VE373 Recitation Class

Week 6

2022.06.18

L9 — Output Compare and PWM

1. **PWM Signal (Pulse-width modulation)**

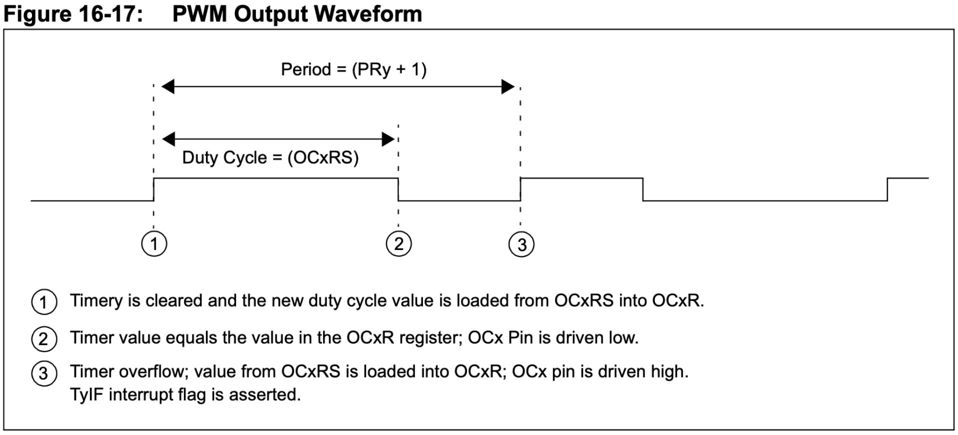
   Information is encoded into the signal width.

   - Advantages of PWM
     - Average value proportional to duty cycle
     - Low power used in transistors used to switch the signal
     - Fast switching possible due to MOSFETS and power transistors at speeds in excess of 100 kHz
     - **Digital signal is resistant to noise less**
     - Heat dissipated versus using resistors for intermediate voltage values
   - Disadvantages of PWM
     - Cost Complexity of circuit
     - Radio Frequency Interference
     - Voltage spikes
     - Electromagnetic noise

2. **PWM Operation Mode**

   In PWM mode, the `OCxR` register is a **read-only** slave duty cycle register and `OCxRS` is a buffer register that is written by the user to update the PWM duty cycle.

   Figure 16-17: PWM Output Waveform

3. **Calculations**

$$T_{\text{PWM}} = (PR + 1) \times T_{\text{PB}} \times \text{Timer Prescale Value}$$
$$2^{\text{PWMResolution}} = \frac{T_{\text{PWM}}}{T_{\text{timer}}}$$
$$\text{PWMResolution (bits)} = \log_2\left(\frac{F_{\text{PB}}}{F_{\text{PWM}} \times \text{Timer Prescale Value}}\right)$$

4. **Comfiguration**

   1. Set the PWM period by writing to the selected timer period register (`PRy`).
   2. Set the PWM duty cycle by writing to the `OCxRS` register.

3. **Write the `OxCR` register with the initial duty cycle.**
4. Enable interrupts, if required, for the timer and Output Compare modules. The output compare interrupt is required for PWM Fault pin utilization.
5. Configure the Output Compare module for one of two PWM Operation modes by writing to the Output Compare mode bits, `OCM<2:0>` (`OCxCON<2:0>`).
6. Set the `TMRy` prescale value and enable the time base by setting `TON` (`TxCON<15>`) `= 1`.

L10 — Analog-to-Digital Conversion (ADC)

1. **Nyquist Theorem**

   A bandlimited analog signal that has been sampled can be perfectly reconstructed from an infinite sequence of samples if the sampling rate f_s exceeds $2f_{max}$ samples per second, where f_{max} is the highest frequency in the original signal.

   In short:

   $$f_{\text{sampling}} > 2f_{\text{original}}$$

2. **Aliasing**

   Aliasing is when the digital signal appears to have a different frequency than the original analog signal.

   If the analog signal does contain frequency components larger than $f_s/2$, then there will be an aliasing error.

