Analog to Digital Conversion

Amplitude Resolution

The smallest change in the input signal that can be detected or distinguished by the system.

Amplitude Range

The range of input values that the system can measure, from the smallest to the largest.

Amplitude Precision

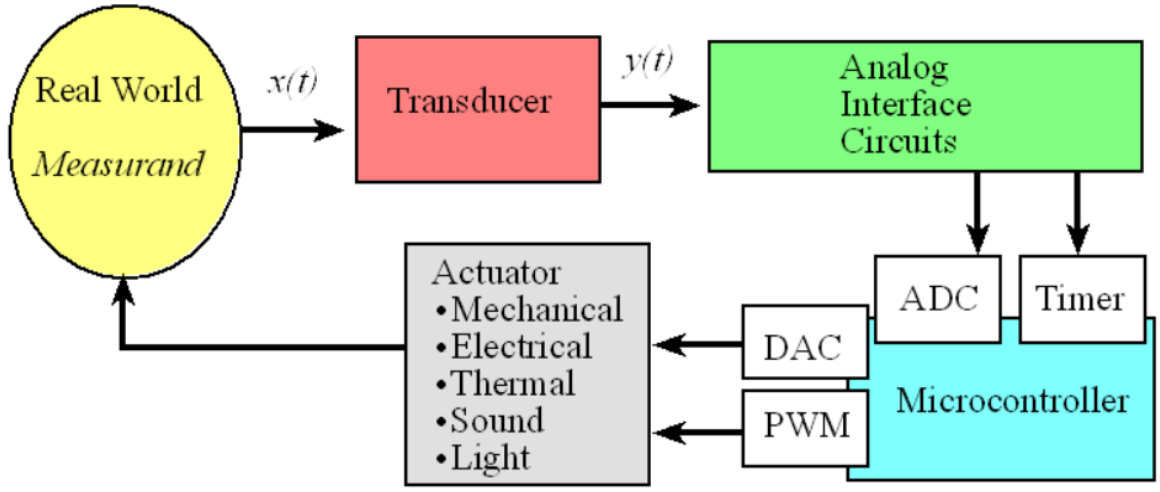
The number of distinct values that the system can represent for a measurement.

Time Quantization

The time difference between consecutive samples in a system. determines how frequently the system captures data, affecting the ability to capture fast-changing signals.

Time Interval

The total duration over which samples are collected. defines the window of observation for the measurand.



The ADC **range** is the maximum and minimum ADC input (e.g., 0 to +3.3V)

The ADC **resolution** is the smallest distinguishable change in input (e.g., 3.3V/4095, which is about 0.81 mV).

Range(volts) = Precision(alternatives) * Resolution(volts)

The **Successive Approximation** technique is one of the most widely used methods for ADC conversion

Process:

- For each clock, the hardware issues a new "guess" on Vdac by setting the bit under test to **1**.
- If Vdac is higher than the unknown input Vin, the bit under test is cleared (set to **0**).
- If Vdac is less than Vin, the bit under test remains **1**.

A 12-bit ADC requires 12 clock cycles to complete the conversion.

Start with the most significant bit (MSB) and work down to the least significant bit (LSB).

ADC0_PC_R Register:

- Specifies the **maximum sampling rate** (see Table 14.2).
- Not the actual sampling rate, but the maximum possible.

Value	*Description*
0x7	1M samples/second
0x5	500K samples/second
0x3	250K samples/second
0x1	125K samples/second

2. **Sampling Rate**:
   - In this chapter, the actual sampling rate is determined by the **SysTick periodic interrupt rate**.
   - The SysTick ISR (Interrupt Service Routine) will take one ADC sample.

Table 13-2. Samples and FIFO Depth of Sequencers

Sequencer	Number of Samples	Depth of FIFO
SS3	1	1
SS2	4	4
SS1	4	4
SS0	8	8

3. **Power and Accuracy**:
   - Running at **125 kHz maximum mode** (ADC0_PC_R = 1) is more accurate and consumes less power compared to **1 MHz maximum mode**.

