Microcomputers

Analog-to-Digital and Digital-to-Analog Conversion

Digital Signal Processing

Analog-to-Digital Converter (ADC) converts an input analog value to an output digital representation.

This digital data is processed by a microprocessor and output to a Digital-to-Analog Converter (DAC) which converts an input binary value to an output voltage.
Vocabulary

- ADC (Analog-to-Digital Converter) – converts an analog signal (voltage/current) to a digital value
- DAC (Digital-to-Analog Converter) – converts a digital value to an analog value (voltage/current)
- Sample period – for ADC, time between each conversion
  - Typically, samples are taken at a fixed rate
- Vref (Reference Voltage) – analog signal varies between 0 and Vref, or between +/- Vref
- Resolution – number of bits used for conversion (8 bits, 10 bits, 12 bits, 16 bits, etc).
- Conversion Time – the time it takes for a analog-to-digital conversion

An N-bit ADC

Maps a voltage \( (Vin) \) to a digital code \( ADC\_code \)

\[
ADC\_code = (Vin/Vref) \times 2^N
\]

\( Vin \) is always considered less than \( Vref \), so \( Vin/Vref \) is always < 1.0. Any fractional part of the code is truncated.
**Example: A 10-bit ADC**

Vref = 4 V  
Maps a voltage \((Vin)\) to a digital code \(ADC\_code\)

\[ ADC\_code = (Vin/Vref) \times 2^N \]

\[ = (3 V/4 V) \times 1024 \]

\[ = 0.75 \times 1024 = 768 \]

Vin = 3.0 V  
ADC\_code = 768

**Going from Code to Voltage**

Vref = 4.0 V  
\(ADC\_code = (Vin/Vref) \times 2^N\)

\[ ADC\_code/2^N \times Vref = Vin \]

Vin = 2.168 V  
\[ Vin = ADC\_code/2^N \times Vref \]

\[ = 555/1024 \times 4 V \]

\[ = 2.167968 \]

\[ = \sim 2.168 \]
ADC Resolution

For an N-bit ADC, the smallest input voltage that can be resolved is 1 LSb, or:

\[ \frac{1}{2^N} \times (V_{ref+} - V_{ref-}) \]

Where \( V_{ref+} \) is the positive reference voltage and \( V_{ref-} \) is the negative reference voltage.

We will use \( V_{ref-} = 0 \) V, and refer to \( V_{ref+} \) as simply \( V_{ref} \), so this simplifies to

\[ \frac{1}{2^N} \times V_{ref} \]

For \( V_{ref} = 4 \) V, and \( N = 4 \), what is 1 LSb?

\[ \frac{1}{2^4} \times 4 \text{ V} = 1/16 \times 4 \text{ V} = 0.25 \text{ V} \]

Example: 10-bit ADC Resolution

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{in} = 3.00390625 \text{ V} \]

\[ V_{in} = 3.0 \text{ V} \]

\[ V_{in} = 0.00390625 \text{ V} \]

\[ V_{in} = 0 \text{ V} \]

\[ 1 \text{ LSB voltage} = \frac{1}{2^N} \times V_{ref} \]

\[ = \frac{1}{1024} \times 4 \text{ V} \]

\[ = 0.00390625 \text{ V} \]

\[ = \approx 3.9 \text{ mV} \]

\[ V_{in} = 3.00390625 \text{ V} \]

\[ V_{in} = 3.0 \text{ V} \]

\[ V_{in} = 0.00390625 \text{ V} \]

\[ V_{in} = 0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]

\[ V_{ref} = 4.0 \text{ V} \]
ADC, DAC Equations

ADC: \( V_{in} = \text{input voltage}, \ V_{ref^+} = \text{reference voltage}, \ V_{ref^-} = 0 \text{ V}, \) \( N = \text{number of bits of precision} \)

\[
\frac{V_{in}}{V_{ref}} \times 2^N = \text{output code}
\]

\[
\frac{\text{output code}}{2^N} \times V_{ref} = V_{in}
\]

