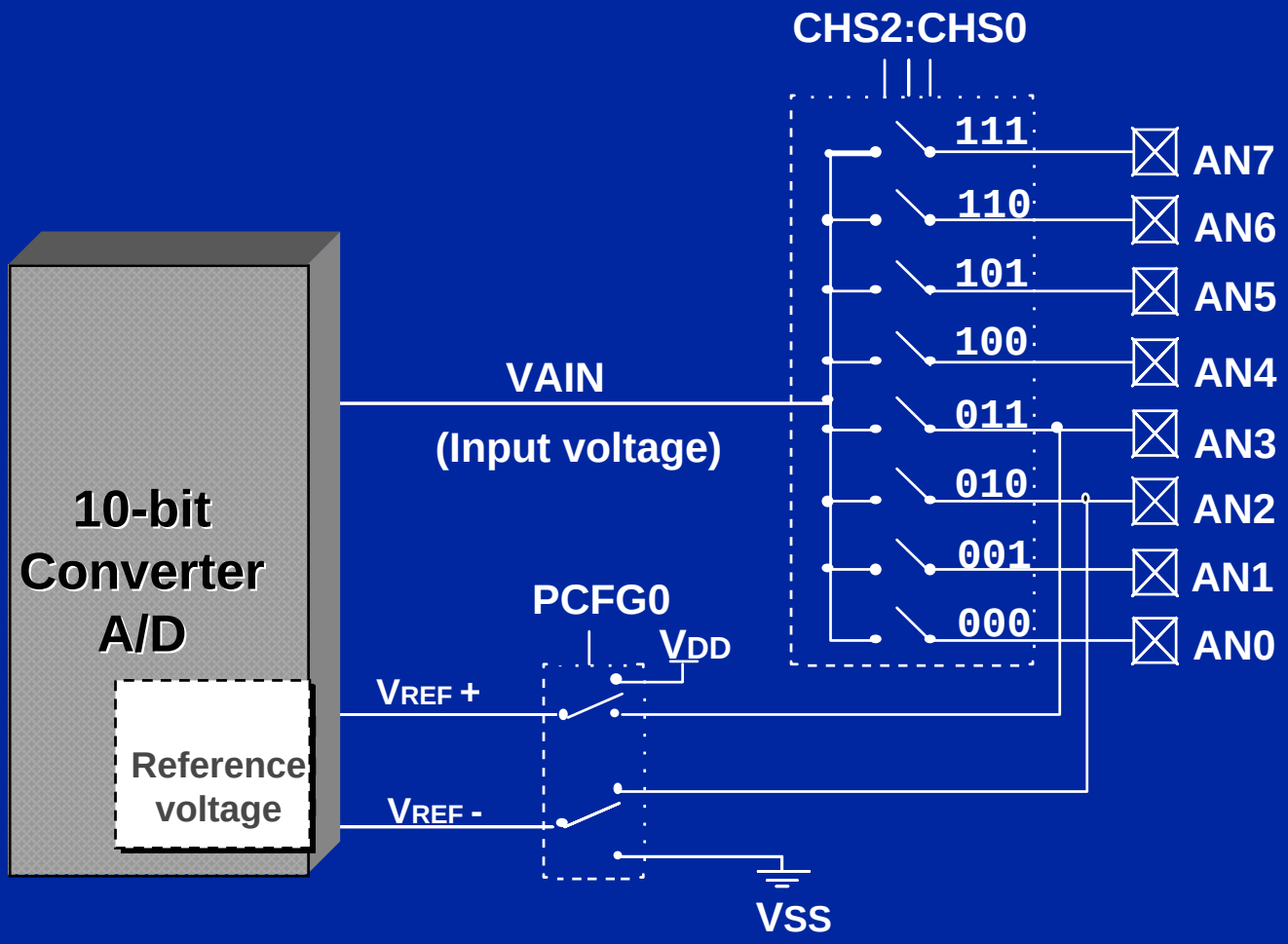


Note: On some devices this is a separate pin called a AVDD. This allows the A/D VDD to be connected to a precise voltage source

Peripherals
10-bit Analog-to-Digital (A/D) Converter

- 5 or 8 analog inputs multiplexed into one A/D converter
- 20 µs typical sampling time, as fast as 19.2 µs conversion time
- 10-bit resolution with ±1 LSb accuracy
- A/D conversion during SLEEP. A/D complete can wake processor
- External reference inputs, VREF+ & VREF-
- A/D Result can be left or right justified
- 39.2 µs for complete A/D result => A/D sampling at 25 kHz
  - 29.2 µs without channel change

Peripherals
10-Bit A/D Block Diagram



Lab 4: Simple A/D

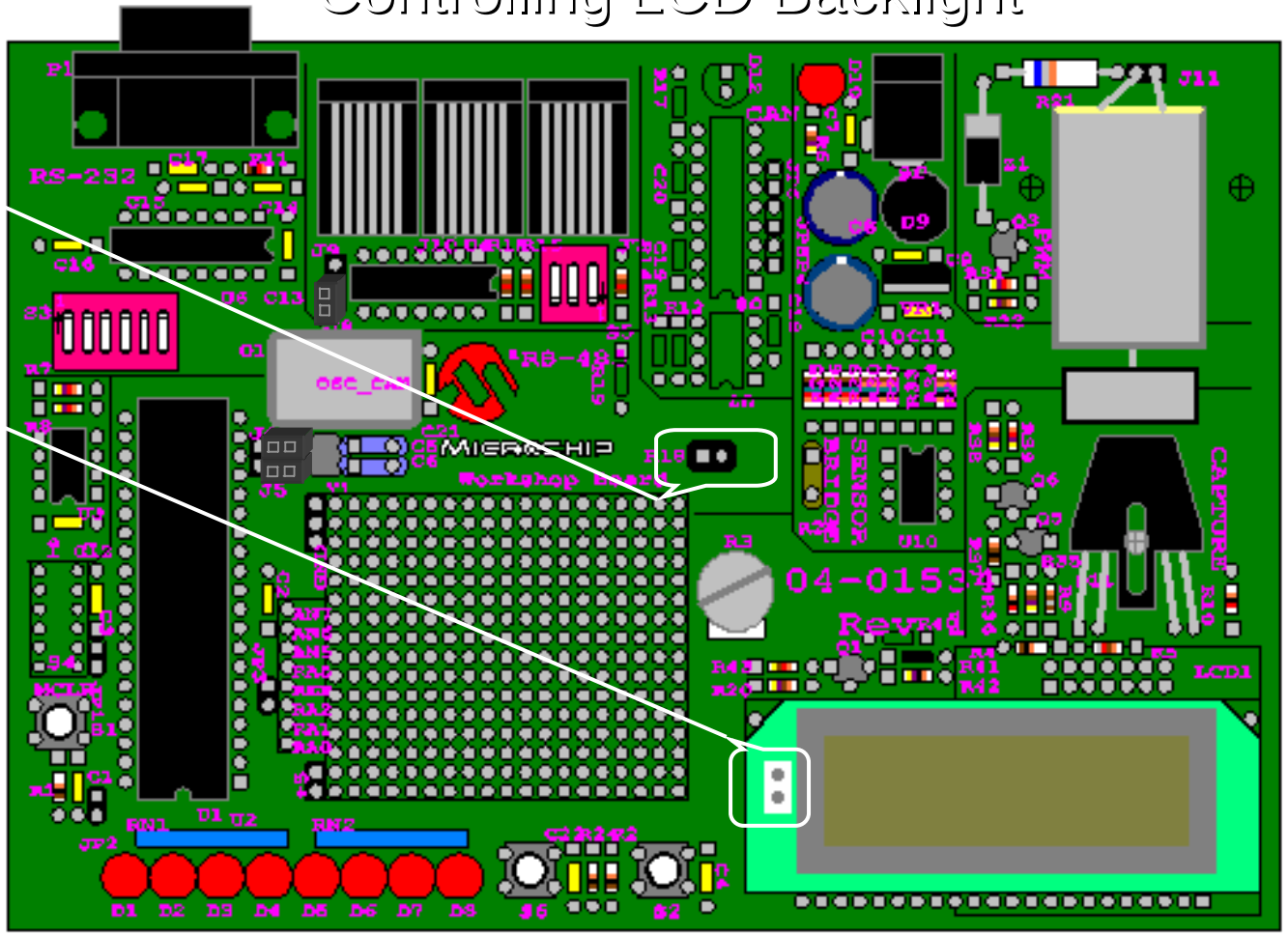
Controlling LCD Backlight

CdS Photocell on A/D ch2

LCD Backlight on RE0

RE0 = 1
Backlight OFF

RE0 = 0
Backlight ON



Continue with Lab 3

When ADRESH > X, turn Backlight OFF

When ADRESH < X, turn Backlight ON

Lab 4: Simple A/D

- Continue with Lab 3, use lab3.pjt & lab3.asm
- Configure A/D:
  - Fosc/8 clock, Channel 2, Left Justified
  - Configure A/D for 5/0 Chan/Refs (Figure 11-2)
- Check if Timer1 has overflowed
  - Update time variables & LCD Display
- If not, then do an A/D conversion
  - If ADRESH > X turn Backlight OFF
  - If ADRESH < X turn Backlight ON
  - Use debugger to find threshold

Lab 4: Goals

- Configuring the A/D converter
- Sampling an analog channel
- Using the result to control LCD backlight
- Multitasking events
  - Timer1 Overflow interrupt service routine
    - Update time variables & LCD Display
  - A/D Conversion
    - Control LCD Backlight
- Arithmetic comparison
- Linking object modules: lcd.o

Peripherals
12-bit Analog-to-Digital (A/D) Converter

- 6 or 10 analog inputs multiplexed into one A/D converter
- 11.5 µs typical sampling time, as fast as 20.8 µs conversion time
- 12-bit resolution with ±2 LSb INL & DNL
- A/D conversion during SLEEP. A/D complete can wake processor
- 8 Reference options
- A/D Result can be left or right justified
- 32.3 µs for complete A/D result => A/D sampling at 31 kHz
  - 25.8 µs without channel change

Peripherals
Voltage Reference

- Accurate to ± 2.5% over temperature
- Temp range -40C to 85C
- 2.048V (V_{REF+} and V_{REF-})
- 4.096V (V_{REF+} only)
- Can be brought out to external pins
- Can sink/source 5mA

Peripherals
Voltage Reference Configuration (12-bit A/D)

VCFG<2:0>	VREFH	VREFL
000	AVDD	AVSS
001	VREF+	VREF -
010	VRH	VRL
011	VREF +	AVSS
100	VRH	AVSS
101	AVDD	VREF -
110	AVDD	VRL
111	VRL	AVSS

