EEE 407 Microprocessors and Microcontrollers Laboratory

EXPERIMENT 7

Analog-to-Digital Conversion (ADC)

Objective: The goal of this experiment is to learn how to convert real-world analog signals into digital data. Students will read a variable voltage from a potentiometer using the **PIC18F452 ADC module** and visualize the resulting 10-bit digital value on a **7-segment display** unit. The lab focuses on understanding resolution, reference voltages, and hardware-software interfacing.

1. Introduction

Digital signals have **two discrete states**, which are decoded as **high** and **low**, and interpreted as **logic 1** and **logic 0**. **Analog signals**, on the other hand, are **continuous**, and can have any value within defined range. **A/D converters** are specialized circuits which can convert analog signals (voltages) into a digital representation, usually in form of an **integer number**. The value of this number is **linearly dependent** on the input voltage value. Most microcontrollers nowadays internally have A/D converters connected to one or more input pins. Some of the most important parameters of A/D converters are **conversion time** and **resolution**. Conversion time determines how fast can an analog voltage be represented in form of a digital number. This is an important parameter if you need fast data acquisition. The other parameter is resolution. Resolution represents the number of discrete steps that supported voltage range can be divided into. It determines the sensitivity of the A/D converter. Resolution is represented in maximum number of bits that resulting number occupies. Most PIC® microcontrollers have 10-bit resolution, meaning that maximum value of conversion can be represented with 10 bits, which converted to integer is $2^{10} = 1024$. This means that supported voltage range, for example from 0-5V, can be divided into 1024 discrete steps of about 4.88mV. EasyPIC™ v7 provides an interface in form of two potentiometers for simulating analog input voltages that can be routed to any of the 10 supported analog input pins.

2. Understanding the ADC Control Registers

To utilize the built-in 10-bit ADC module of the **PIC18F452**, we must configure its operation through two primary control registers: **ADCON0** and **ADCON1**. These registers act as the "dashboard" where we define which pins are analog, how fast the conversion happens, and where the result is stored.

a) ADCON1: Port Configuration and Result Format

The **ADCON1** register is the first step in setting up the hardware interface. In our experiment, we load it with the hex value **0x82** (binary 10000010).

- **ADFM (Bit 7) = 1 (Right Justified):** Since the ADC provides a 10-bit result but stores it in two 8-bit registers (ADRESH and ADRESL), we must choose an alignment. We set this to **1** so the result is "right-justified," making it easy to read as a single 10-bit integer ($0-1023$).
- **PCFG3:PCFG0 (Bits 3-0) = 0010:** These bits determine which pins act as Analog inputs and which are Digital. By choosing 0010, we configure **AN0 through AN4** as analog inputs.

Note for Students: In the PIC18F452 architecture, analog pins are often grouped. To use **RA5 (AN4)**, we must also enable the pins before it (RA0-RA3) as analog-capable.

b) ADCON0: Channel Selection and Operation Control

The **ADCON0** register controls the "live" process of conversion.

- **ADCS1:ADCS0 (Bits 7-6):** These bits select the **A/D Conversion Clock**. We use a frequency divider (like $\frac{F_{osc}}{8}$) to ensure the ADC has enough time to accurately sample the voltage.
- **CHS2:CHS0 (Bits 5-3):** These bits select the active channel. To read from the potentiometer on **RA5**, we select **Channel 4 (100)**.
- **GO/DONE (Bit 2):** This is the "start" button. Setting this bit to **1** begins the conversion. The hardware automatically clears it to **0** once the process is finished.
- **ADON (Bit 0):** This bit simply powers on the ADC module.

3. The Conversion Process: From Voltage to Integer

The ADC module does not "know" the actual voltage value; it only knows the **ratio** between the input signal and the **Reference Voltage (V_{ref})**.

In this lab, our V_{ref} is tied to the microcontroller's supply voltage (V_{DD}), which we measured as **4.855V (It may vary and should be measured again during the experiment.)**. The 10-bit resolution means the range from 0 to 4.855 is divided into $2^{10} = 1024$ discrete steps.

The mathematical relationship is defined as:

$$Digital\ Result = \frac{V_{input}}{V_{reference}}\ x\ 1023$$

4. Data Processing for Display

Once the **adc_result** is obtained, it is a raw 10-bit integer. To display this on the 7-Segment Display, we perform Integer Division and Modulo operations to isolate each digit:

- **Thousands**: result / 1000
- **Hundreds:** (result / 100) % 10
- **Tens:** (result / 10) % 10
- **Ones:** result % 10