1 LSB = \( \frac{V_{ref}}{2^N} \)

DAC: \( V_{out} = \text{output voltage}, \ V_{ref} = \text{reference voltage}, \) \( N = \text{number of bits of precision} \)

\[
\frac{V_{out}}{V_{ref}} \times 2^N = \text{input code}
\]

\[
\frac{\text{input code}}{2^N} \times V_{ref} = V_{out}
\]

1 LSB = \( \frac{V_{ref}}{2^N} \)

Sample ADC, DAC Computations

If \( V_{ref} = 5\text{ V}, \) and a 10-bit A/D output code is 0x12A, what is the ADC input voltage?

\[
V_{in} = \frac{\text{output code}}{2^N} \times V_{ref} = \frac{0x12A}{2^{10}} \times 5 \text{ V} = \frac{298}{1024} \times 5 \text{ V} = 1.46 \text{ V (ADC Vin)}
\]

If \( V_{ref} = 5\text{ V}, \) and an 8-bit DAC input code is 0xA9, what is the DAC output voltage?

\[
V_{out} = \frac{\text{input code}}{2^N} \times V_{ref} = \frac{0xA9}{2^8} \times 5 \text{ V} = \frac{169}{256} \times 5 \text{ V} = 3.3 \text{ V (DAC Vout)}
\]

If \( V_{ref} = 4\text{ V}, \) and an 8-bit A/D input voltage is 2.35 V, what is the ADC output code?

\[
\text{output code} = \frac{V_{in}}{V_{ref}} \times 2^N = \frac{2.35 \text{ V}}{4 \text{ V}} \times 2^8 = 0.5875 \times 256 = 150.4 \approx 150 = 0x96 \text{ (ADC output code)}
\]
Digital-to-Analog Conversion

For a particular binary code, output a voltage between 0 and \( V_{\text{ref}} \)

\[
D[7:0] \quad \text{DAC} \quad V_{\text{out}}
\]

Assume a DAC that uses an unsigned binary input code, with 0 < \( V_{\text{out}} < V_{\text{ref}} \). Then

- \( D = 0000\ 0000 \) \( V_{\text{out}} = 0V \)
- \( D = 0000\ 0001 \) \( V_{\text{out}} = V_{\text{ref}}(1/256) \) (one LSB)
- \( D = 0000\ 0010 \) \( V_{\text{out}} = V_{\text{ref}}(2/256) \)
- ...
- \( D = 1111\ 1111 \) \( V_{\text{out}} = V_{\text{ref}}(255/256) \) (full scale)

DAC Output Plot

Output signal increases in 1 LSB increments.

\[
\begin{align*}
4/256\ V_{\text{ref}} \\
3/256\ V_{\text{ref}} \\
2/256\ V_{\text{ref}} \\
1/256\ V_{\text{ref}} \\
\end{align*}
\]

Input code
An N-bit DAC

Maps a digital code \(\text{DAC\_code}\) to a voltage \(\text{Vout}\)

\[
\text{Vout} = \frac{\text{DAC\_code}}{2^N} \times \text{Vref}
\]

A 1-bit ADC

Vin > Vref/2: Vout = Vdd
Vin < Vref/2: Vout = 0
**Counter Ramp ADC**

Control logic use a counter to apply successive codes 0, 1, 2, 3, 4... to DAC (Digital-to-Analog Converter) until DAC output is greater than Vin. This is SLOW, and have to allocate the worst case time for each conversion, which is $2^n$ clock cycles for an N-bit ADC.