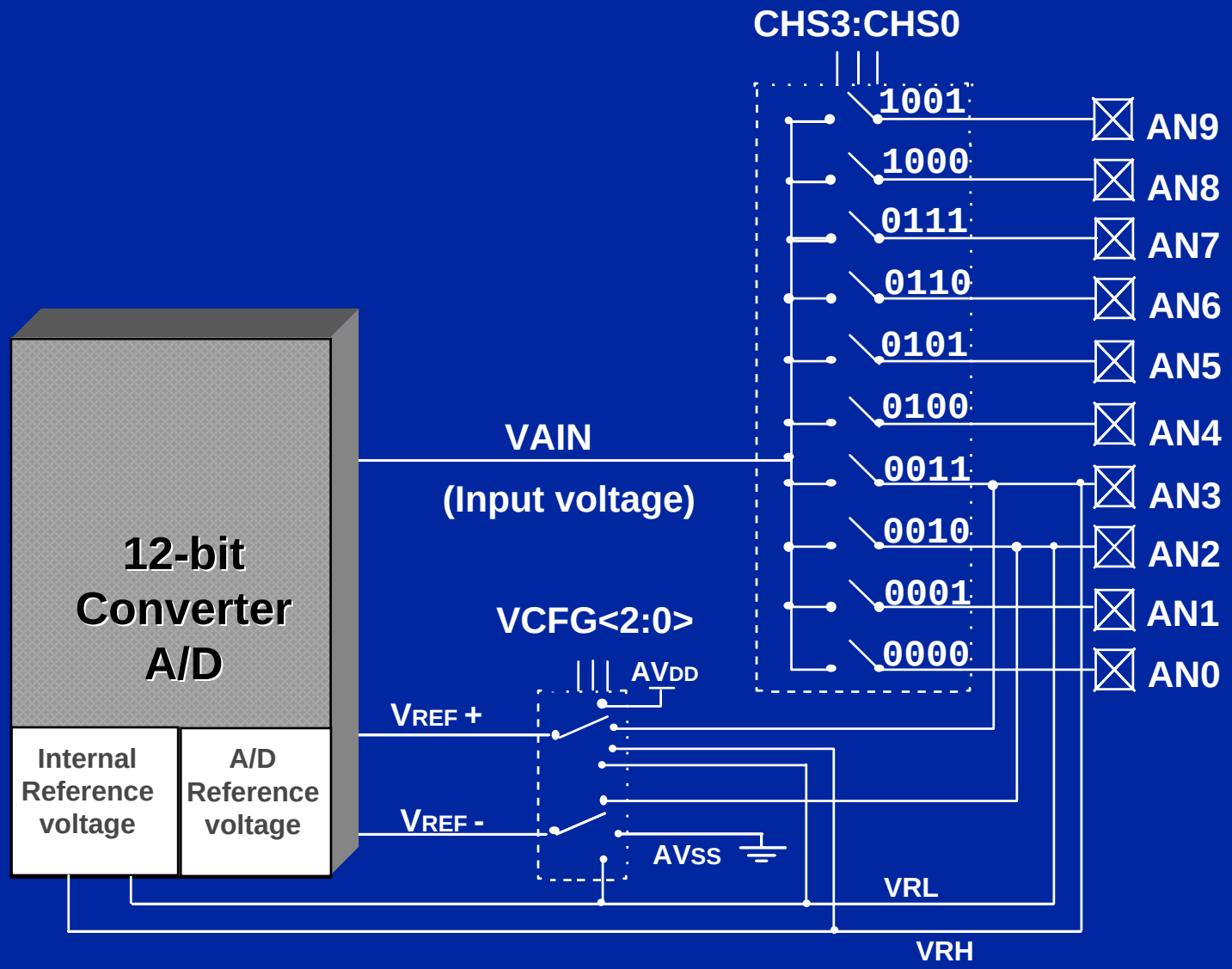
Peripherals
Voltage Reference Configurations

- Lowest Power: AVDD & AVSS
- 1mV per bit: VRH & AVSS
- 0.5mV per bit: VRL & AVSS or VRH & VRL

- Windowing:
  - VREF+ & VREF- VRH & AVSS
  - VREF+ & AVSS VRH & VRL
  - AVDD & VREF- VRL & AVSS

- For example: set $V_{REF}+ = V_{REF}-$
  - AVDD & AVSS : full range
  - AVDD & $V_{REF}-$: upper 1/2 of range
  - VREF+ & AVSS: lower 1/2 of range
  - VREF+ & VREF- : unusable

Peripherals
12-Bit A/D Block Diagram

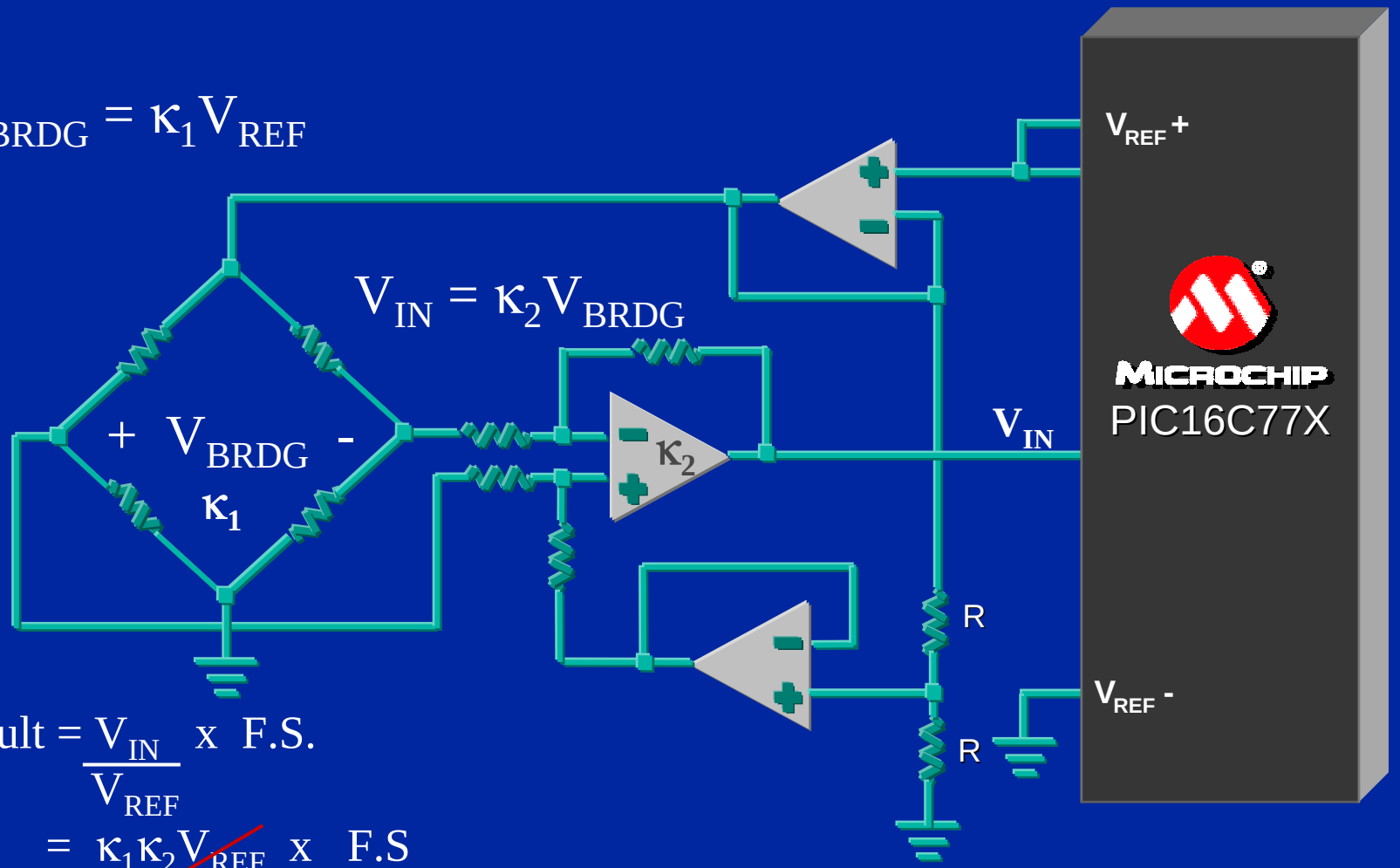


Peripherals
Ratiometric Sensing Using the A/D

- Ratiometric A/Ds use an external reference
- A/D result is a ratio of the input to this external reference
- By using a reference to the A/D that is also the excitation voltage to the sensor/analog signal conditioning circuit, remove effects from
  - Temperature drift
  - Reference drift
  - Power supply drift

Peripherals
Ratiometric Sensing Example

$$V_{BRDG} = \kappa_1 V_{REF}$$



$$\text{A/D Result} = \frac{V_{IN}}{V_{REF}} \times \text{F.S.}$$

$$= \frac{\kappa_1 \kappa_2 V_{REF}}{V_{REF}} \times \text{F.S}$$

$$= \kappa_1 \kappa_2 \times \text{F.S.}$$

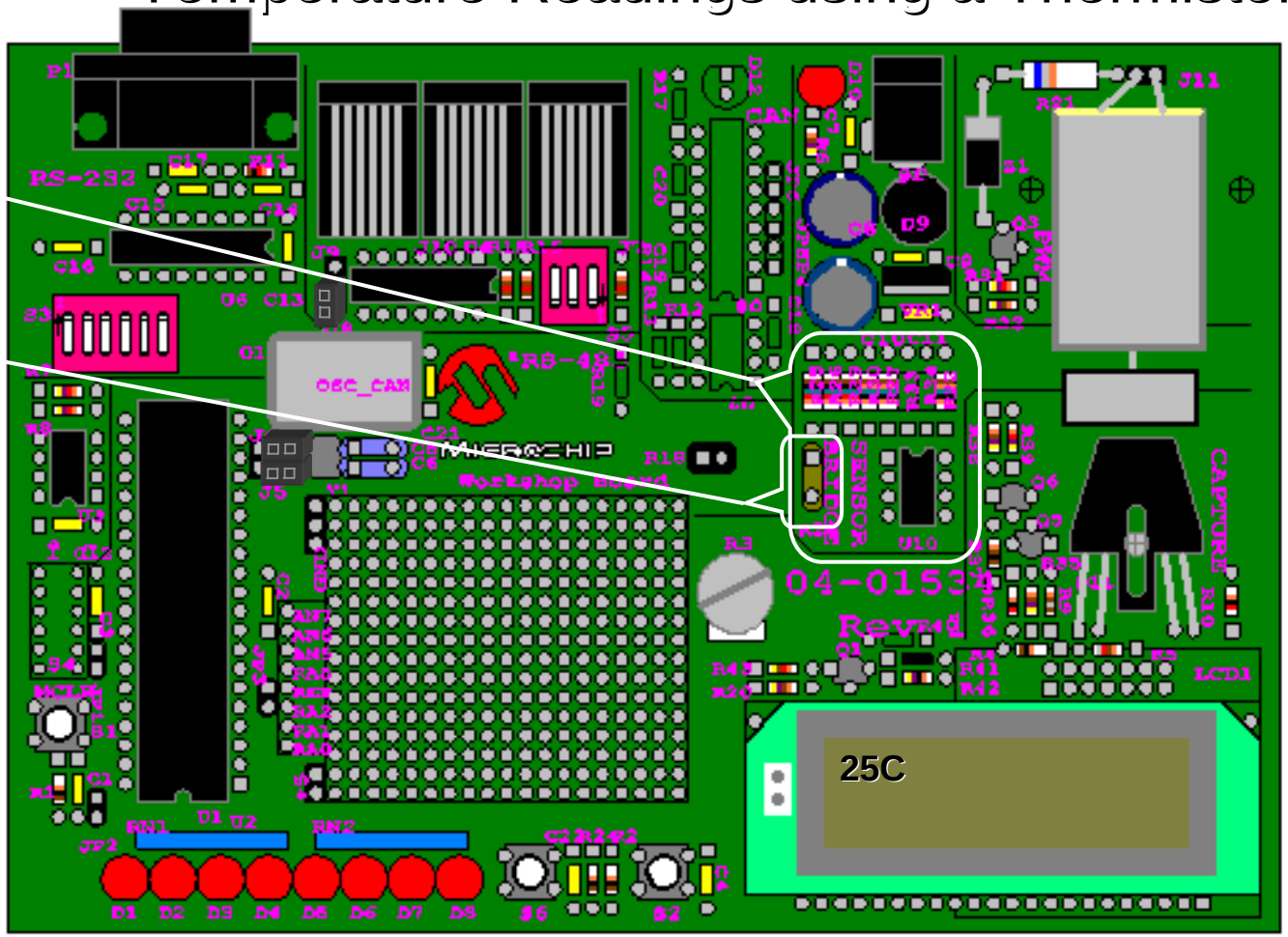
$$V_{REF} = V_{REF}^{+} - V_{REF}^{-}$$

Lab 5: Advanced A/D

Temperature Readings using a Thermistor

Bridge Circuit on A/D ch1

Thermistor



Delay for 20uS @ 4MHz
Convert on A/D channel 1

Send result to calibration routine
Convert calibrated value to ASCII
Update LCD Display

