

Lab 4: Wake Up For Interrupt

This section will show you how to run the ADCC in continuous operation mode. Through this your ADCC will continuously sample once initially started in software. Once it is started, you can put the device to sleep. We will also set the ADCC up to generate an interrupt whenever the noise level exceeds a certain threshold. The device will then be able to wake itself from sleep and print out the noise level that caused it to wake up.

1. Configure ADCC:

- Set Clock Source to FRC
- Enable Continuous Operation
- Set Acquisition Count to 6000 (You should now see the field **Acquisition Time** reflecting around 10 ms. This provides a 10 ms delay in-between each ADCC sample reading. This will help minimize the ADCC interrupting multiple times when responding to the same noise outburst.)

ADCC

Easy Setup
 Registers

Hardware Settings

☒ Enable ADC

Operating

Basic_mode

▼ ADC

ADC Clock

Clock Source

Frc

Clock

FOSC/2

1 TAD

1.7 us

Sampling Frequency

51.1509kHz

Conversion Time

= 11.5 * TAD = 19.55 us

Result Alignment

right

Positive Reference

VDD

Negative Reference

VSS

Auto-conversion Trigger

disabled

☒ Enable Continuous Operation

☐ Enable Stop on Interrupt

Acquisition Count 0 ≤

6000

≤ 8191

☐ Enable Double Sample

Acquisition Time

10.2 ms

Future Lab Manual

2. Enable Thresholding

- Set Error Calculation to **Actual result vs setpoint**
- Change Setpoint to 0
- Change Threshold Interrupt to **ADERR > ADUTH**
- Change Upper Threshold to 100

The **ADERR** register will be the result of the **ADC Actual Result – the Setpoint**. Since we set the setpoint to 0, this value will simply be the Actual Result of the ADCC reading. The ADERR register will then be compared to our upper threshold which we set to **100**. If our ADC result exceeds the value of 100 then we need to generate an interrupt. The correct configuration should look like the image below.

▼ Computation Feature

Error Calculation	Actual result vs setpoint	
Setpoint	-32768 ≤ 0 ≤ 32767	
Threshold Interrupt	ADERR > ADUTH	
Lower Threshold	-32768 ≤ 0 ≤ 32767	
Upper Threshold	-32768 ≤ 100 ≤ 32767	
Repeat	0 ≤ 0 ≤ 255	
Acc Right Shift	0 ≤ 0 ≤ 5	

3. Enable Interrupts:

► CVD Features

☐ Enable ADC Interrupt

☒ Enable ADC Threshold Interrupt

4. Ensure that Interrupts are enabled in the Interrupt Module

MPLAB X IDE v5.20 - NoiseClick: default

File Edit View Navigate Source Refactor Production Debug Team Tools Window Help

Projects Files Classes Resource Management (MCC) x

Tree View Flat View

Project Resources Generate Import... Export

▼ System

Interrupt Module

Pin Module

System Module

▼ Peripherals

EUSART1 [PIC10 / PIC12 / PIC16 / PIC18 MCUs by Microchip

ADCC [PIC10 / PIC12 / PIC16 / PIC18 MCUs by Microchip T

Device Resources

► NCO

► PWM

► SMT

► Timer

► ZCD

▼ Libraries

Bootloader Generator

Foundation Services

LIN

Interrupt Module

Easy Setup

Interrupts

Please remember to enable the Peripheral and Global Interrupts in yo

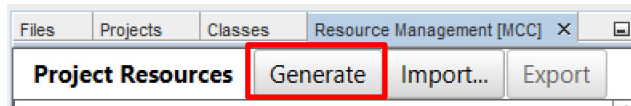
▼ Interrupt Vector

Order Up Down Single ISR per interrupt

Order	Module	Interrupt	Enabled
1	EUSART1	TXI	<input type="checkbox"/>
2	EUSART1	RCI	<input type="checkbox"/>
3	ADCC	ADI	<input type="checkbox"/>
4	ADCC	ADTI	<input checked="" type="checkbox"/>
5	Pin Module	IOCI	<input type="checkbox"/>

Future Lab Manual

5. **Generate Code:** Click on generate code from the top of the MCC window



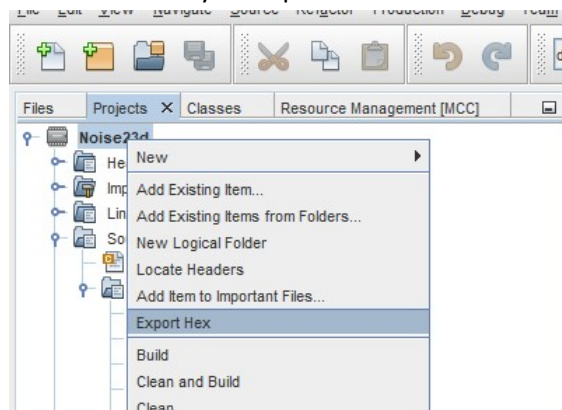
6. **Edit main.c:** Now go back to Main.c. Make the following changes:

```
50 void main(void)
51 {
52     // Initialize the device
53     SYSTEM_Initialize();
54
55     // If using interrupts in PIC18 High/Low Priority Mode you need to enable the Global High and Low Interrupts
56     // If using interrupts in PIC Mid-Range Compatibility Mode you need to enable the Global and Peripheral Interrupts
57     // Use the following macros to:
58
59     // Enable the Global Interrupts
60     INTERRUPT_GlobalInterruptEnable();
61
62     // Disable the Global Interrupts
63     INTERRUPT_GlobalInterruptDisable();
64
65     // Enable the Peripheral Interrupts
66     //INTERRUPT_PeripheralInterruptEnable();
67
68     // Disable the Peripheral Interrupts
69     //INTERRUPT_PeripheralInterruptDisable();
70     ADCC_StartConversion(sensor);
71     while (1)
72     {
73         SLEEP();
74         printf("\n %i", ADCC_GetConversionResult());
75         __delay_ms(100);
76     }
77 }
```

As mentioned previously, in this part of the lab we only want to start the ADCC conversion once. After it is started in software, the ADCC will run continuously. Therefore the device can be put into sleep mode in the while(1) loop. However, we need to enable global and peripheral interrupts so that our ADCC can wake the device once the threshold is exceeded.

Once the device is woken up from the threshold being exceeded, we want to print out the ADCC conversion result. We will add another delay to ensure that everything is successfully printed out to the serial terminal. The device will then go to sleep until the next interrupt is generated.

7. **Compile and Program:** Go to the top level of your project and right click. In the dropdown menu you will see **Export Hex**. It will then ask you to provide a file name. Do this and click save.



Future Lab Manual

8. **Connect and See Output:** Navigate back to CoolTerm and click connect if the settings have not changed. Now you should only see values being printed out when you tap on the noise click or make a loud enough noise into the microphone.