---

**Successive Approximation ADC**

Initially set VDAC to $\frac{1}{2}$ Vref, then see if Vin higher or lower than VDAC. If $> \frac{1}{2}$ Vref, then next guess is between Vref and $\frac{1}{2}$ Vref, else next guess is between $\frac{1}{2}$ Vref and GND. Do this for each bit of the ADC. Takes N clock cycles.
**Successive Approximation Example**

- Given a 4-bit Successive Approximation ADC, and $V_{\text{ref}} = 4\, \text{V}$. Let $V_{\text{in}} = 3.14159\, \text{V}$. Clear DAC input to 0b0000.
  1. First guess, DAC input = 0b1000 = 8, so $V_{\text{dac}} = 8/24 \times 4\, \text{V} = 8/16 \times 4\, \text{V} = 2\, \text{V}$. $V_{\text{dac}} (2\, \text{V}) < V_{\text{in}} (3.14159\, \text{V})$, so guess of ‘1’ for MSb of DAC was correct.
  2. Set next bit of DAC to ‘1’. DAC input = 0b1100 = 12, so $V_{\text{dac}} = 12/16 \times 4\, \text{V} = 3\, \text{V}$. $V_{\text{dac}} (3\, \text{V}) < V_{\text{in}} (3.14159\, \text{V})$, so guess of ‘1’ for bit2 of DAC was correct.
  3. Set next bit of DAC to ‘1’. DAC input = 0b1110 = 14, so $V_{\text{dac}} = 14/16 \times 4\, \text{V} = 3.5\, \text{V}$. $V_{\text{dac}} (3.5\, \text{V}) > V_{\text{in}} (3.14159\, \text{V})$, so guess of ‘1’ for bit1 of DAC was incorrect. Reset this bit to ‘0’.
  4. Set last bit of DAC to ‘1’. DAC input = 0b1101 = 13, so $V_{\text{dac}} = 13/16 \times 4\, \text{V} = 3.25\, \text{V}$. $V_{\text{dac}} (3.25\, \text{V}) > V_{\text{in}} (3.14159\, \text{V})$, so guess of ‘1’ for bit0 of DAC was incorrect. Reset this bit to ‘0’.

- Final ADC output code is 0b1100.
- Check result: output code = $V_{\text{in}}/V_{\text{ref}} \times 2^N = 3.14159/4 \times 16 = 12.57 = 12$ (truncated).

**A 2-bit Flash ADC**

![A 2-bit Flash ADC Diagram](image)

- Fast, conversion time is settling time of comparators and digital logic.
A 3-bit Flash ADC

ADC Architecture Summary

- **Flash** ADCs
  - Fastest possible conversion time
  - Requires the most transistors of any architecture
  - N-bit converter requires $2^N - 1$ comparators.
  - Commercially available flash converters up to 12 bits.
  - Conversion done in one clock cycle

- **Successive approximation** ADCs
  - Use only one comparator
  - Take one clock cycle per bit
  - High precision (16-bit converters are available)
Commercial ADCs

- Key timing parameter is conversion time – how long does it take to produce a digital output once a conversion is started.
- Up to 16-bit ADCs available.
- Separated into fast/medium/low speed families.
  - Serial interfaces common on medium/low speed ADCs.
- For high-precision ADCs, challenge is keeping system noise from affecting conversion.
  - Assume a 16-bit DAC, and a 4.1V reference, then 1 LSB = 4.1/2^{16} = 62 \mu V.

Flash DAC

N-bit DAC requires 2^N resistors!

Eliminates large capacitive load at one node.
### R-2R Ladder DAC

Resistor ladder divides the Vref voltage to a binary weighted value 4-bit value, with the 4-bits equal to X3 X2 X1 X0.

If the switch Xn is connected to Vref, then that bit value is ‘1’, if the switch Xn is not connected to Vref, then that bit value is ‘0’.

Majority of DACs use this architecture as requires far less resistors than flash DACs.

#### Sample DAC Computations

If Vref = 5V, and the 8-bit input code is 0x8A, what is the DAC output voltage?

\[
\text{input\_code} / 2^N \times Vref = (0x8A) / 2^8 \times 5 \text{ V} = 138/256 \times 5 \text{ V} = 2.70 \text{ V (Vout)}
\]

If Vref = 4V, and the DAC output voltage is 1.25 V, what is the 8-bit input code?

\[
\text{Vout} / Vref \times 2^N = 1.25 \text{ V} / 4 \text{ V} \times 2^8 = 0.3125 \times 256 = 80 = 0x50 \text{ (input\_code)}
\]
Commercial DACs