Lab 5: Advanced A/D

- Open project lab5.pjt
- Configure A/D:
  - Fosc/8 clock, Channel 1, Left Justified
  - Configure A/D for 3/0 Chan/Refs (Figure 11-2)
- Delay for 20us @ 4MHz
- Do an A/D conversion
- Calibrate the A/D result

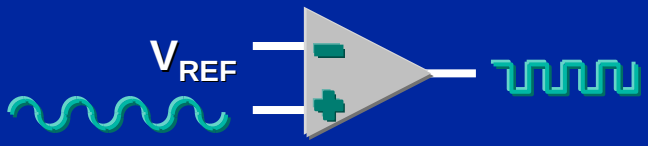
Lab 5: Advanced A/D

- Delay routine
  - Delay for 20 instruction cycles to get 20us
  - Use looping method to achieve delay
- Calibration parameters have been provided
  - Setup the parameters as a retlw table
  - Send the A/D result to the table in WREG
  - The calibrated temperature is returned in WREG

Lab 5: Goals

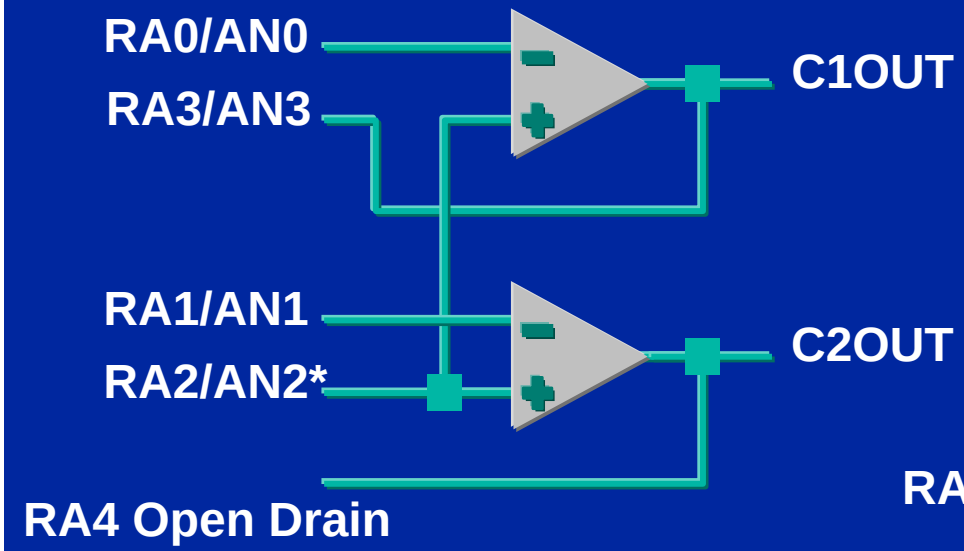
- Using the A/D in a ratiometric application
  - Thermistor in a bridge configuration
- Writing delay routines
  - Calculating the number of cycles needed
  - Calculating the number of cycles in looped code
- Using retlw tables for calibration
  - Sending the A/D result to the table
  - Returning the calibrated value
- Linking object modules
  - lcd.o

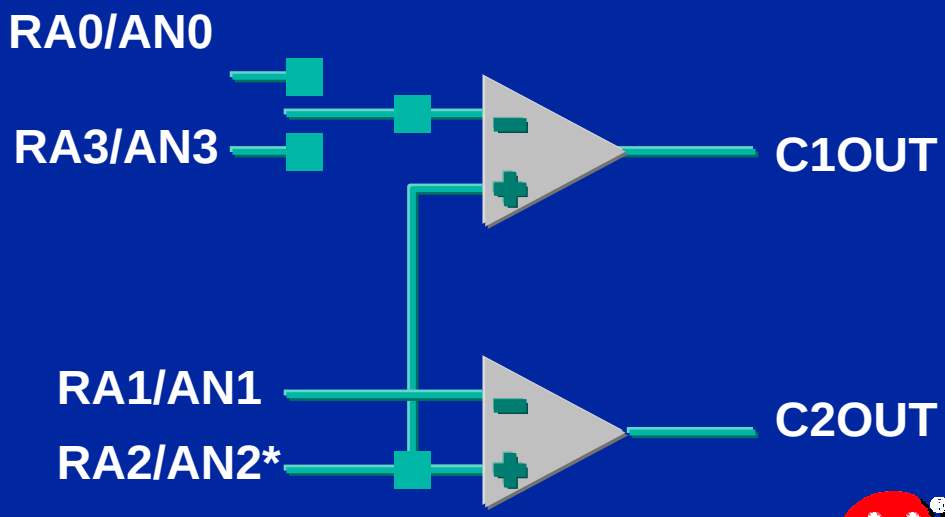
Peripherals
Analog Comparator Module



- 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 I/O multiplexed with digital I/O
- Comparator output pin available to drive I/O pins for hysterisis

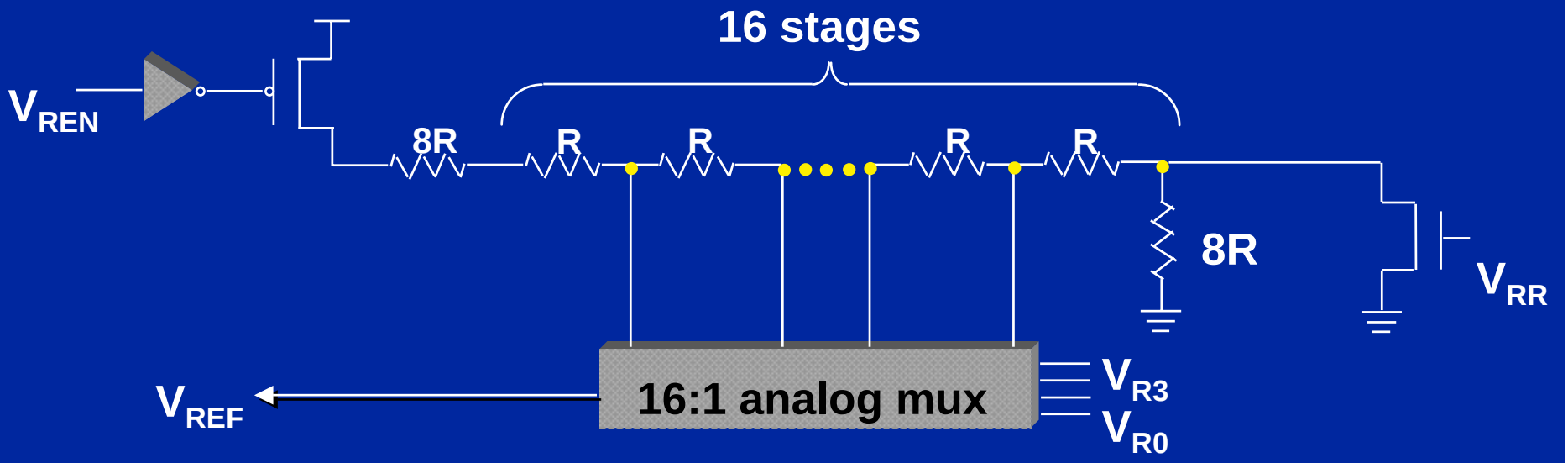
Peripherals
Analog Comparator Module (cont)





* Can be internally connected to V_{REF} Module

Peripherals
Internal VREF: Block Diagram



- VREF has 32 steps of output voltage
- VREN turns ON/OFF power to VREF circuit
- Can be used as a D/A converter
- VREF can be configured to an output pin

Peripherals
Capture / Compare / PWM (CCP) Module

- Can be configured as Input Capture, Output Compare, *or* PWM
- TMR1 used as timer for input capture and output compare modes
- TMR2 used for period and duty cycle time base in PWM mode

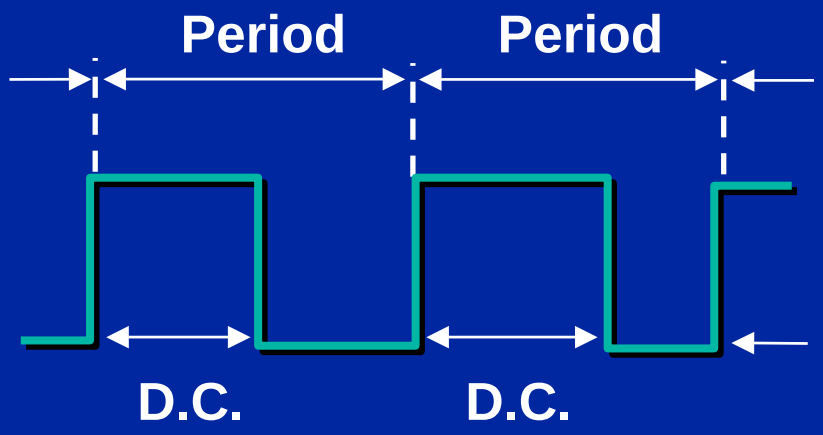
Peripherals
Capture / Compare / PWM (CCP) Module (cont)

- Definitions:
  - Input Capture: Capturing the time an event occurred (based on TMR1 as a free running timer) such as a low to high transition on an input pin.
  - Output Compare: Trigger an event based on a timer value matching a value in another register.
  - PWM (Pulse Width Modulation): A square wave of typically fixed frequency with varying duty cycle (time signal is high).

Peripherals
CCP Module: PWM Mode

- Up to 10-bit resolution:
  - 20 kHz frequency @ 10-bit resolution
  - 80 kHz frequency @ 8-bit resolution
  - 160 kHz frequency @ 7-bit resolution
- Up to 50 ns resolution (@ 20 MHz, high-resolution mode)

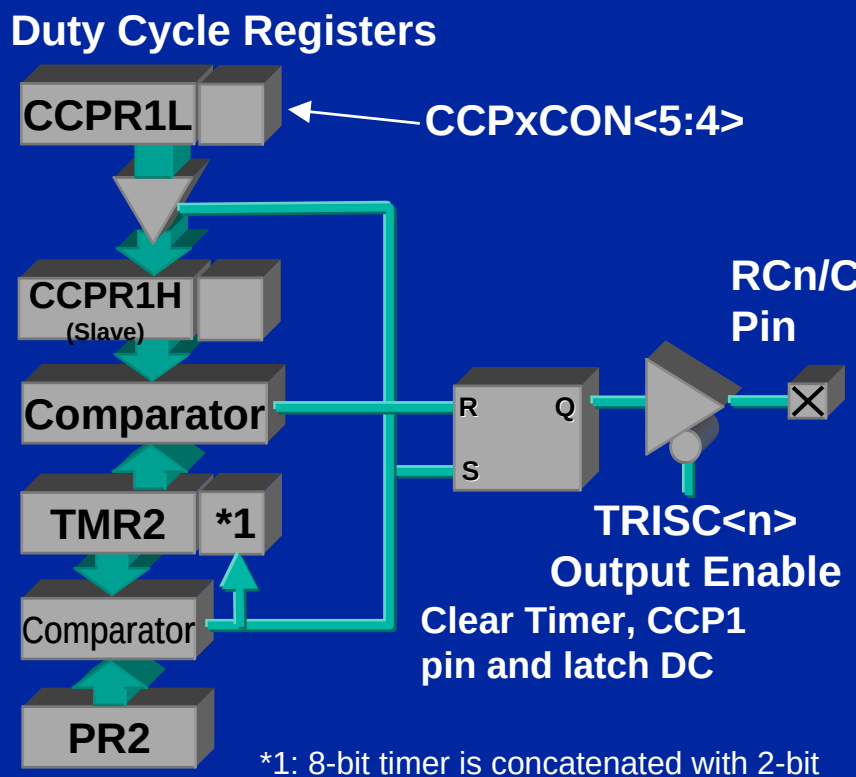
Peripherals
CCP Module: PWM Mode



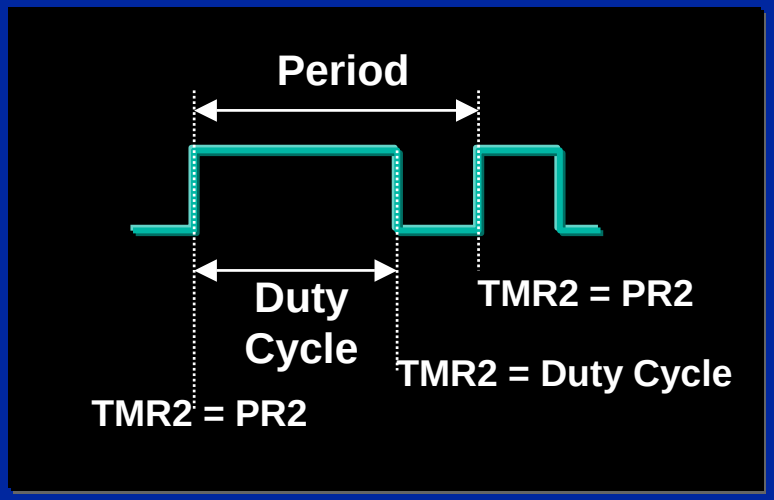
- At the end of a Period, the output goes high
- At the end of the Duty Cycle (D.C.), the output goes low
- At the end of the Period, a new Duty Cycle value is loaded

Peripherals
CCP Module: PWM Mode

Glitch free PWM with up to 50ns resolution



*1: 8-bit timer is concatenated with 2-bit internal Q clock or two bits of the prescaler to create a 10-bit time base.



