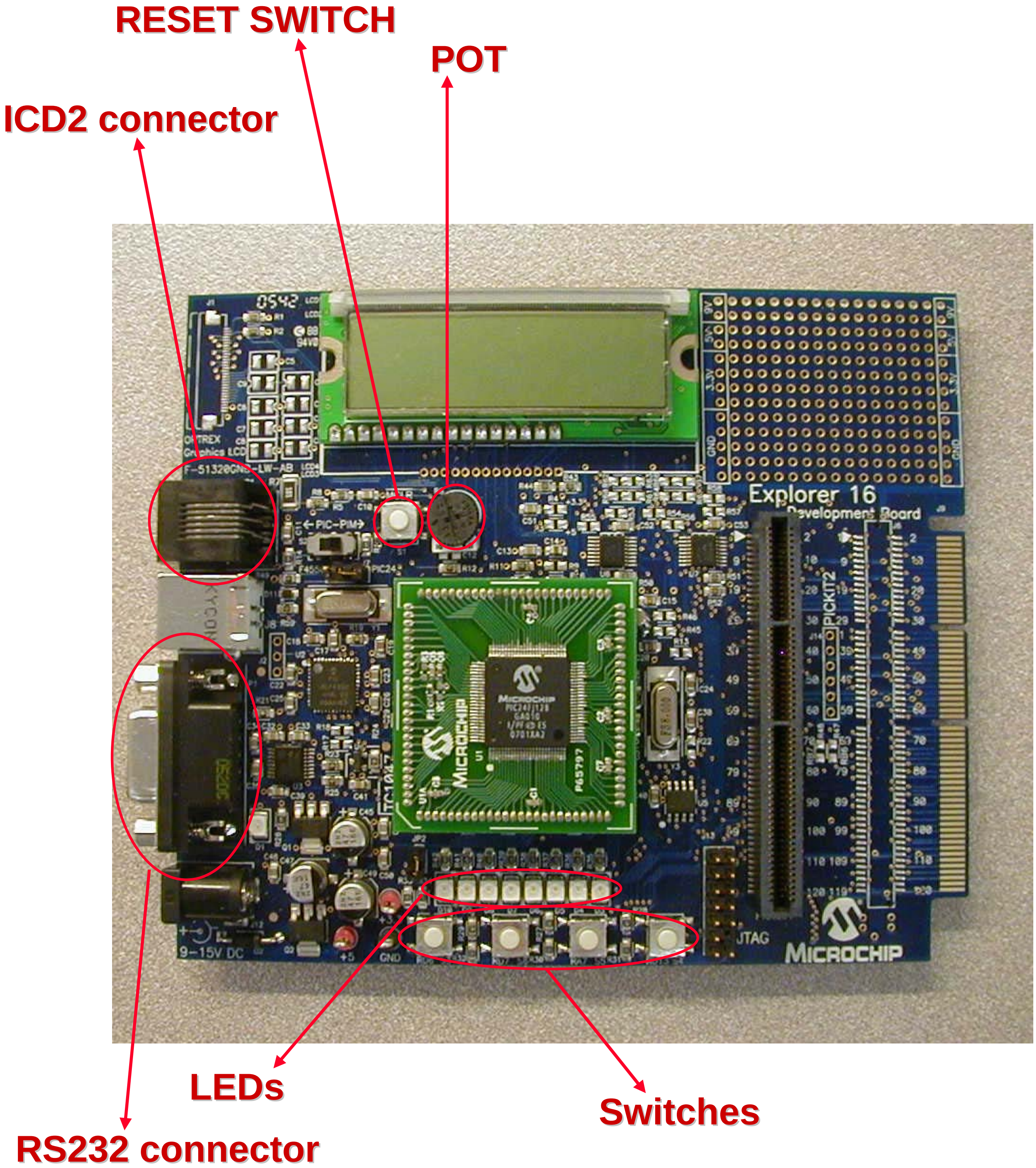
11010 SBC
16-bit Architecture, C30 & Standard Peripherals