- Either voltage or current DACs
  - Current DACs require an external operational amplifier to convert to voltage
- Precision up to 16 bits
- Key timing parameter is *settling time* - amount of time it takes to produce a stable output voltage once the input code has changed
- We will use an 8-bit voltage DAC with a SPI interface from Maxim semiconductor

PIC24 ADC

- The PIC24 μC has an onboard ADC
  - Successive approximation
  - 10-bit (default) or 12-bit resolution
  - Reference voltage can be Vdd or separate voltage (min AVSS + 2.7 V)
  - Multiple input (more than one input channel)
  - Clock source for ADC is either a divided Fosc, or an internally generated clock. The ADC clock period (Tad) cannot be less than 76 ns for 10-bit mode, or 118 ns for 12-bit mode. The internally generated clock has a period of ~ 250 ns (~ 4 MHz).
Note that different ANx inputs are mapped to different 'Channels', so have to select both a Channel and an ANx input.

Conversion Time

- Total conversion time is sampling time + conversion time
- Sampling looks at the input voltage and uses a storage capacitor to acquire the input.
  - This time is configurable; we will use a conservative 31 Tad periods which is the maximum for the PIC24.
- Conversion time is Number of bits + 31 Tad periods.
- So, for these settings, takes 31 (sampling) + 12 (bits) + 2 = 45 clock periods.
- Using the internal clock (250 ns), one conversion takes about 11.25 µs (88.9 kHz).
Voltage References

Stability of voltage reference is critical for high precision conversions. We will use Vdd as our voltage reference for convenience, but will be throwing away at least two bits of precision due to Vdd fluctuations.

Example Commercial voltage reference: 2.048v, 2.5v, 3v, 3.3v, 4.096v, 5v (Maxim 6029). The PIC24 can only use a voltage reference of either 3.0 V or 3.3 V.

Key parameter for a voltage is stability over temperature operating range. Need this to be less than \( \frac{1}{2} \) of a LSB value.

Configuring the ADC

```c
void configADC1_ManualCH0(uint16_t u16_ch0PositiveMask,  
  uint8_t u8_autoSampleTime,  
  uint8_t u8_use12bit) {
  if (u8_autoSampleTime > 31) u8_autoSampleTime=31;
  AD1CON1bits.ADON = 0;  // turn off ADC
  /** Configure the internal ADC **/  
  AD1CON1 = ADC_CLK_AUTO | ADC_AUTO_SAMPLING_OFF;
  if (u8_use12bit)  
    AD1CON1bits.AD12B = 1;
  else
    AD1CON1bits.AD12B = 0;
  AD1CON3 = ADC_CONV_CLK_INTERNAL_RC | (u8_autoSampleTime<<8);  
  AD1CON2 = ADC_VREF_AVDD_AVSS;
  AD1CHS0 = ADC_CH0_NEG_SAMPLEA_VREFN | u16_ch0PositiveMask;
  AD1CON1bits.ADON = 1;  //turn on the ADC
}
```
Configuring the ADC (cont.)

- Configures for internal ADC clock, uses manual sample start/auto conversion
- `u16_Ch0PositiveMask` selects the ANx input from Channel 0 to convert
- Uses AVDD, AVSS as references
- Parameter `u8_autoSampleTime` sets the number of sample clocks
- `u8_Use12bits` determines if 12-bit or 10-bit conversion is done

Starting a Conversion, Getting result:

```c
uint16_t convertADC1(void) {
    uint8_t u8_wdtState;

    sz_lastTimeoutError = "convertADC1()";
    u8_wdtState = _SWDTEN; // save WDT state
    _SWDTEN = 1; // enable WDT since we block
    SET_SAMP_BIT_ADC1(); // start sampling
    NOP(); // takes one clock to clear previous
            // DONE flag, delay before checking.
    WAIT_UNTIL_CONVERSION_COMPLETE_ADC1(); // wait for conversion to finish
    _SWDTEN = u8_wdtState; // restore WDT
    sz_lastTimeoutError = NULL; // reset error message
    return(ADC1BUF0);
}
```