Resolution	Maximum Frequency
10-bits	20kHz
8-bits	80kHz
7-bits	156kHz
5.5-bits	208kHz

f_{osc} = 20MHz

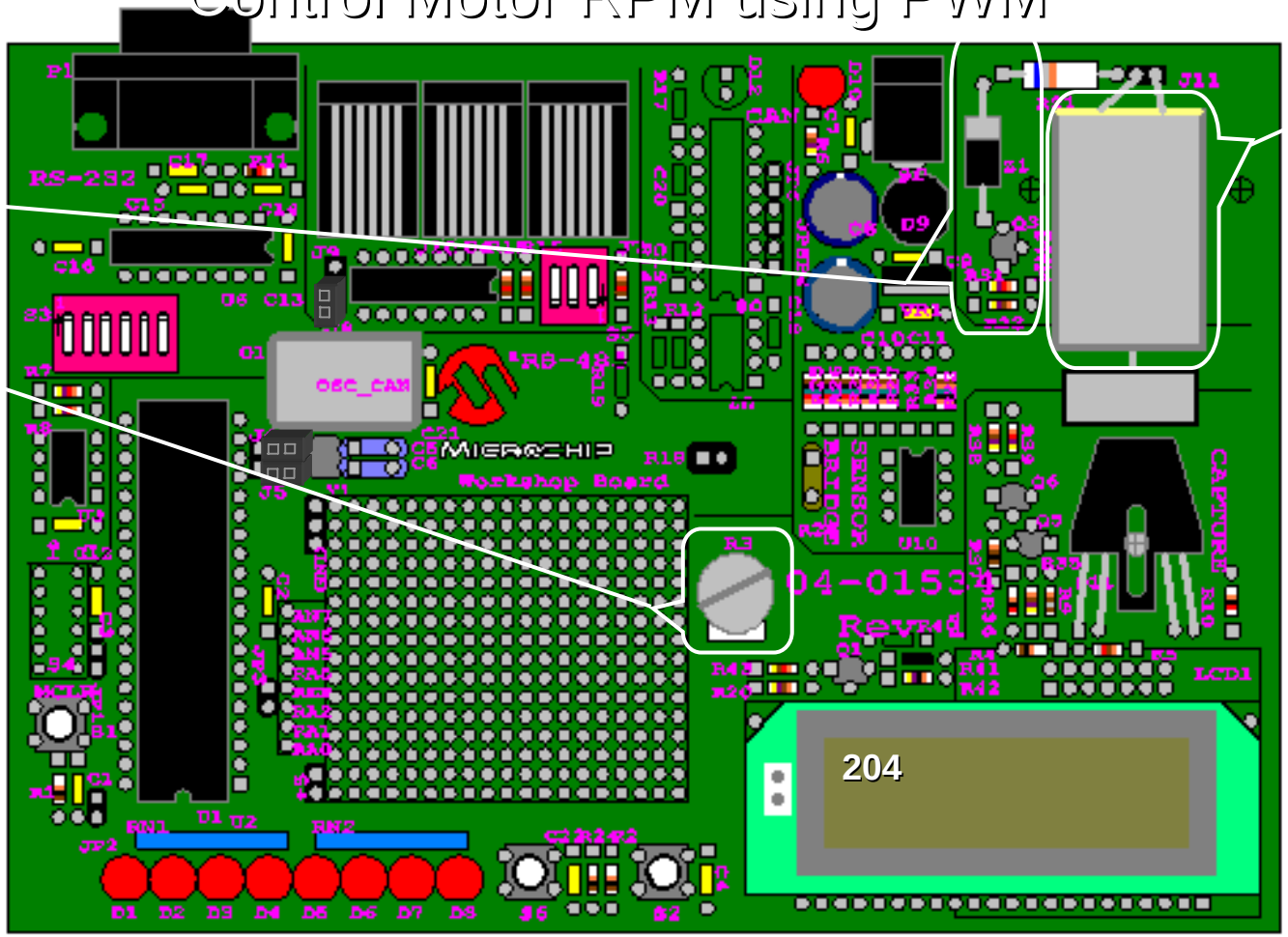
Lab 6: Motor Control

Control Motor RPM using PWM

Motor Driver Circuit using PWM

RPM Control

DC Motor



Delay for 20uS @ 4MHz

Convert on A/D channel 0

Use A/D result as PWM Duty Cycle & Display on LCD

MICROCHIP

Lab 6: PWM Motor Control

- Open project lab6.pjt
- Configure A/D:
  - Fosc/8 clock, Channel 0, Left Justified
  - Configure A/D for 1/0 Chan/Refs (Figure 11-2)
- Configure Timer2:
  - 1:1 Prescale & Postscale, PR2 = 0xFF
- Configure CCP1:
  - PWM Mode
  - Duty Cycle = A/D result (CCPR1L = ADRESH)
- Make RC2 an output

Lab 6: PWM Motor Control

- Delay for 20us
- Take an A/D conversion
- Write A/D result to PWM duty cycle

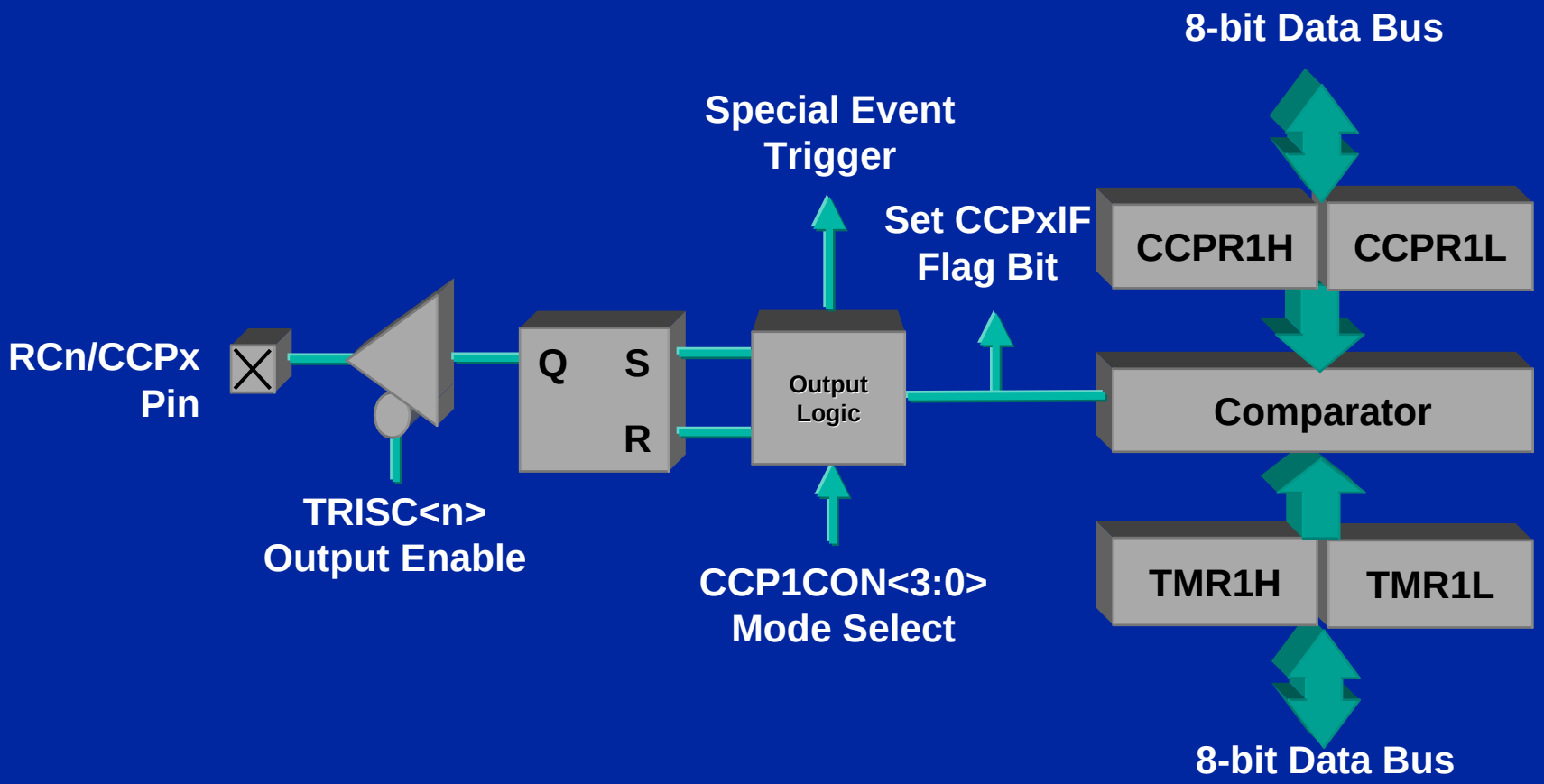
Lab 6: Goals

- Configure CCP Module
  - Using PWM
  - Using Timer2
- Changing the duty cycle of the PWM
- Controlling a DC motor

Peripherals
CCP Module: Output Compare Mode

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

Peripherals
CCP Module: Output Compare Mode (cont)