Hand Out

MPLAB Navigation

- **Quick ways to find functions or variables in MPLAB**
  - Source Locator
    - **To Enable**
      - Right-click on editor and go to “Properties…”
      - Check “Enable Source Locator”
      - On the Project window, click on the “Symbols” tab. Right click and check “Enable Tag Locators”
    - **Use this feature to quickly navigate through large applications**
      - Right-click on a function or variable in code and select “Goto Locator” to jump its definition
      - In the project window under the symbols tab, you can browse through and double click items to jump there in code
  - Edit->Find in Files (ctrl+shift+F)
    - **Use this to search all files in the project for a variable, function name, or anything else**

Explorer 16


RESET SWITCH
POT
ICD2 connector
LEDs
Switches
RS232 connector
Explorer 16
Development Board
MICROCHIP
JTAG
9–15V DC

LAB 1
Working with C30 & MPLAB® IDE

LAB 1 Goals

- **To work with MPLAB® IDE environment**
- **To Do:**
  - Follow the presenter
- **Expected Result:**
  - Successfully build the project and program the device
  - LED D3 should blink

LAB 1 Solution

MPLAB IDE Editor

temp_128GA010.c*

```c
/************** START OF MAIN FUNCTION **********************************/

int main ( void )
{

    TRISAbits.TRISA0 = 0; //setup for LED output

    while(1){

        __builtin_btg(LATA,0x0); //toggle the LED pin

        delay(); //wait for some time

    }
}

void delay(void)
{
    unsigned int i;

    //delay for a while
    for(i = 0; i < 0xFFFF; i++);


}
/************** END OF MAIN FUNCTION ************************************/
```

LAB 2
Working with PSV

LAB 2 Goals

- To initialize PSV
- To store a “Hello World” string in PSV space
- To read and display this string on the LCD

LAB 2 To Do

- **Open the project**
    - **C:\Masters\11010\Student\Lab2\Lab2.mcp**
- **Open the file**
    - **C:\Masters\11010\Student\Lab2\Lab2.c**
- **Step 1**
    - Use Space attribute and define a array "MyString" and place it in PSV space.
        - **__attribute__ ((space(psv)))**
- **Step 2**
    - In the PSVInit() function use the built-in function of C30 to load PSVPAG register with the page of PSV space where the array is placed.
        - **__builtin_psvpage(variable)**
- **Build the Project and program the device**