The ADC has four sequencers , To prioritize sequencers:

- Set **ADC0_SSPRI_R = 0x0123**, making sequencer 3 the highest priority.
- Ensure all sequencers have unique priorities.

Trigger settings are defined in the **ADC0_EMUX_R register**:

- **Bits 15–12 (EM3)** control the trigger mechanism.
- **EM3 = 0x0** configures software-triggered conversion.

Start conversion by writing **8 (SS3)** to the **ADC0_PSSI_R register**.

Use **ADC0_ACTSS_R** to enable or disable specific sequencers.

TM4C123 has **12 ADC channels** for different input pins, configured via **ADC0_SSMUX3_R**.

 ADC**0_SSPRI_R (Sequencer Priority Register)**

- Sets priority for the four sequencers.
- Example: 0x0123 makes **sequencer 3** the highest priority.
- Ensures no conflicts between sequencers.

 ADC0_EMUX_R (Event Multiplexer Register)

- Configures the trigger source for ADC conversions.
- **Bits 15–12 (EM3)**:
    - 0x0 for software trigger.
    - Other values allow hardware triggers (e.g., timer, PWM).

 ADC0_PSSI_R (Processor Sample Sequencer Initiate Register)

- Used to start a conversion on a sequencer.
- Writing **8 (SS3)** initiates conversion on sequencer 3.

 ADC0_ACTSS_R (Active Sample Sequencer Register)

- Enables or disables sequencers.
- To use a sequencer, set the corresponding bit.
- Example: **Bit 3** for sequencer 3.

 ADC0_SSMUX3_R (Sample Sequencer Input Multiplexer Select Register)

- Selects the ADC channel to sample.
- Example: Writing 9 selects **Channel 9 (PE4)**.

 ADC0_SSCTL3_R (Sample Sequencer Control Register)

- Configures sample behavior for sequencer 3.
- **Key bits**:
  - **TS0**: Set for temperature measurement; clear for voltage input.
  - **IE0**: Enable interrupt when the sample is complete.
  - **END0**: Marks the end of the sequence (always set for sequencer 3).
  - **D0**: Set for differential sampling; clear for single-ended.

 ADC0_RIS_R (Raw Interrupt Status Register)

- **INR3**: Flag set when conversion is complete on sequencer 3.
- Check this flag to know when the result is ready.

 ADC0_ISC_R (Interrupt Status and Clear Register)

- Clears the interrupt flag (INR3).
- Write **8** to clear the sequencer 3 interrupt.

Channel 9 is connected to **Pin PE4**.

IO	Ain	0
PB4	Ain10	Port
PB5	Ain11	Port
PD0	Ain7	Port
PD1	Ain6	Port
PD2	Ain5	Port
PD3	Ain4	Port
PE0	Ain3	Port
PE1	Ain2	Port
PE2	Ain1	Port
PE3	Ain0	Port
PE4	Ain9	Port
PE5	Ain8	Port

e3210,d3210,e54,b54

Step 1. We enable the ADC clock bit 0 in **SYSCTL_RCGCADC_R** for ADC0.

Step 2. We enable the port clock for the pin that we will be using for the ADC input.

Step 3. We wait for the two clocks to stabilize (some people found extra delay prevented hard faults)

Step 4. Make that pin an input by writing zero to the **DIR** register.

Step 5. Enable the alternative function on that pin by writing one to the **AFSEL** register.

Step 6. Disable the digital function on that pin by writing zero to the **DEN** register.

Step 7. Enable the analog function on that pin by writing one to the **AMSEL** register.

Step 8. We set the **ADC0_PC_R** register specify the maximum sampling rate of the ADC. In this example, we will sample slower than 125 kHz, so the maximum sampling rate is set at 125 kHz. This will require less power and produce a longer sampling time, creating a more accurate conversion.

Step 9. We will set the priority of each of the four sequencers. In this case, we are using just one sequencer, so the priorities are irrelevant, except for the fact that no two sequencers should have the same priority.