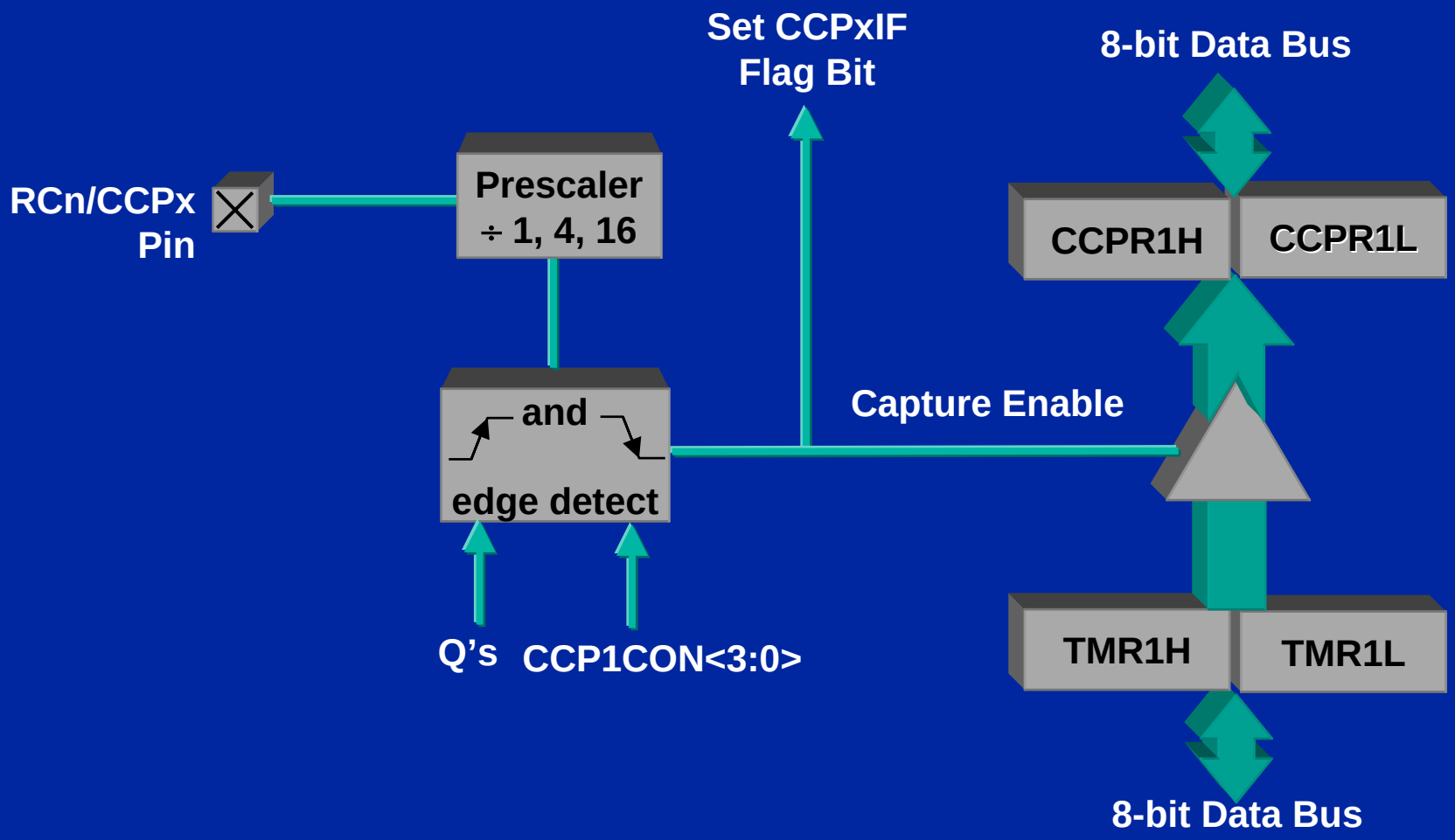


Peripherals
CCP Module: Input Capture Mode

- 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

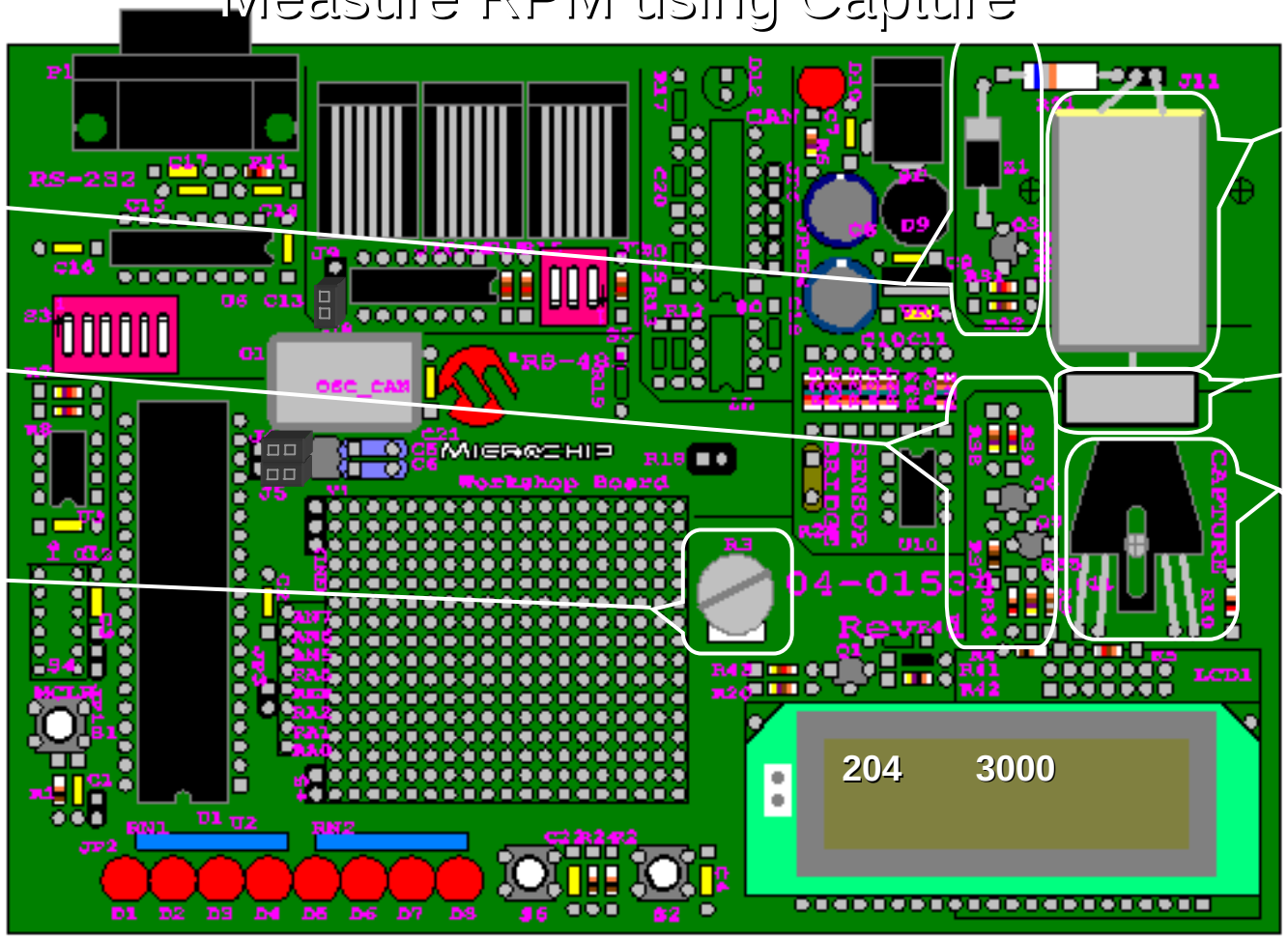
Peripherals

CCP Module: Input Capture Mode (cont)



Lab 7: RPM Measurement

Measure RPM using Capture



Continue with Lab 6

Use capture to measure pulse train from slotted wheel

Use A/D result as PWM Duty Cycle & Display on LCD

Also Display RPM on LCD

Lab 7: RPM Measurement

- Continue with Lab 6, use lab6.pjt & lab6.asm
- Add the file math.o to the project
- Configure Timer1:
  - 1:1 Prescale, Oscillator OFF, Internal Clock
- Configure CCP2
  - Capture, every rising edge
  - Enable CCP2 Interrupt
  - Enable PEIE and GIE

Lab 7: RPM measurement

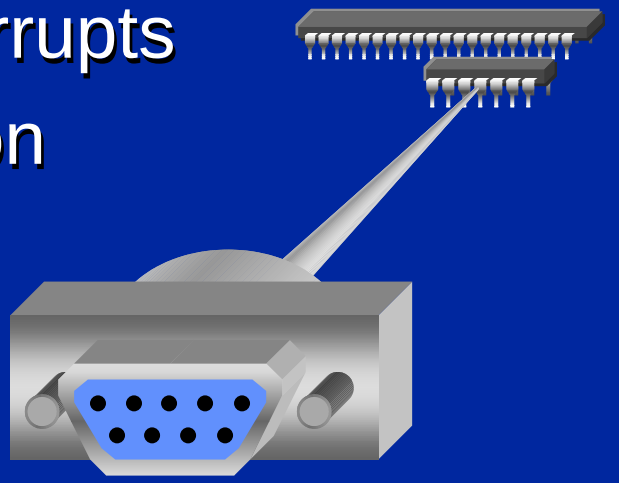
- Main routine
  - A/D conversions to determine PWM1 duty cycle
  - Load AARG with 01C9C380h
  - Load BARG with CapH:CapL
  - Call FXD3216U to get AARG/BARG result
- Write Capture2 Interrupt Service Routine
  - Clear Timer1 (TMR1H:TMR1L)
  - Save capture value in CapH:CapL
  - Clear interrupt flags

Lab 7: Goals

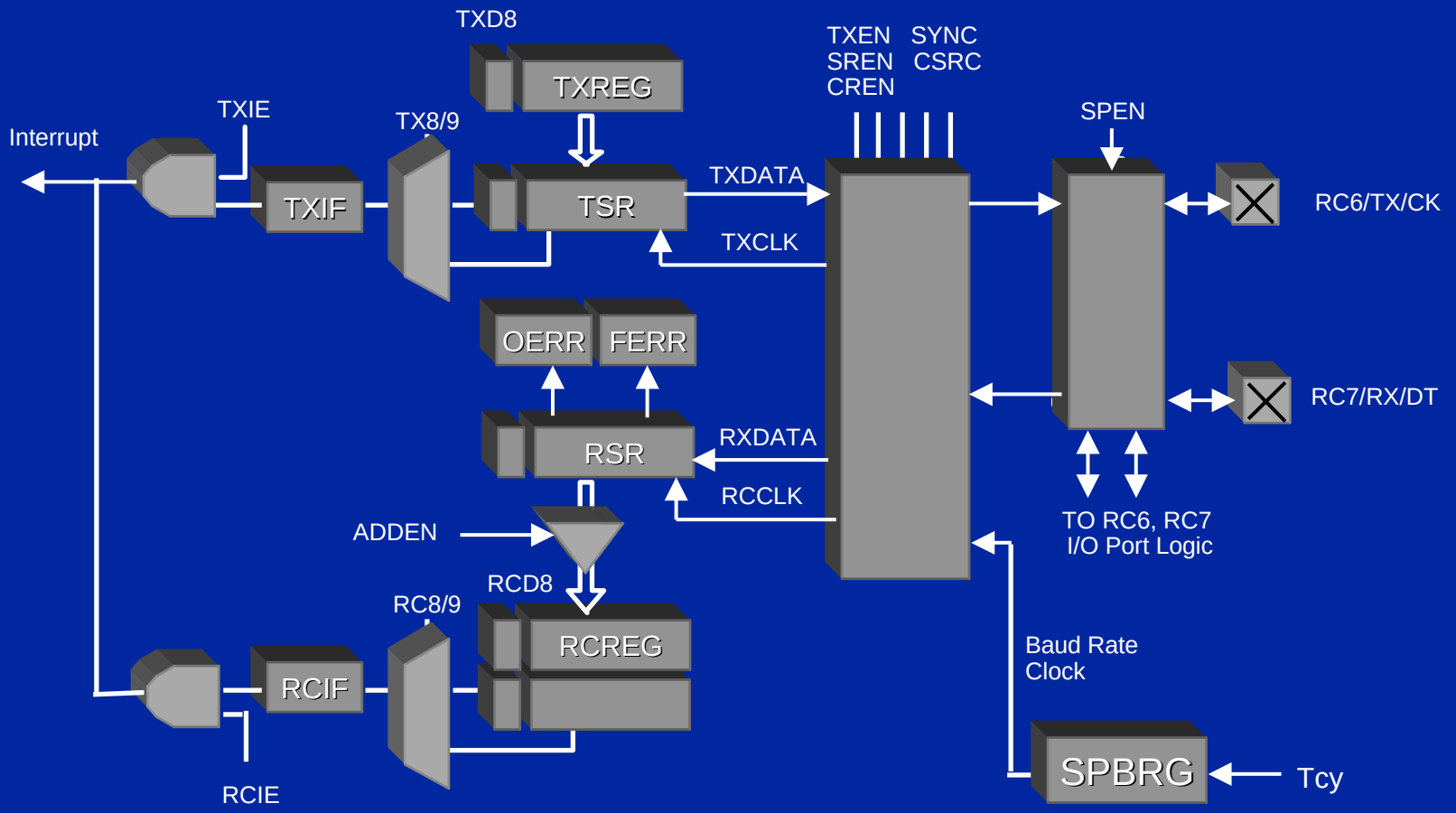
- Configuring the capture module
- Using & Servicing interrupts
- Math routines
  - Divide
- Linking object modules
  - lcd.o
  - math.o