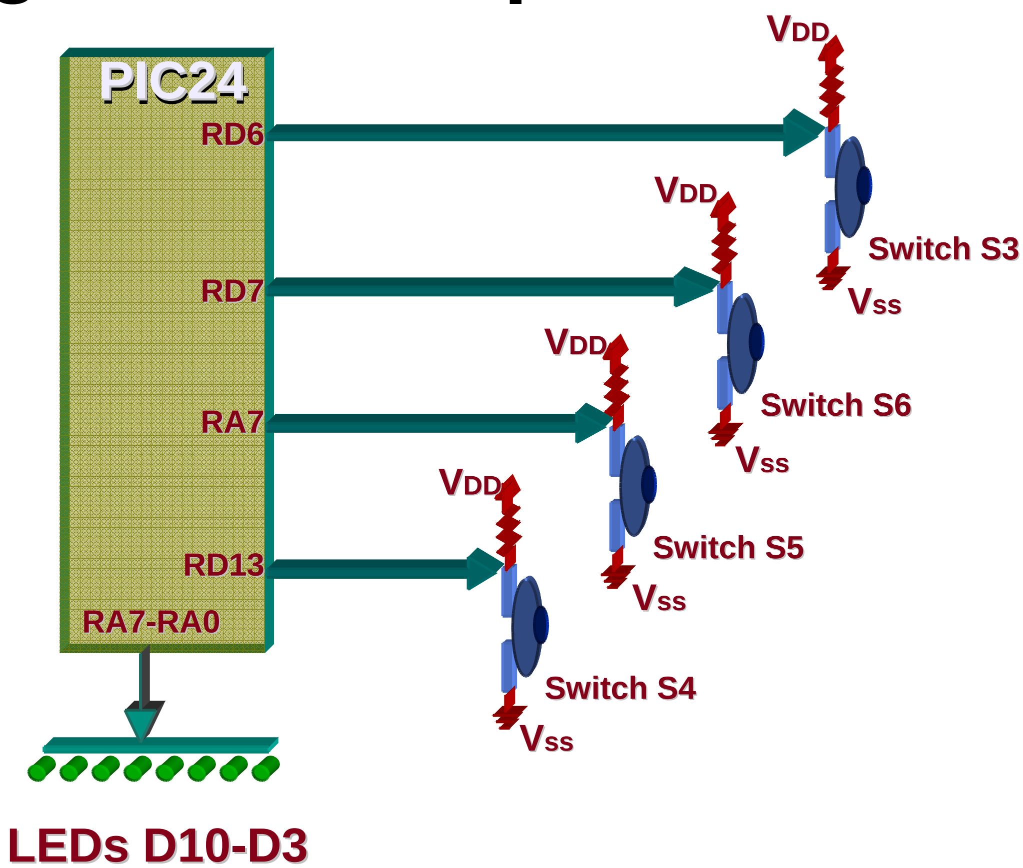
LAB 2 Expected Result

- Compare the content of the LCD display with the array defined.
- Both should match

LAB 3
Interrupt Handling

LAB 3 Goals

- Understand Interrupt configuration
- Understand Interrupt priority
- Writing Interrupt handler for a given Interrupt vector



LAB 3 To Do

- **Open the project**
  - C:\Masters\11010\Student\Lab3\Lab3.mcp
- **Open the file**
  - C:\Masters\11010\Student\Lab3\Lab3.c
- **Here Timer 3 is configured to give .5 sec ticks and Timer 5 is configured to give .25 Sec ticks.**
- **LED D3 indicates CPU is in T3 ISR and LED D7 indicates CPU is in T5 ISR.**
- **Look at TMR3Init and**

LAB 3 To Do

- **Step 1: Case 1 – S3**
  - Look for Switch Case1 and configure T3 priority to be higher than T5 priority.
  - By pressing S3 Case 1 will be executed.
- **Step 2: Case 2 – S6**
  - Look for Switch Case2 and configure T5 priority to be higher than T3 priority.
  - By pressing S6 Case 2 will be executed.
- **Step 3: Case 3 – S5**
  - Look for Switch Case3 and configure CPU priority to be higher than T5 priority and lower than T3 priority.
  - By pressing S5 Case 3 will be executed.
- **Step 4: Case 4 – S4**
  - Look for Switch Case4 and configure CPU priority to be higher than both T3 and T5 priority.
  - By pressing S4 Case 4 will be executed.

- **Watch what switch you are pressing!**
  - They are not in ascending order on the board
- **To configure the priority levels you must write into some registers, IPC2, IPC7 and SR. The details are given in following pages**
- **Build the project and program the device**