   Aliasing example:

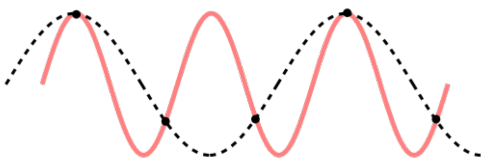
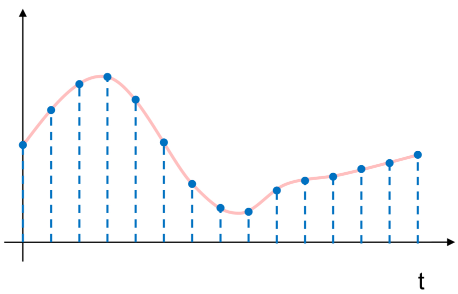


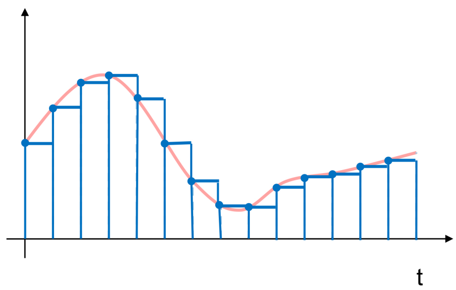
Figure 1: The samples of two sine waves can be identical when at least one of them is at a frequency above half the sample rate.

3. **Steps to convert analog signal to digital signal**

   1. **Sampling & Holding**



(a) Sampling



(b) Holding

   2. **Quantization**
      - Separating the input signal into a discrete states with K increments
      - $K = 2^N$
      - N is the number of bits of the ADC

3. **Coding**

   Assigning a unique digital code to each state for input into the microprocessor

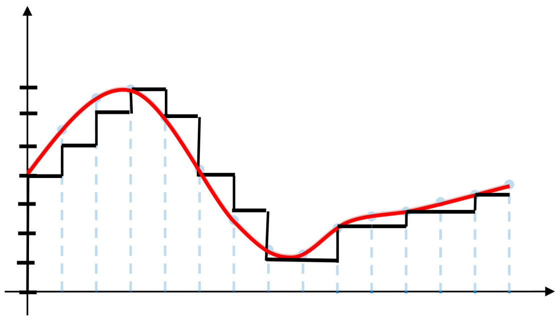


Figure 3: Quantization and Coding

4. **Resolution**

   Amount of information the digital code is able to represent

$$Q = \frac{V_{\text{max}} - V_{\text{min}}}{2^N}$$

   Larger range means lower resolution given fixed number of bits

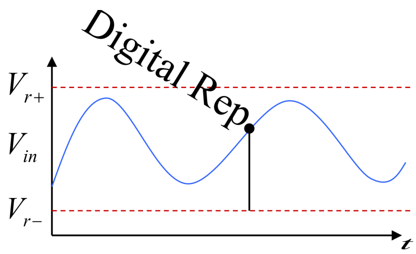
5. **Quantization error**

   How far off discrete value is from actual value

$$\text{Quantization error} = \frac{1}{2}Q$$

   Larger range means larger quantization error

6. **Digital representation**



$$\text{Digital Rep} = 2^N \cdot \frac{V_{in} - V_{r-}}{V_{r+} - V_{r-}}$$

$$V_{in} = \text{Digital Rep} \cdot \frac{V_{r+} - V_{r-}}{2^N} + V_{r-}$$

7. **ADC implementation**

   1. **Method 1: Voltage Divider**

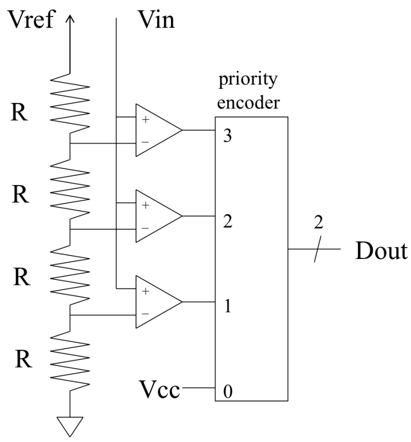


Figure 4: Voltage Divider

2. **Method 2: Successive Approximation**
   Most used in PIC MCUs.
   Example: 4-bit resolution, $2^4 = 16$ discrete representations
   - B3 = 1 (upper half of 1/2)
   - B2 = 0 (lower half of 3/4)
   - B1 = 0 (lower half of 5/8)
   - B0 = 1 (upper half of 9/16)