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 baud rate generator
- Max baud rates @ 20MHz
  - Synchronous: 5 Mbaud
  - Asynchronous: 312.5 Kbaud / 1.25 Mbaud
- 9 bit addressable mode

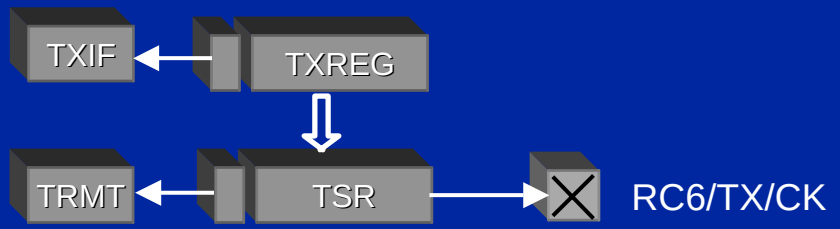


Peripherals
USART Block Diagram



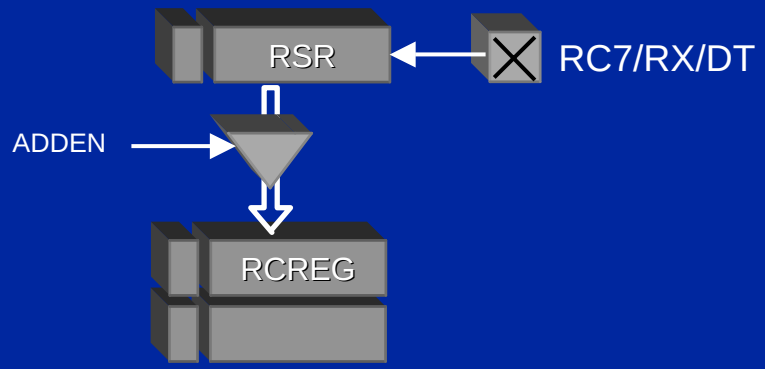
Peripherals
TXIF and TRMT Operation

- If TXREG is empty, then TXIF = 1
- Loading TXREG causes TXIF = 0
- If TSR is empty then TRMT = 1
- Loading TSR causes TRMT = 0
- IF TXREG is loaded and TRMT = 1, then data is immediately loaded to TSR. Serial data shifting starts and TXIF = 1.



Peripherals
RCIF Operation

- RSR receives 8 bits of data with valid start/stop.
- Data loaded into RCREG FIFO and RCIF set
- 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

Peripherals
RS-485 Description

- RS-485 used in noisy environments, multi-drop
  - RS-232 w/ differential drive on TXD & RXD
- Half-duplex systems use 1 pair of lines
- Full-duplex systems use 2 pair of lines
- Data with 9th bit set considered a command or address byte
  - Only the slave with this address should receive the following data bytes
- Data bytes have the 9th bit cleared

Peripherals
RS-485 Description

- ADDEN simply tells the USART to ignore data with the 9th bit clear
  - The byte is not transferred to the receive fifo
  - The interrupt flag is not set
- When transmitting, the firmware must set/clear the 9th bit appropriately

Lab 8: RS-485 Network

Using the 9-bit Addressable USART

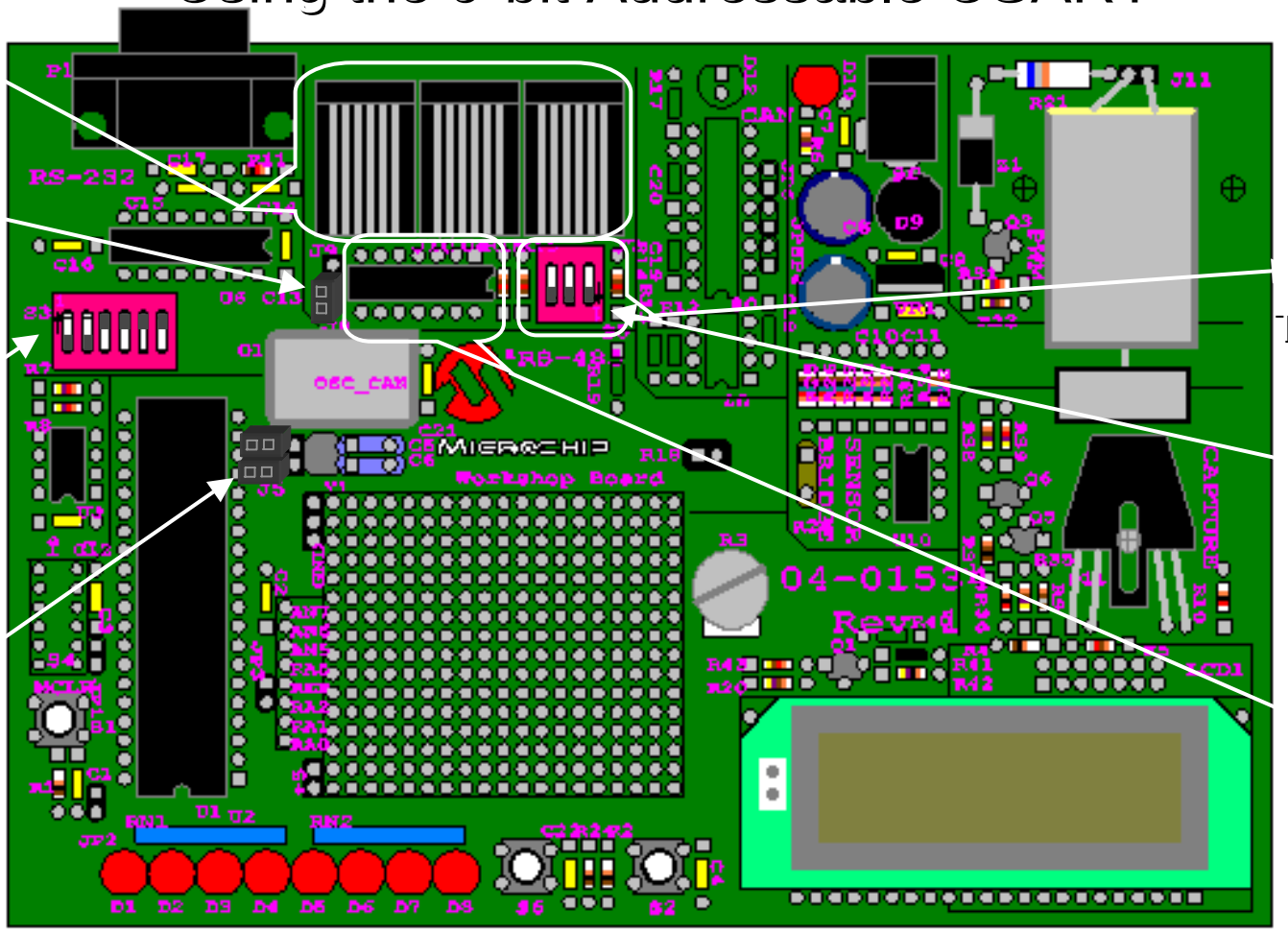
Modular Cable Network Interface

Install Jumper

Set DIP Switches

Install Jumpers

Use 9600 Baud



RS-485 Network Terminations

Set DIP Switches

RS-485 Driver TX Enable on RC0

Wait for your address
- 9th bit is set
- 8 data bits match

Clear ADDEN to Receive next byte
- 9th bit is clear

Respond:

Write data byte to PORTD
- Send Master address 0x00
- Send data byte

Lab 8: RS-485 Network

- Open project lab8.pjt
- Configure USART:
  - Asynchronous, 9-bit transmit/receive, 9600 Baud
  - Address detect enabled, 9600 baud
  - Continuous receive, High speed mode
- RC0 controls transmit enable on RS485 chip
  - Leave low until you transmit, then set high
  - Don’t forget to set back low after transmission is complete

Lab 8: RS-485 Network

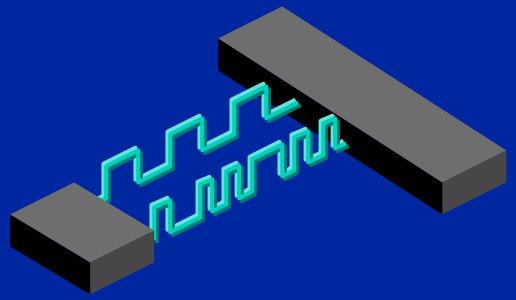
- Wait until you receive a byte that matches your address
  - Disable address detect to get the next data byte
- Write data byte to PORTD LEDs
- Send a packet to the master
  - Set the 9th data bit, set RC0 to enable transmit
  - Transmit master's address: 0x00
  - Clear the 9th data bit
  - Transmit the data byte
  - Wait for transmit to complete, clear RC0
  - Enable address detect

Lab 8: Goals

- Learn RS-485 protocol
  - Commands have 9th bit set
  - Data has 9th bit clear
- Configure the USART
  - Use address detect circuitry
- Calculate baud rates for oscillator frequencies
- Using MPASM in absolute mode
  - No relocatable code

Peripherals
Synchronous Serial Port (SSP)

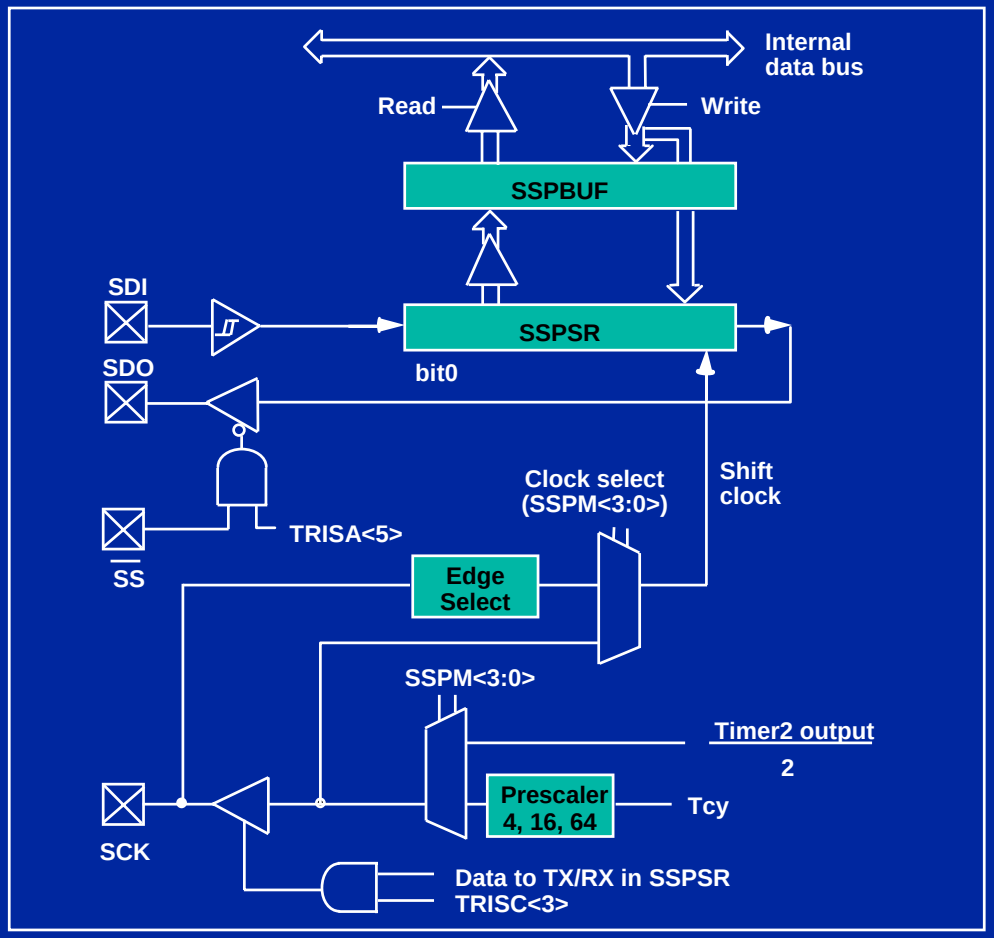
- Operates in either SPI™ or I²C™ mode
- SPI Mode
  - Programmable baud rate
  - Maximum baud rates (@ 20MHz)
    - Master: 5 Mbaud
    - Slave: 1.25 Mbaud Single Byte Tx
  - Programmable clock polarity for transmit / receive
  - All four SPI modes supported
- I²C Mode
  - Supports standard (100kHz) and fast (400kHz) I²C standards