LAB 3 Interrupt Registers

IPC2: Interrupt priority control Register 2

Bit:15 … Bit:8

--	U1RXIP2	U1RXIP1	U1RXIP0	--	SPI1IP2	SPI1IP1	SPI1IP0

U1RXIP2–U1RXIP0: UART 1 Receive Interrupt Priority Level

SPI1IP2–SPI1IP0: SPI 1 Event Interrupt Priority Level

Bit:7 … Bit:0

--	SPF1IP2	SPF1IP1	SPF1IP0	--	T3IP2	T3IP1	T3IP0

SPF1IP2–SPF1IP0: SPI 1 Error Interrupt Priority Level

T3IP2–T3IP0: Timer 3 Interrupt Priority Level

LAB 3 Interrupt Registers

IPC7: Interrupt priority control Register 7

Bit:15 … **Bit:8**

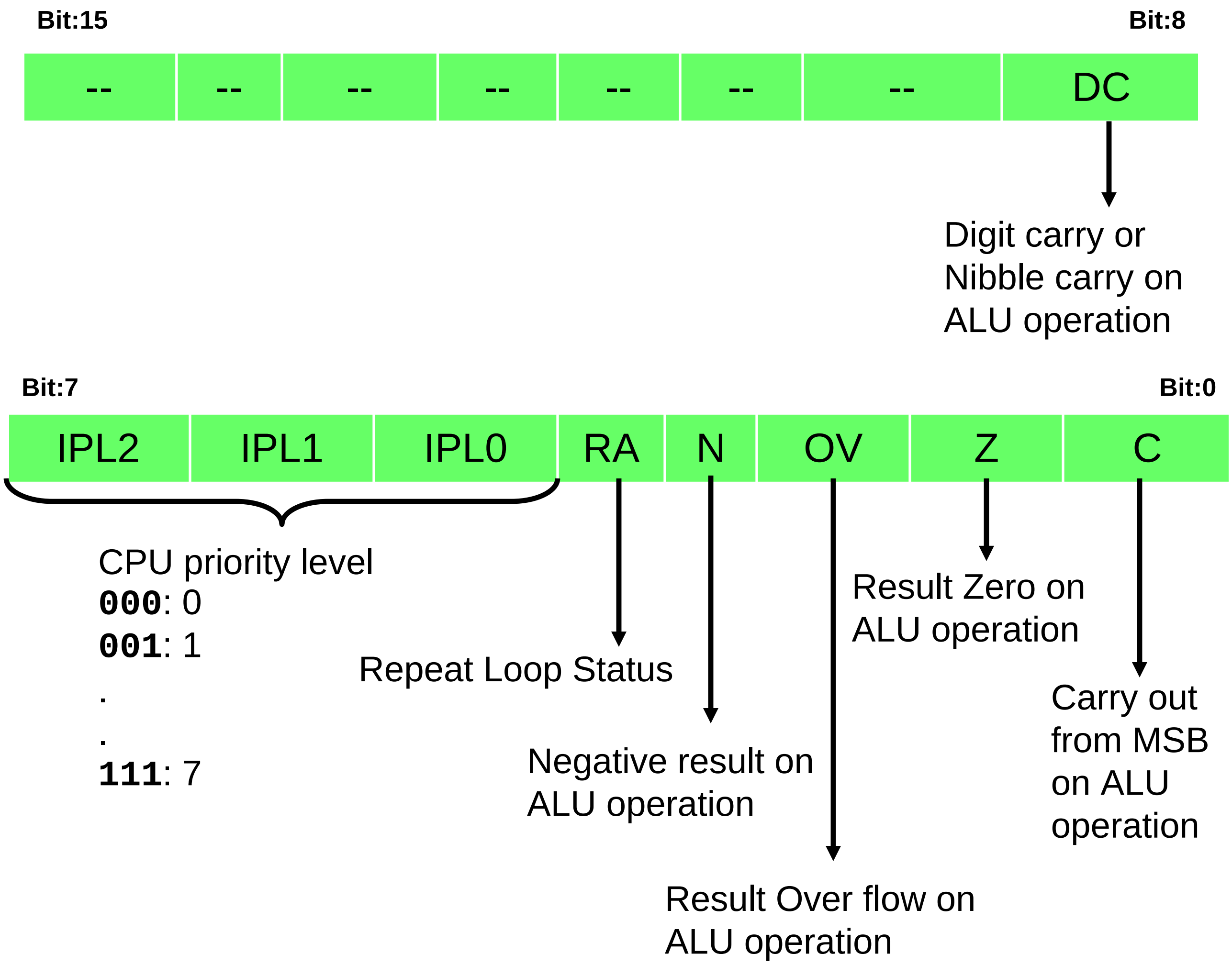
--	U2TXIP2	U2TXIP1	U2TXIP0	--	U2RXIP2	U2RXIP1	U2RXIP0
	UART 2 Transmit Interrupt Priority Level				UART 2 Receive Interrupt Priority Level		

Bit:7 … **Bit:0**

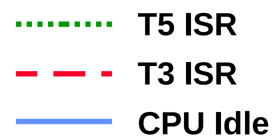
--	INT2IP2	INT2IP1	INT2IP0	--	T5IP2	T5IP1	T5IP0
	External Interrupt 2 Interrupt Priority Level				Timer 5 Interrupt Priority Level		

LAB 3 CPU Registers

SR: CPU Status Register



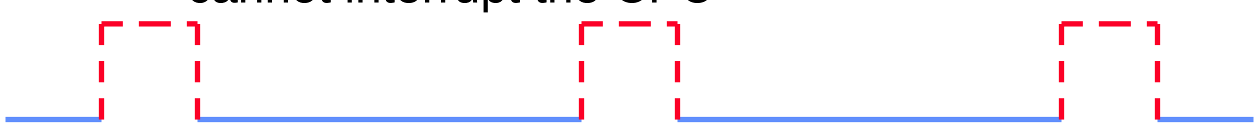
LAB 3 Expected Result



- **Case 1**
  - The LEDs D3 and D7 will be flashing but one after the other
  - D7 will never be ON when D3 is ON as T5 cannot preempt T3 ISR.