- In this mode, tell ADC to start sampling, after sampling is done the ADC conversion is started
  - A status bit is set when the conversion is finished
**Max548A SPI Command Format**

First Byte: DAC command byte

Second Byte: Command data

Command byte to do conversion: 0x09 (0b00001001)

Data is the value to convert.
Function for doing a DAC conversion

```c
#define CONFIG_SLAVE_ENABLE() CONFIG_RB3_AS_DIG_OUTPUT()
#define SLAVE_ENABLE() _LATB3 = 0 // low true assertion
#define SLAVE_DISABLE() _LATB3 = 1

void writeDAC (uint8_t dacval) {
    SLAVE_ENABLE(); // assert Chipselect line to DAC
    ioMasterSPI1(0b00001001); // control byte that enables DAC A
    ioMasterSPI1(dacval); // write DAC value
    SLAVE_DISABLE();
}
```

Testing the ADC and DAC

- Read the voltage from the potentiometer via the PIC24 ADC, write this digital value to the DAC
- The DAC output voltage should match the potentiometer voltage

The MAX548 is an 8-bit DAC with a SPI port

Potentiometer has three pins - middle pin is the wiper, connect the end pins to Vdd/Gnd (ordering does not matter).
**Potentiometer**

A variable resistor. Tie outer two legs to Vdd/GND. Voltage on middle leg will vary between Vdd/GND as potentiometer is adjusted, changing the position of the wiper on the resistor.

---

**adc_spidac_test.c**

```c
void configDAC() {
    // chip select for DAC
    CONFIG_SLAVE_ENABLE();
    SLAVE_DISABLE();
}

int main (void) {
    uint16_t u16_adcVal;
    uint8_t u8_dacVal;
    float f_adcVal;
    float f_dacVal;
    // chip select for DAC
    configDAC();

    configBasic(HELLO_MSG);
    CONFIG_AN0_AS_ANALOG();
    configADC1_ManualCH0(ADC_CH0_POS_SAMPLEA_AN0, 31, 1);
    configSPI1();
    configDAC();
```

- Use input AN0 on Channel 0 as ADC input
- Number of sampling periods, 31
- Support function, configures for manual sampling, auto conversion
- Value of ‘1’ selects 12-bit mode, ‘0’ selects 10-bit mode.
### Lecture 7-39

#### adc_spidac_test.c (cont)

```c
while (1) {
    u16_adcVal = convertADC1(); // get ADC value
    u8_dacVal = (u16_adcVal >> 4) & 0x00FF; // upper 8 bits to DAC value
    writeDAC(u8_dacVal);
    f_adcVal = u16_adcVal;
    f_adcVal = f_adcVal / 4096.0 * VREF; // convert to float 0.0 to VREF
    f_dacVal = u8_dacVal;
    f_dacVal = f_dacVal / 256.0 * VREF;
    printf("ADC in: %4.3f V (0x%04x), To DAC: %4.3f V (0x%02x)n", 
           (double) f_adcVal, u16_adcVal, (double) f_dacVal, u8_dacVal);
    DELAY_MS(300); // delay so that we do not flood the UART.
} // end while(1)
```

- **u16_adcVal** is 12-bit ADC value
- **f_adcVal** is **u16_adcVal** converted to a voltage (0-3.3V) using a float data type.
- **u8_dacVal** is the 8-bit value to send to the DAC (upper 8 bits of **u16_adcVal**)
- **f_dacVal** is **u8_dacVal** converted to a voltage (0-3.3V) using a float data type

### Program Output

- 12-bit ADC code as voltage
- 8-bit DAC code as voltage

---

Electrical & Computer Engineering – Microcomputers  Lecture 7-39

Electrical & Computer Engineering – Microcomputers  Lecture 7-40

20
What do you have to know?

- Vocabulary
- DAC R/2R architecture
- ADC Flash, Successive approximation architectures
- PIC24 ADC
  - How to configure
  - Acquisition, Conversion time
  - How to start do conversion, read result
- MAX548A DAC usage