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

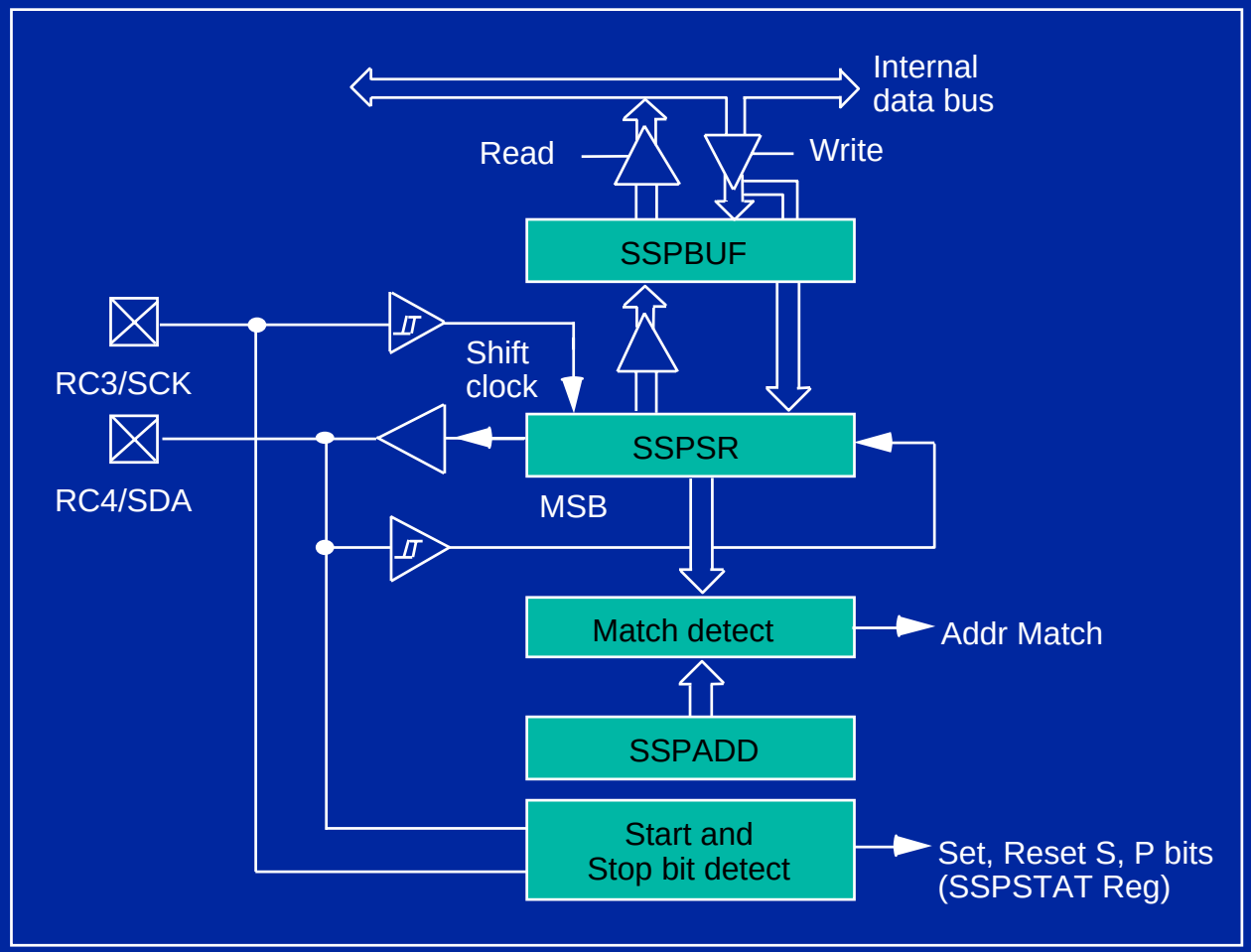
Peripherals
SSP Module Block Diagram

SPI
Mode

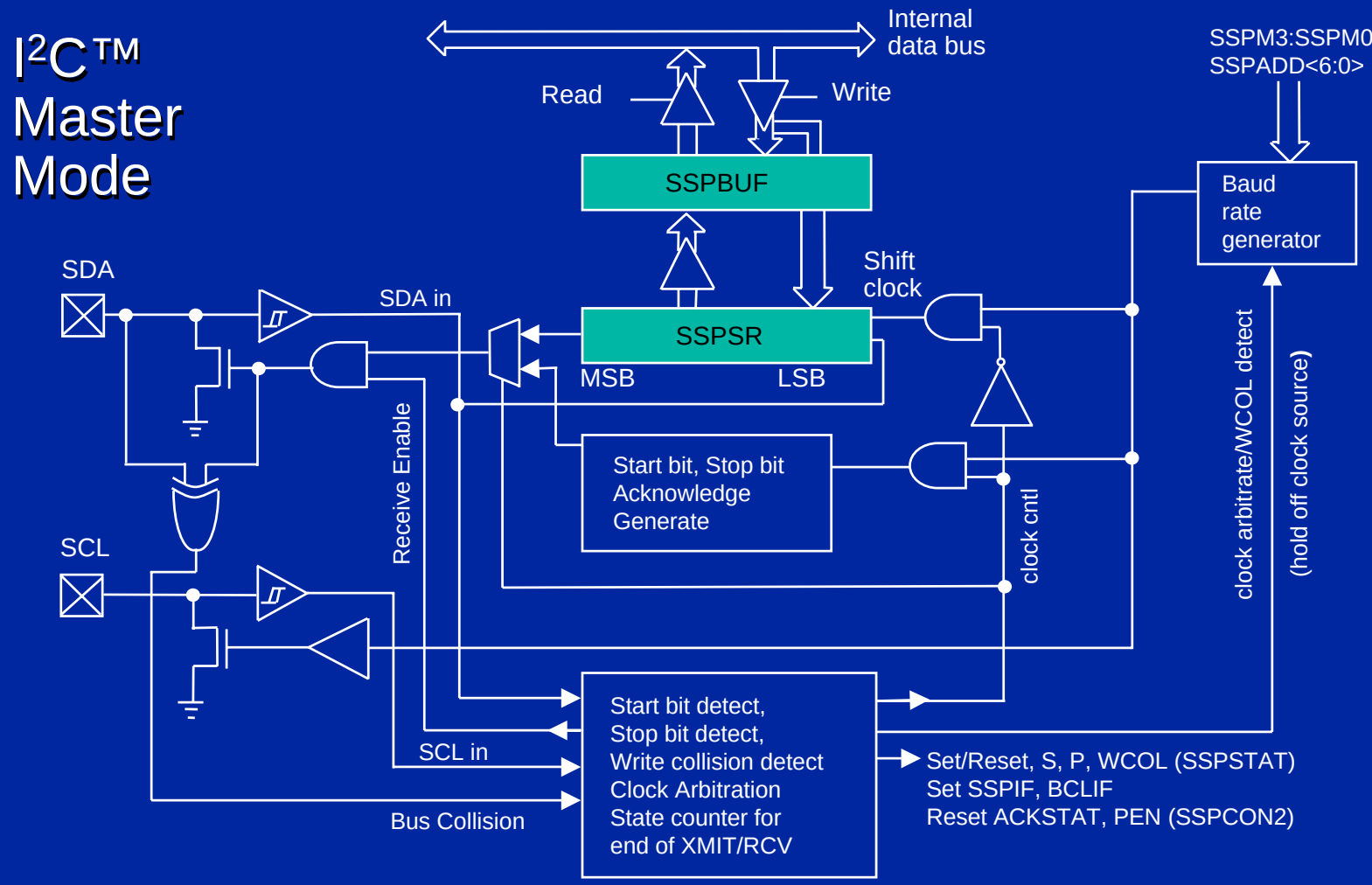


Peripherals
SSP Module Block Diagrams

I²C™ Slave Mode



Peripherals
SSP Module Block Diagrams



Peripherals
Integrated EEPROM Data Memory

- Up to 256 bytes non-volatile bytes
- 1M E/W cycle endurance (typical)
- EEPROM write complete interrupt
- Full V_{DD} range data memory programming
- EEPROM data retention > 40 years
- Available on “F” devices: e.g. PIC16F84A, PIC16F877

Peripherals
Dual Die EEPROM Data Memory

- From 16 to 256 bytes available
- Read/Written to like Serial EEPROM
- 1M E/W cycle endurance (typical)
- Full VDD range data memory programming
- EEPROM data retention > 40 years
- Available on “CE” devices: e.g. PIC12CE518

Peripherals
EEPROM/FLASH Registers

- EEDATA (10Ch)
  - Holds data for Data EEPROM/Program FLASH
- EEADR (10Dh)
  - Holds addr for Data EEPROM/Program FLASH
- EECON1 (18Ch), New Bit EEPGD
  - EEPROM Read/Write Control Register
- EECON2 (18Dh)
  - Not a physically implemented register
  - reads all ‘0’s
  - Used exclusively for memory writes

Peripherals
Reading Internal Data EEPROM

- Write the desired address to EEADR
- Clear the EEPGD bit, EECON1<7>
- Set the RD bit, EECON1<0>
- Data will be available in the EEDATA register in the next instruction cycle

Peripherals
Reading Internal Data EEPROM

Move to Bank2	`BSF STATUS,RP1`
	`BCF STATUS,RP0`
Write data memory addr	`MOVLW ADDRESS`
to EEADR register	`MOVWF EEADR`
Move to Bank3	`BSF STATUS,RP0`
Point to Data Memory	`BCF EECON1,EEPGD`
Start Read Operation,	`BSF EECON1,RD`
data available in next Tcy	`BCF STATUS,RP0`
Move the data into WREG	`MOVF EEDATA,W`

MICROCHIP

Peripherals
Writing Internal Data EEPROM

- Write the desired address to EEADR
- Write the desired data to EEDATA
- Clear the EEPGD bit, EECON1<7>
- Set the WREN bit, EECON1<2>
- Disable all interrupts
- Write 55h to EECON2
- Write AAh to EECON2
- Set the WR bit, EECON1<1>
- Clear the WREN bit, EECON1<2>
- Wait for WR to clear or EEIF to set, indicates write operation has completed

Peripherals
Writing Internal Data EEPROM

Move to Bank2	`BSF STATUS,RP1` `BCF STATUS,RP0`
Write desired address to EEADR	`MOVLW ADDRESS;MOVWF EEADR`
Write desired data to EEDATA	`MOVLW VALUE; MOVWF EEDATA`
Move to Bank3	`BSF STATUS,RP0`
Enable EEPROM writes	`BSF EECON1,WREN`
Disable interrupts	`BCF INTCON,GIE`

Next five instructions are required sequence to initiate a write sequence	`MOVLW 55h`
	`MOVWF EECON2`
	`MOVLW AAh`
	`MOVWF EECON2`

Initiate the write sequence	`BSF EECON1,WR`
Enable interrupts	`BSF INTCON,GIE`
Disable EEPROM writes, does not affect the current write cycle	`BCF EECON1,WREN`

Peripherals
FLASH Program Memory

- Up to 8K Words Flash Program Memory
- Device can read/write itself
- Programmable over entire operating voltage range
- Industry's lowest FLASH MCU operating voltage at 2.0V

Peripherals
New EEPROM/FLASH Registers

- EEDATH (10Eh)
  - Used exclusively for Program Memory operations
  - Holds the 6 MSbs of the data value
- EEADRH (10Fh)
  - Used exclusively for Program Memory operations
  - Holds the 5 MSbs of the address
- **The PIC16F87X devices can now calculate program memory checksums!!!!!**