- **Case 2**
  - The LEDs D3 and D7 will be flashing at the same time
  - D7 becomes ON even when D3 is ON as T5 can preempt T3 ISR

- **Case 3**
  - The LED D3 will be flashing but not D7 as T5 cannot interrupt the CPU



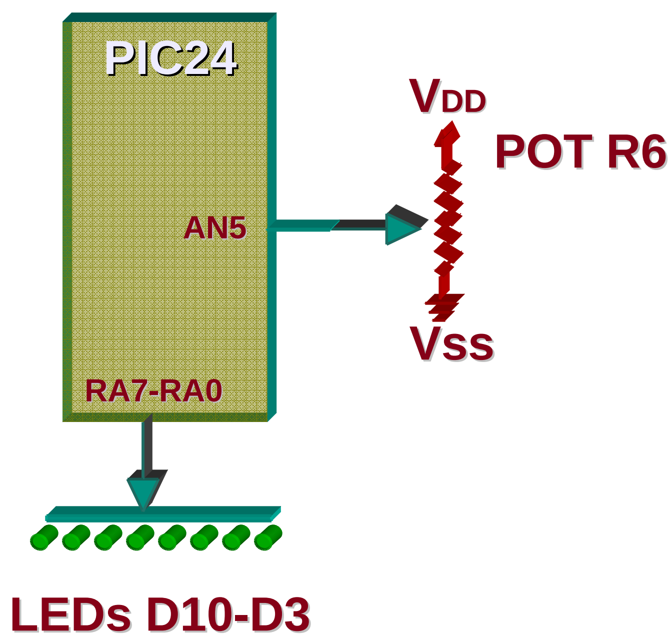
- **Case 4**
  - Both the LEDs D3 and D7 stops flashing as both T3 and T5 cannot interrupt the CPU

LAB 4

Working with ADC

LAB 4 Goals

- To configure ADC
- To configure I/O ports
- To read ADC and display on LEDs



LAB 4 To Do

- **Open the project**
  - C:\Masters\11010\Student\Lab4 \Lab4.mcp
- **Open the file**
  - C:\Masters\11010\Student\Lab4 \Lab4.c
- **Look for ADCInit() function and configure ADC by initializing the registers AD1CON1, AD1CON2, and AD1CON3 looking into the Register details on the next few pages.**
  - STEP 1: AD1CON1

LAB 4 To Do

- **Continue to configure ADC by initializing the registers AD1CHS, AD1PCFG, and AD1CSSL looking into the Register details on the next few pages.**
  - STEP 4: AD1CHS
    - **Set the positive sample input channel for MUX A to use AN5**
    - **Set the negative input channel for MUX A to use VR-**
  - STEP 5: AD1PCFG
    - **Set AD1PCFG so that the only pin using analog functionality is AN5**
  - STEP 6: AD1CSSL
    - **Channel scanning is not enabled, so no input channels should be selected for scanning**
- **Build the project and program the device**
- **Procedure to Test**
  - Vary the POT and observe LEDs

LAB 4 ADC Registers

AD1CON1: A/D CONTROL REGISTER 1

Bit:15 — Bit:8

ADON	--	ADSIL	--	--	--	FORM1	FORM0

ADON: ADC Module enable bit

ADSIL: ADC Module enable/disable in IDLE mode

FORM1:FORM0: **Result Format**
00: Intiger (0000 00dd dddd dddd)
01: Signed Intiger (ssss sssd dddd dddd)
10: Fractional (dddd dddd dd00 0000)
11: Signed Fractional (sddd dddd dd00 0000)