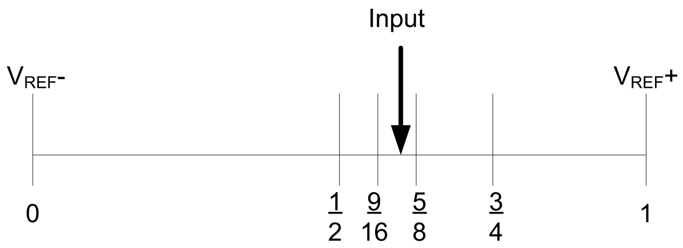
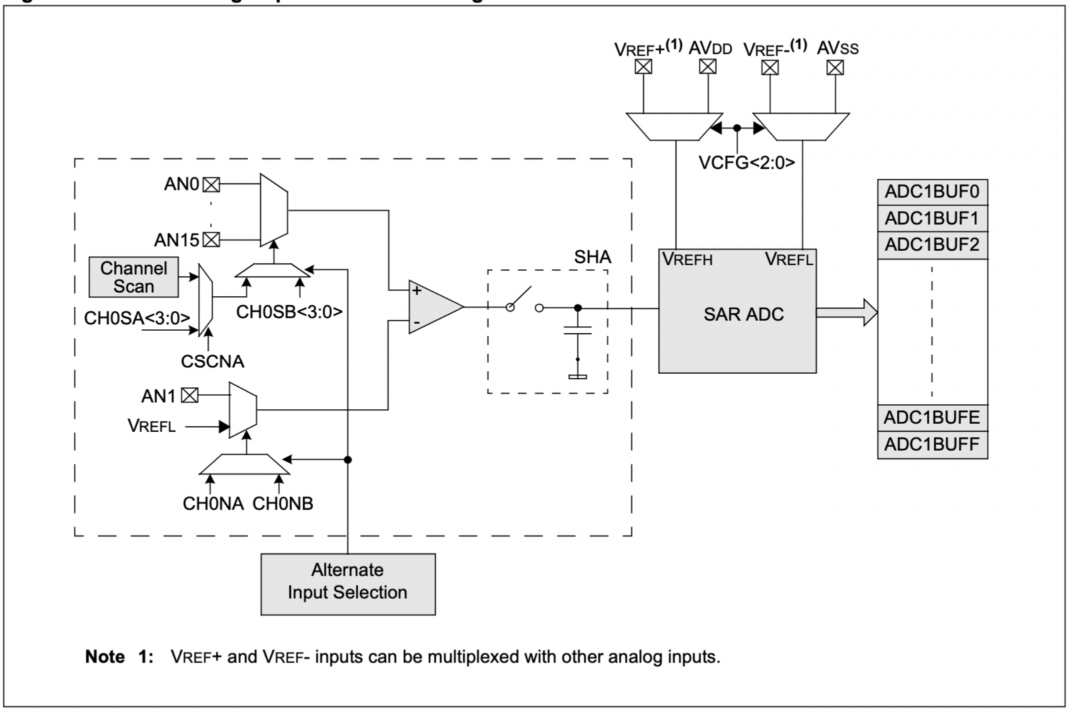


Figure 5: Successive Approximation

8. **ADC of PIC32**
   - 10 bits, i.e. $N = 10$.
   - 16 analog input channels
   - 8 conversion result formats
   - Up to 16 words storage for converted digital value
   - Selectable conversion trigger source
9. **ADC Block Diagram**

Figure 17-1: 10-bit High-Speed ADC Block Diagram



10. **ADC time taken**

    1. **Acquisition**
       Analog signal sampled and hold
       SHA: Sample and Hold Amplifier
       Enough time must be given, variable
       Set by `SAMC` (`AD1CON3<12:8>`)
    2. **Conversion**
       Sampled analog signal converted to digital
       Requires $12T_{AD}$ time to finish
       - One T_{AD} per bit + 2 additional T_{AD}
       - T_{AD} is configurable, > 65 ns

       T_{AD} source can be selected from internal RC source or prescaled `PBCLK` by configuring `ADRC` (`AD1CON3<15>`)