Step 10. Before configuring the sequencer, we need to disable it. To disable sequencer 3, we write a 0 to bit 3 (**ASEN3**) in the **ADC0_ACTSS_R** register. Disabling the sequencer during programming prevents erroneous execution if a trigger event were to occur during the configuration process.

Step 11. We configure the trigger event for the sample sequencer in the **ADC0_EMUX_R** register. For this example, we write a 0000 to bits 15–12 (**EM3**) specifying software start mode for sequencer 3.

Step 12. Configure the corresponding input source in the **ADC0_SSMUX3** register. In this example, we write the channel number to bits 3–0 in the **ADC0_SSMUX3_R** register. In this example, we sample channel 9, which is PE4.

Step 13. Configure the sample control bits in the corresponding nibble in the **ADC0_SSCTL3** register. When programming the last nibble, ensure that the **END** bit is set. Failure to set the **END** bit causes unpredictable behavior. Sequencer 3 has only one sample, so we write a 0110 to the **ADC0_SSCTL3_R** register. Bit 3 is the **TS0** bit, which we clear because we are not measuring temperature. Bit 2 is the **IE0** bit, which we set because we want to the **RIS** bit to be set when the sample is complete. Bit 1 is the **END0** bit, which is set because this is the last (and only) sample in the sequence. Bit 0 is the **D0** bit, which we clear because we do not wish to use differential mode.

END0 Must Be Set: Without it, the ADC may attempt to read further samples that don't exist, causing errors.

Interrupt Handling: Setting **IE0** allows the program to monitor the conversion status via the **RIS** flag.

Step 14. Disable interrupts in ADC by clearing bits in the **ADC0_IM_R** register. Since we are using sequencer 3, we disable SS3 interrupts by clearing bit 3.

Step 15. We enable the sample sequencer logic by writing a 1 to the corresponding **ASEN3**. To enable sequencer 3, we write a 1 to bit 3 (**ASEN3**) in the **ADC0_ACTSS_R** register.

```
void ADC0_Init(void) {
 SYSCTL_RCGCADC_R |= 1; // Enable clock to ADC0
SYSCTL_RCGCGPIO_R |= 0x10; // Enable clock to GPIOE
 GPIO_PORTE_AFSEL_R |= 0x10; // Enable alternate function on PE4
GPIO_PORTE_DEN_R &= ~0x10; // Disable digital I/O on PE4
 GPIO_PORTE_AMSEL_R |= 0x10; // Enable analog function on PE4
ADC0_ACTSS_R &= ~0x08; // Disable sequencer 3 during configuration
ADC0_EMUX_R = (ADC0_EMUX_R & ~0xF000) | 0x0000; // Set EM3 to software start
ADC0_SSPRI_R = 0x0123; // Set sequencer 3 as highest priority
ADC0_SSMUX3_R = 9; // Select channel 9 (PE4)
ADC0_SSCTL3_R = 0x0006; // Set IE0 and END0, single-ended input
ADC0_ACTSS_R |= 0x08; // Enable sequencer 3 }

uint16_t ADC0_Read(void) {
ADC0_PSSI_R = 0x08; // Start conversion on sequencer 3
 while ((ADC0_RIS_R & 0x08) == 0); // Wait for conversion to complete
uint16_t result = ADC0_SSFIFO3_R & 0xFFF; // Read 12-bit result
ADC0_ISC_R = 0x08; // Clear completion flag return result; }
```

31-2					1	0	Name
					ADC1	ADC0	SYSCTL_RCGCADC_R
31-14	13-12	11-10	9-8	7-6	5-4	3-2	1-0	
	SS3		SS2		SS1		SS0	ADC0_SSPRI_R

31-16	15-12	11-8	7-4	3-0	
	EM3	EM2	EM1	EM0	ADC0_EMUX_R

31-4	3	2	1	0	
	ASEN3	ASEN2	ASEN1	ASEN0	ADC0_ACTSS_R
	MUX0				ADC0_SSMUX3_R
	TS0	IE0	END0	D0	ADC0_SSCTL3_R
	SS3	SS2	SS1	SS0	ADC0_PSSI_R
	INR3	INR2	INR1	INR0	ADC0_RIS_R
	MASK3	MASK2	MASK1	MASK0	ADC0_IM_R
	Speed				ADC0_PC_R

31-12	11-0	
	DATA	ADC0_SSFIFO3_R

the analog input is 3.3V, the digital output will be 4095.