Bit:7 — Bit:0

SSRC2	SSRC1	SSRC0	--	--	ASAM	SAMP	DONE

SSRC2:SSRC0: Conversion Trigger Source Selection Bits
000: Manual Conversion Trigger
001: Active transition on INT0 pin triggers conversion
010: Timer3 compare triggers conversion
111: Auto conversion

SAMP: Start Sampling, If ASAM is ‘0’

DONE: Conversion Status bit

ASAM: Auto Sample Selection bit
1: Sample immediately after completion of last conversion.
0: Sample on setting of ‘SAMP’

LAB 4 ADC Registers

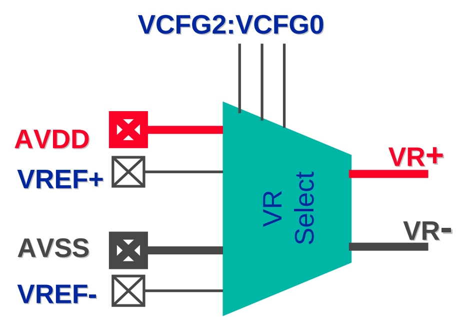
AD1CON2: A/D CONTROL REGISTER 2

Bit:15 … Bit:8

VCFG2	VCFG2	VCFG0	--	--	CSCNA	--	--

VCFG2:VCFG0	VR+	VR-
000	AVDD	AVSS
001	VREF+	AVSS
010	AVDD	VREF-
011	VREF+	VREF-
1xx	AVDD	AVSS

CSCNA: **Scan CH0 Mux A Input**



Bit:7 … Bit:0

BUFS	--	SMPI3	SMPI2	SMPI1	SMPI0	BUFM	ALTS

SMPI3:SMPI0	Interrupt Event (Sample/convert sequence)
0000	each
0001	alternate
....	....
1110	Every 15th
1111	Every 16th

ALTS: Sample alternatively MUX-A & MUX-B

BUFS: Buffer Status bit, is valid only when BUFM = '**1**'
1: Buffer 8-F is being filled, can access Buffer 0-7
0: Buffer 0-7 is being filled, can access Buffer 8-F

BUFM: Buffer Mode Select bit
1: Buffer configured as two 8-words buffers
0: Buffer configured as one 16-words buffers

LAB 4 ADC Registers

AD1CON3: A/D CONTROL REGISTER 3

Bit:15 … Bit:8

ADRC	--	--	SAMC4	SAMC3	SAMC2	SAMC1	SAMC0

A/D conversion Clock Source is ADRC OR system clock

A/D Sample Time Selection bits

SAMC4:SAMC0	Sampling Time
00000	0 T_{AD}
00001	1 T_{AD}
....	....
11110	30 T_{AD}
11111	31 T_{AD}

Bit:7 … Bit:0

ADCS7	ADCS6	ADCS5	ADCS4	ADCS3	ADCS2	ADCS1	ADCS0

A/D Conversion Clock Selection bits

ADCS7:ADCS0	Conversion Clock
00000000	T_{CY} (F_{CY})
00000001	2*T_{CY} (F_{CY} / 2)
....	....
11111110	255*T_{CY} (F_{CY} / 255)
11111111	256*T_{CY} (F_{CY} / 256)

LAB 4 ADC Registers

AD1CHS : A/D Input Select Register

Bit:15 Bit:8

CH0NB	--	--	--	CH0SB3	CH0SB2	CH0SB1	CH0SB0

CH0 Negative input for MUX B
1: AN1
0: VR-

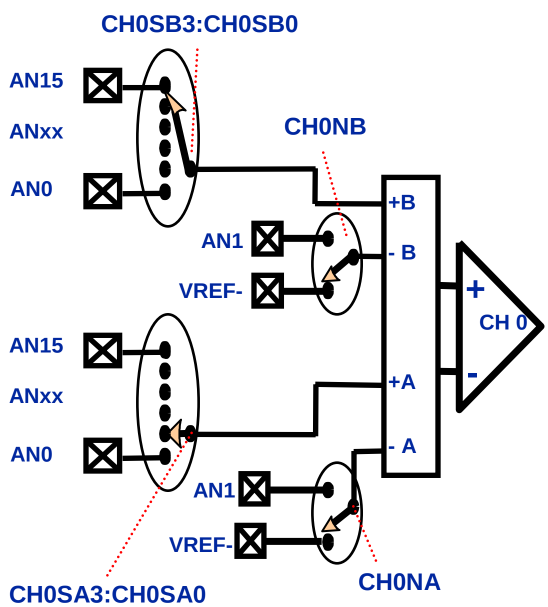
CH0SB3:CH0SB0	CH0 Positive Input for MUX B
0000	AN0
0001	AN1
....	....
1110	AN14
1111	AN15