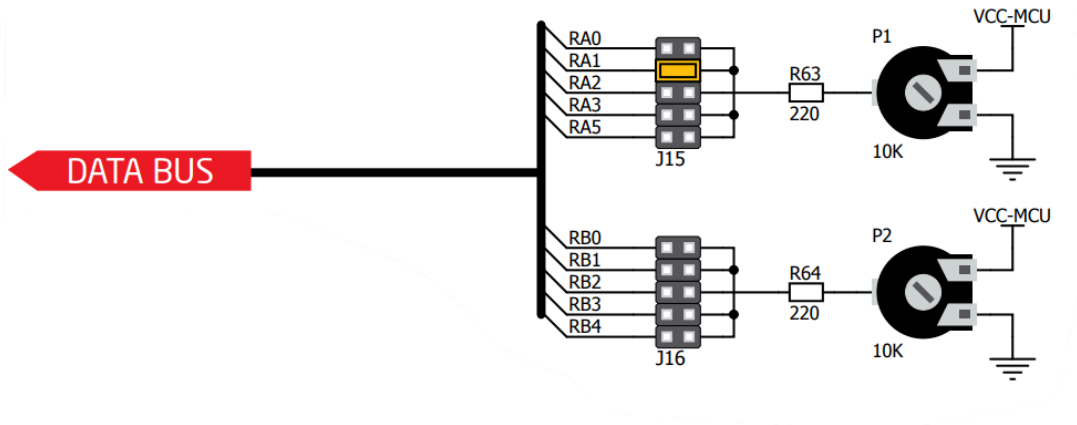
This process allows us to translate a binary value stored in the MCU's memory into a human-readable format on the development board's interface.

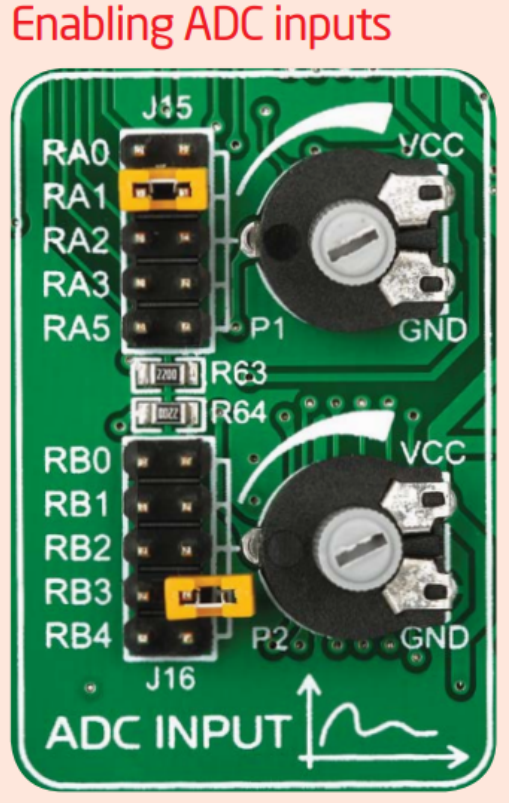
Experimental Work

1. Hardware Setup and Jumper Configuration

Before powering up the EasyPIC v7 board, ensure the hardware is correctly configured to route the analog signal to the microcontroller:

1. **Potentiometer Selection:** Locate the **J15** jumper group. Place the jumper on the RA5 position. This connects the center pin of potentiometer **P1** to the **AN4** analog input of the PIC18F452.
2. **Display Activation:** Locate the **SW4** DIP switch on the board. Turn ON the switches for **Digit 1,Digit 2, Digit 3, and Digit 4** (connected to RA0, RA1, RA2, and RA3). This enables the 4-digit 7-segment display.
3. **Port Output:** Ensure that the LED switches for **PORTD** are ON, as PORTD will be used to send segment data (a-g) to the displays.





2. Software Implementation

Load your **XC8 compiler** project and implement the following logic. Follow the register configurations discussed in the theory section.

1. **Initialize ADC:** Configure ADCON1 to set RA5 as an analog input and ensure the result is right-justified.
2. **Sampling Loop**: Create a continuous loop that:
    - Starts an ADC conversion on **Channel 4.**
    - Waits for the **GO/DONE** bit to clear.
    - Stores the 10-bit result (***ADRESH*** **:** ***ADRESL***) into an unsigned int variable.
3. **Multiplexing Display:** Implement a multiplexing routine to drive the 4 digits. Use a small delay (e.g., 5ms) between each digit to ensure a flicker-free visual experience.

3. Procedure and Observations

Step 1: Calibration Check

Turn the potentiometer P1 fully counter-clockwise (GND position).

- Observe: What value is displayed on the 7-segment display? (Expected: 0)
- Measurement: Use a Digital Multimeter (DMM) to measure the voltage at the RA5 pin. It should be approximately 0.00V.

Step 2: Linearity and Mid-point Test

Slowly rotate the potentiometer until the DMM reads exactly half of your supply voltage (approx. 2.41V if V_{DD} = 4.83V).

- **Observe:** What is the digital value on the display?
- **Calculation:** Calculate the theoretical value using the formula: $\frac{Result = (V_{in}}{V_{ref})} \; x \; 1023$. Does the calculation match the display?

Step 3: Maximum Range Analysis

Turn the potentiometer fully clockwise to the maximum position.

- **Observe:** Record the maximum value shown on the display.
- **Analysis:** If the value is less than 1023 (e.g., 1009), measure the voltage at the RA5 pin and the V_{DD} pin. Explain why the digital result does not reach the theoretical maximum based on these voltage measurements.