Peripherals
Reading Internal Program FLASH

- Write the 8 LSbs of the desired address to EEADR
- Write the 5 MSbs of the desired address to EEADRH
- Set the EEPGD bit, EECON1<7>
- Set the RD bit, EECON1<0>
- Next two instructions ignored while the CPU reads program memory
- Data will be available in the EEDATH:EEDATA registers in the next instruction cycle

MICROCHIP

Peripherals
Reading Internal Program FLASH

Move to Bank2

Write data memory address into the EEADRH:EEADR register

Move to Bank3
Point to Data Memory
Start the Read Operation, data will be available in the third T_{CY}

Move the data into WREG

```
BSF    STATUS,RP1
BCF    STATUS,RP0
MOVLW  ADDRH
MOVWF  EEADRH
MOVLW  ADDRL
MOVWF  EEADR
BSF    STATUS,RP0
BCF    EECON1,EEPGD
BSF    EECON1,RD
NOP
NOP
BCF    STATUS,RP0
MOVF   EEDATA,W
MOVF   EEDATH,W
```

Peripherals
Writing Internal Program FLASH

- Write the desired address to EEADRH:EEADRL
- Write the desired data to EEDATH:EEDATA
- Set the EEPGD bit, EECON1<7>
- Set the WREN bit, EECON1<2>
- Disable all interrupts
- Write 55h to EECON2
- Write AAh to EECON2
- Set the WR bit, EECON1<1>
- Next two instructions ignored
- CPU now halts while memory is programmed, this is NOT sleep mode as the clocks and peripherals continue to run
- Clear the WREN bit, EECON1<2>

Peripherals
Writing Internal Program FLASH

Move to Bank2	`BSF STATUS,RP1` `BCF STATUS,RP0`
Write desired address to EEADRH:EEADR	`MOVLW ADDRH; MOVWF EEADRH` `MOVLW ADDRL; MOVWF EEADR`
Write desired data to EEDATH:EEDATA	`MOVLW DATAH; MOVWF EEDATH` `MOVLW DATAL; MOVWF EEDATA`
Move to Bank3	`BSF STATUS,RP0`
Enable EEPROM writes	`BSF EECON1,WREN`
Disable interrupts	`BCF INTCON,GIE`

Next five instructions are required sequence to initiate a write sequence	`MOVLW 55h`
	`MOVWF EECON2`
	`MOVLW AAh`
	`MOVWF EECON2`

Initiate the write sequence	`BSF EECON1,WR`
Instruction is ignored, processor halts	`NOP`
Enable interrupts	`NOP` `BSF INTCON,GIE`
Disable EEPROM writes	`BCF EECON1,WREN`

Peripherals
Internally Accessing FLASH Program Memory

Configuration Bits			Memory Code Protected	Memory Location	Internal Read	Internal Write
CP1	CP0	WRT				
0	0	X	All	All Program Memory	Yes	No
0	1	0	Upper 1/2	Unprotected Areas	Yes	No
0	1	0	Upper 1/2	Protected Areas	Yes	No
0	1	1	Upper 1/2	Unprotected Areas	Yes	Yes
0	1	1	Upper 1/2	Protected Areas	Yes	No
1	0	0	Upper 256	Unprotected Areas	Yes	No
1	0	0	Upper 256	Protected Areas	Yes	No
1	0	1	Upper 256	Unprotected Areas	Yes	Yes
1	0	1	Upper 256	Protected Areas	Yes	No
1	1	0	None	All Program Memory	Yes	No
1	1	1	None	All Program Memory	Yes	Yes

Always perform Read Operations
No Write Operations when WRT = 0
No Write Operations in code protected memory areas

Peripherals
Accessing FLASH Program Memory using ICSP

Configuration Bits			Memory Code Protected	Memory Location	ICSP Read	ICSP Write
CP1	CP0	WRT				
0	0	X	All	All Program Memory	No	No
0	1	0	Upper 1/2	Unprotected Areas	Yes	No
0	1	0	Upper 1/2	Protected Areas	No	No
0	1	1	Upper 1/2	Unprotected Areas	Yes	No
0	1	1	Upper 1/2	Protected Areas	No	No
1	0	0	Upper 256	Unprotected Areas	Yes	No
1	0	0	Upper 256	Protected Areas	No	No
1	0	1	Upper 256	Unprotected Areas	Yes	No
1	0	1	Upper 256	Protected Areas	No	No
1	1	0	None	All Program Memory	Yes	Yes
1	1	1	None	All Program Memory	Yes	Yes

No Read Operations from Code Protected areas

No Write Operations **anywhere** when Code Protected

WRT has no effect on ICSP Read/Write Operations

No Write Operations when Data Memory Code Protected

MICROCHIP

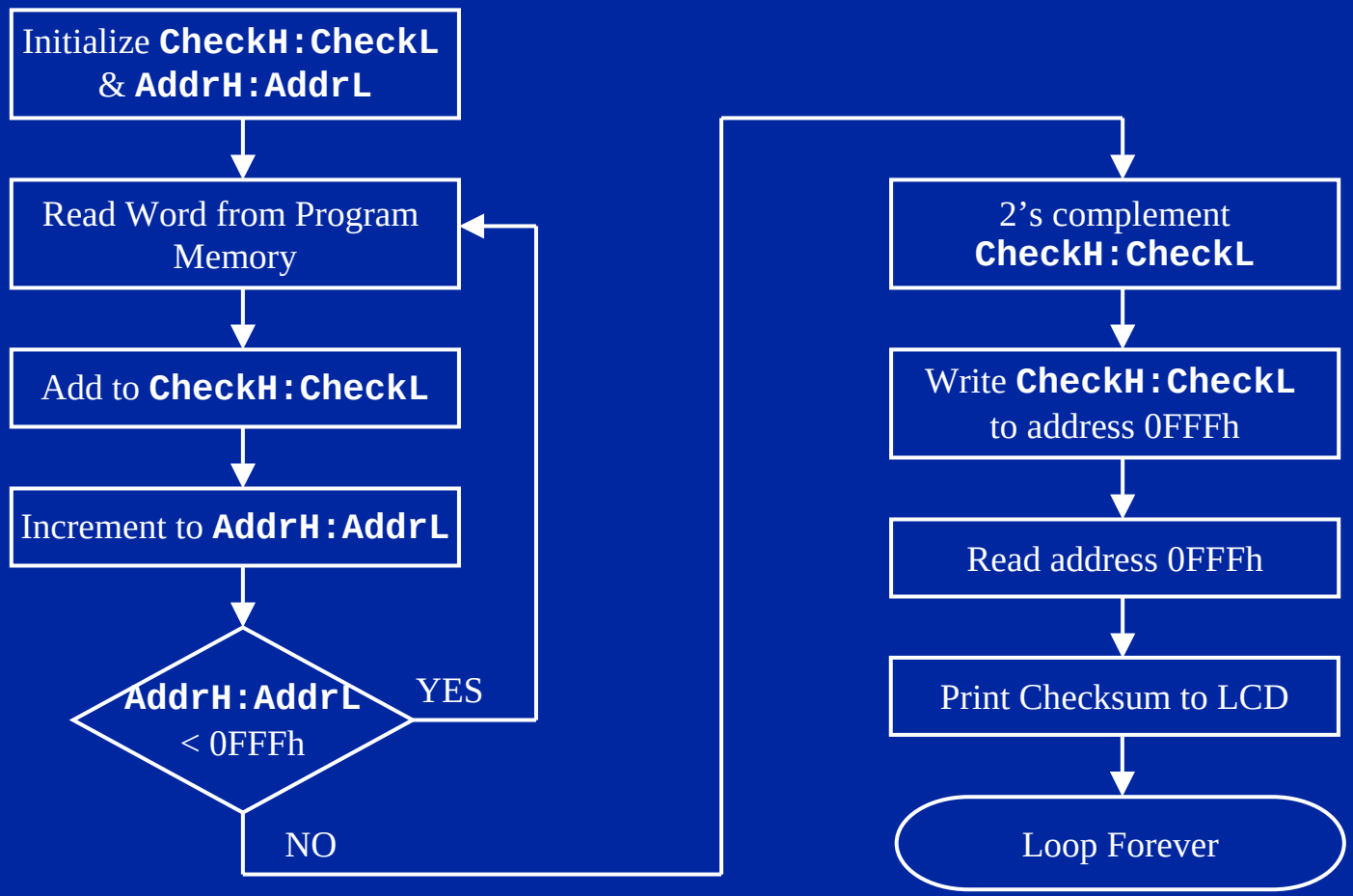
Peripherals
Read/Write Operations Over Voltage

- Internal Read/Write Operations
  - Can always read any locations at any voltage
  - When writes can occur, can write at any voltage
- External Read/Write Operations
  - When able, can read from any location over voltage
  - When able, can perform a write to a single location over voltage
  - When able, can perform a erase/write cycle to a single location over voltage
  - Can only bulk erase or remove code protection when $V_{DD} > 4.5V$

Lab 9: Calculating Checksums

- Open project lab9.pjt
- Memory read/write routines provided
- Add up each program memory location between 0000h and 0FFEh
- 2's complement the result
- Store the 14 LSbs into location 0FFFh
- Read back the value, print to LCD

Lab 9: Calculating Checksums
Flowchart



Lab 9: Goals

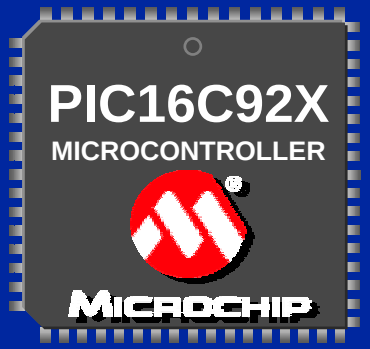
- Reading/Writing FLASH program memory
- Calculating checksums

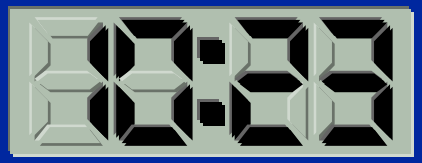
Peripherals
LCD Module (PIC16C9XX)

- Direct drive of LCD glass (operating & sleep modes)
- 1/1 and 1/3 bias LCD drive
- Static, 1/2, 1/3, 1/4 multiplex
- No software overhead
- Internal charge pump
- Multiple clock sources
  - Main OSC, TMR1, internal RC

Peripherals
LCD Module

- Up to 32 segments and 4 commons (S/W configurable I/O)
- 16 bytes of LCD RAM
- Can drive LCD while in SLEEP

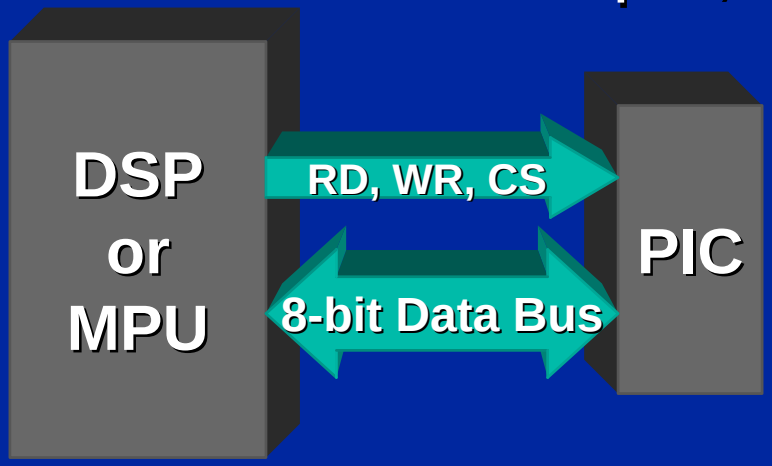




Commons	Segments	Pixels
1	32	32
2	31	62
3	30	90
4	29	116

Peripherals
Parallel Slave Port

- Provides an 8-bit interface such that the PICmicro 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



Peripherals

Parallel Slave Port Block Diagram