Bit:7 Bit:0

CH0NA	--	--	--	CH0SA3	CH0SA2	CH0SA1	CH0SA0

CH0 Negative input for MUX A
1: AN1
0: VR-

CH0SA3:CH0SA0	CH0 Positive Input for MUX A
0000	AN0
0001	AN1
....	....
1110	AN14
1111	AN15



LAB 4 Expected Result

- **POT value is averaged for 16 samples over 1 ms.**
- **POT value is displayed on LEDs as a binary value from 0 to 255**
- **Pin RB2 toggles each time 16 samples are taken (a frequency of 500 Hz)**

LAB 5

Working with a 32 bit Timer

LAB 5 Goals

- Understand working of Timers in 32bit mode
- Configure the Timer 2/3 pair for 32 bit mode
- Implement a stop watch

LAB 5 To Do

- **Open the project**
  - C:\Masters\11010\Student\Lab5\Lab5.mcp
- **Open the file**
  - C:\Masters\11010\Student\Lab5\Lab5.c
- **Look for Timer23Init() function and configure Timer 2 & 3 by initializing the registers T2CON, PR2 and PR3.**
  - STEP 1: T2CON
    - **Select Internal clock as clock source (Fosc/2)**
    - **1:1 Pre-scale**

LAB 5 Timer Registers

T2CON: Timer 2 Control Register

Bit:15 … Bit:8

TON	--	TSIDL	--	--	--	--	--

- TON: Timer 2 ON
- TSIDL: Timer 2 enable/disable in IDLE mode bit

Bit:7 … Bit:0

--	TGATE	TCKPS1	TCKPS0	T32	--	TCS	--

- TGATE: Enable Gated Time Accumulation
- TCKPS1, TCKPS0: Timer2 Prescale Selection bits
  - **11**: 1 : 256
  - **10**: 1 : 64
  - **01**: 1 : 8
  - **00**: 1 : 1
- T32: Enable 32-bit mode
- TCS: Timer2 Clock Source Selection bits
  - **1**: External Clock from T2CK pin
  - **0**: Internal Clock (FOsc/2)

LAB 5 Expected Result

- The On-Board LCD will be Displaying
  - Press S3 - Start
- Press the Switch S3 to start timer
- Again press the Switch S3 to stop timer and LCD displays the Time elapsed between the start and stop

LAB 6

Working with UART

LAB 6 Goals

- **Understand Configuration of UART module**
- **Understand Transmit and Receive interrupts**
- **Understand the advantages of the FIFO**
  - A slow baud rate is used to make this more easily visible
- **Write a software to Transmit and Receive data using UART**

LAB 6 To Do

- **Open the project**
  - C:\Masters\11010\Student\Lab6\Lab6.mcp
- **Open the file**
  - C:\Masters\11010\Student\Lab6\Lab6.c
- **Look for UARTInit() function and configure UART by initializing the registers U2MODE, U2STA and U2BRG.**
  - STEP 1: U2BRG
    - **Load the count to get 300 baudrate**
    - **BRG = Fcy/(16*BaudRate)-1 where Fcy = 4MHz**

LAB 6 UART Registers

UxMODE: UART Mode register

Bit:15 … Bit:8

UARTEN	--	USIDL	IREN	RTSMD	--	UEN1	UEN0

- UARTEN: UART Module enable bit
- USIDL: UART Module enable/disable in IDLE mode bit
- IREN: IrDA Encoder/Decoder enable bit
- RTSMD: UxRTS pin Mode Selection bit
  1: Simplex mode
  0: Flow control mode
- UEN1:UEN0: UART Flow control Selection bits
  The pins enabled with UxTx & UxRx
  11: BCLKx
  10: UxCTS & UxRTS
  01: UxRTS
  00: none