Figure 17-2: ADC Sample/Conversion Sequence

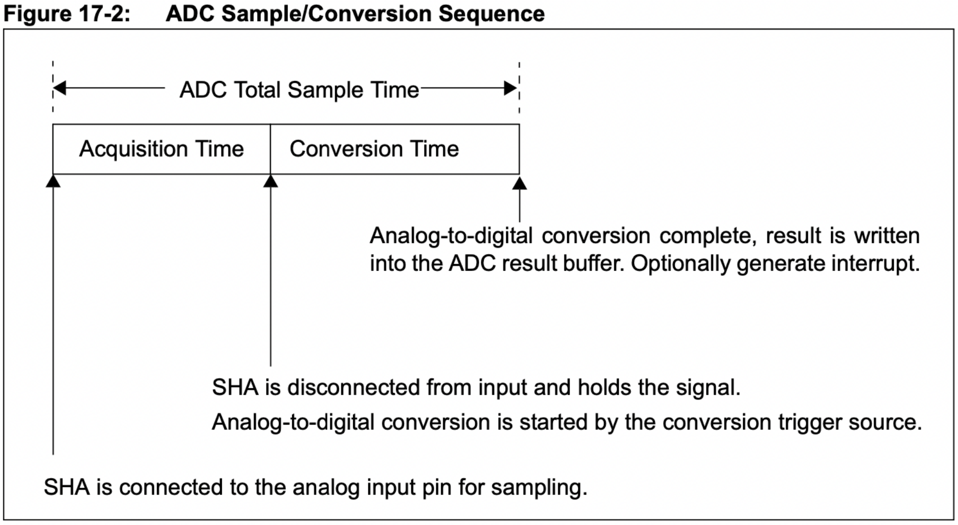
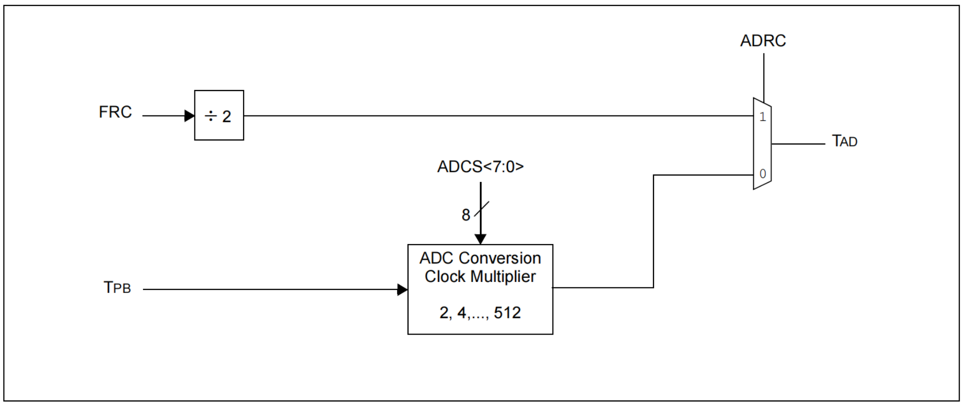