Bit:7 … Bit:0

WAKE	LPBACK	ABAUD	RXINV	BRGH	PDSEL1	PDSEL0	STSEL

- WAKE: Wake up from SLEEP on detection of START bit
- LPBACK: Loop back mode selection bit
- ABAUD: Auto Baud measurement enable bit
- RXINV: Rx Polarity inversion bit
  1: UxRx Idle state is '0'
  0: UxRx Idle state is '1'
- BRGH: High Baud Rate enable bit
  1: BRG is prescaled by 4
  0: BRG is prescaled by 16
- PDSEL1:PDSEL0: Parity and Data width selection bit
  11: 9-bit data, No parity
  10: 8-bit data, Odd parity
  01: 8-bit data, Even parity
  00: 8-bit data, No parity
- STSEL: STOP bit selection bit
  1: Two STOP bits
  0: One STOP bit

LAB 6 UART Registers

UxSTA: UART Status and Control register

Bit:15 … **Bit:8**

UTXISEL1	UTXINV	UTXISEL0	--	UTXBRK	UTXEN	UTXBF	TRMT

- UTXINV: IrDA Transmit Polarity bit
  1: IrDA UxTx Idle State is ‘1’
  0: IrDA UxTx Idle State is ‘0’
- UTXISEL1, UTXISEL0: Transmission Interrupt mode selection bit
  10: Interrupt on Tx Buffer fully empty
  01: Interrupt on complete of transmission
  00: Interrupt on Tx buffer not full.
- UTXBRK: Transmit Break bit
- UTXEN: Transmit Enable bit
- UTXBF: Transmit buffer full Status bit
- TRMT: Transmit Shift Register Empty Status bit

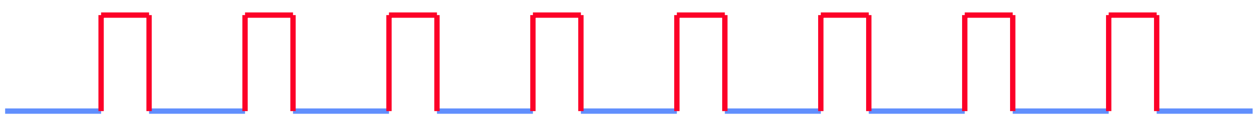
Bit:7 … **Bit:0**

URXISEL1	URXISEL0	ADDEN	RIDLE	PERR	FERR	OERR	URXDA

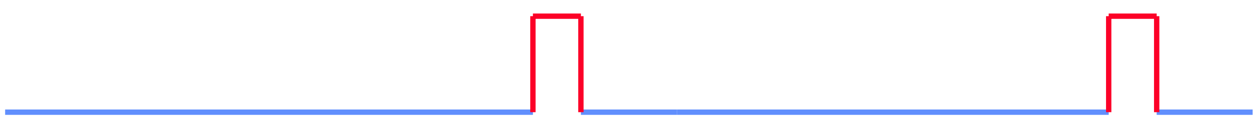
- URXISEL1, URXISEL0: Receive Interrupt mode selection bit
  11: Interrupt on Rx Buffer full
  10: Interrupt on Rx Buffer 3/4full
  0x: Interrupt on every Receive
- ADDEN: Address Detect mode Enable bit
- RIDLE: Receiver Idle Status bit
- PERR: Parity Error Status bit
- FERR: Framing Error Status bit
- OERR: Rx Buffer Over Run Error Status bit
- URXDA: Rx Buffer Data available Status bit

LAB 6 Expected Result

- **LED D3 indicates amount of CPU time spent processing UART data. Slower blinking means indicates less time spent servicing UART interrupt routine.**
- **Press Switch S4, the rate at which LED flashes decreases as now UART is using the RxBuffer and interrupting only on Buffer.**



Normal LED pattern – blink each receive



S4 Pressed – blink each full buffer, 4 characters received

- **The transmitted data can be observed on the screen as the received Data is transmitted back**
  - Without S4, the data comes back as you enter it
  - With S4 pressed, the data will come back in packets of 4

Thank You