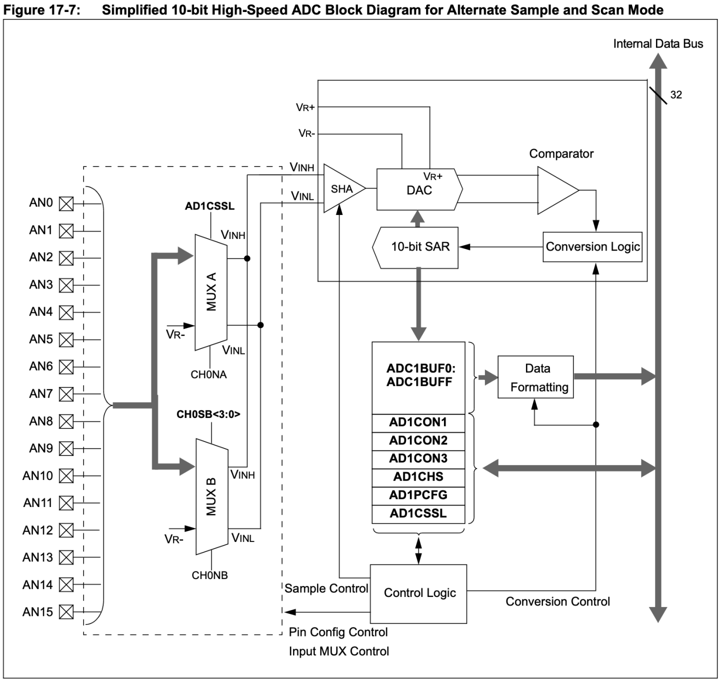
FIGURE 22-2: ADC CONVERSION CLOCK PERIOD BLOCK DIAGRAM



11. ADC Modes

- Manual
  - Manually start acquisition
  - Manually start conversion
- Automatic
  - Continuous conversion sequence
  - Multiple sample and conversions
- Scan mode
  - Scan through multiple inputs
  - Inputs selectable
  - MUX A only
- Alternate scan mode
  - MUX A and MUX B
  - Input selectable

Figure 17-7: Simplified 10-bit High-Speed ADC Block Diagram for Alternate Sample and Scan Mode



12. ADC Interrupt

Configured by `SMPI<2:0>` (`AD1CON2<5:2>`). Can generate interrupt after 1 – 16 samples.

The next sequence starts filling the buffer from the top even if the number of samples in the previous sequence was less than 16.

13. **ADC Setup**

    1. Configure the analog port pins in `AD1PCFG<15:0>`.
    2. Select the analog inputs to the ADC multiplexers in `AD1CHS<32:0>`.
    3. Select the format of the ADC result using `FORM<2:0>` (`AD1CON1<10:8>`).
    4. Select the sample clock source using `SSRC<2:0>` (`AD1CON1<7:5>`).
    5. Select the voltage reference source using `VCFG<2:0>` (`AD1CON2<15:13>`).
    6. Select the Scan mode using `CSCNA` (`AD1CON2<10>`).
    7. Set the number of conversions per interrupt `SMP<3:0>` (`AD1CON2<5:2>`), if interrupts are to be used.
    8. Set Buffer Fill mode using `BUFM` (`AD1CON2<1>`).
    9. Select the MUX to be connected to the ADC in `ALTS` `AD1CON2<0>`.
    10. Select the ADC clock source using `ADRC` (`AD1CON3<15>`).
    11. Select the sample time using `SAMC<4:0>` (`AD1CON3<12:8>`), if auto-convert is to be used.
    12. Select the ADC clock prescaler using `ADCS<7:0>` (`AD1CON3<7:0>`).
    13. Turn the ADC module on using `AD1CON1<15>`.
    14. To configure ADC interrupt (if required):
        a) Clear the `AD1IF` bit (`IFS1<1>`).
        b) Select ADC interrupt priority `AD1IP<2:0>` (`IPC<28:26>`) and subpriority `AD1IS<1:0>` (`IPC<24:24>`) if interrupts are to be used.
    15. Start the conversion sequence by initiating sampling.
        - *Manual mode*: setting `1` to `SAMP` bit (`AD1CON1<1>`).
          Software must manually manage the start and end of the acquisition period by setting and then clearing the SAMP bit after the desired acquisition period has elapsed.
        - *Auto-sample mode*: settings `1` to `ASAM` bit (`AD1CON1<2>`)
          Acquisition is automatically started after a conversion is completed. Auto-Sample mode can be used with any trigger source other than manual.

    Note: Steps 1 through 12, above, can be performed in any order, but Step 13 must be the final step in every case.

14. **DONE bit**

    The `DONE` bit (`AD1CON1<0>`) is set when a conversion sequence is complete.

    - In Manual mode:
      The `DONE` bit is persistent.
      Remains set until it is cleared by software.
      Can be polled to determine when the conversion has completed.
    - In all automatic sample modes (`ASAM` bit = 1):
      the DONE bit is not persistent.
      Set at the end of a conversion sequence and cleared by hardware when the next acquisition is started.
      Polling the DONE bit is not recommended when operating the ADC in automatic modes.
      The `AD1IF` flag bit (`IFS1<1>`) is latched after a conversion sequence is completed and can therefore be polled.