For ex 1.65 → 3.3

? → 4095

Digital Sample = (Analog Input (volts) • 4095) / 3.3V(volts)

The four steps of analog to digital conversion: 1) initiate conversion, 2) wait for the ADC to finish, 3) read the digital result, and 4) clear the completion flag.

according to the **Nyquist Theorem**, we must sample that signal at

$$f_s > 2f$$

if the analog signal does contain frequency components larger than ½ f_s, then there will be an aliasing error during the sampling process (performed with a frequency of f_s).

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

Central Limit Theorem or clt states that if you have a signal with mean μ and standard deviation σ and take sufficiently large random samples (n>30), calculate the average

there is a mode on the ADC to automatically take multiple samples and return the average. The ADC0_SAC_R register can be any value from 0 to 6. The possible choices are

```
ADC0_SAC_R = 0;  // take one sample
ADC0_SAC_R = 1;  // take 2 samples, return average
ADC0_SAC_R = 2;  // take 4 samples, return average
ADC0_SAC_R = 3;  // take 8 samples, return average
ADC0_SAC_R = 4;  // take 16 samples, return average
ADC0_SAC_R = 5;  // take 32 samples, return average
ADC0_SAC_R = 6;  // take 64 samples, return average
```

SUMMARY :

Registers:

1. **ADC0_PC_R (ADC Sample Rate Control Register)**
    - Sets the **maximum sampling rate** (e.g., 125 kHz or 1 MHz). Lower rates improve accuracy and save power.
2. **ADC0_SSPRI_R (Sequencer Priority Register)**
    - Assigns **priority levels** to the four ADC sequencers. Ensure all sequencers have unique priorities.
3. **ADC0_EMUX_R (Event Multiplexer Register)**
    - Configures the **trigger source** for ADC conversions.
      Example:
        - 0x0 → Software-triggered conversion.
        - Other values → Hardware triggers (e.g., Timer, PWM).
4. **ADC0_PSSI_R (Processor Sample Sequencer Initiate Register)**
    - Starts a conversion on a selected sequencer. Writing 8 initiates sequencer 3.
5. **ADC0_ACTSS_R (Active Sample Sequencer Register)**
    - Enables/disables specific sequencers. Example: Setting bit 3 enables sequencer 3.
6. **ADC0_SSMUX3_R (Sample Sequencer Input Multiplexer Select Register)**
    - Selects the ADC **channel** to sample.
      Example: Writing 9 selects Channel 9 (PE4).
7. **ADC0_SSCTLn (Sample Sequencer Control Register)**

    - **TS (Temperature Sensor Select)**:
      Enables the internal temperature sensor input for a specific sample step.

    - **IE (Interrupt Enable)**:
      Generates an interrupt when the current sample is complete.

    - **END (Sequence End)**:
      Marks the end of the conversion sequence for the specified step.

    - **D (Differential Input Select)**:
      Selects between **single-ended** or **differential** input mode.

8. **ADC0_RIS_R (Raw Interrupt Status Register)**
    - Indicates conversion completion with the **INR3 flag**.
9. **ADC0_ISC_R (Interrupt Status and Clear Register)**

- Clears the **interrupt flag** for sequencer 3.

10. **ADC0_SAC_R (Sample Averaging Control Register)**
    - Configures the ADC to take multiple samples and return the average.
      Values:
      - 0: Single sample.
      - 1: Average of 2 samples.
      - 6: Average of 64 samples.
11. ADC Digital Comparator Range (ADCDCCMPn)
    Defines how the comparator will evaluate ADC results, such as checking if a result is above, below, or within a range.

    Operational Modes

    1.**Always Mode**:
    Interrupt/trigger continuously asserts when ADC value matches the comparison.

    2. **Once Mode**:
    Interrupt/trigger asserts once when ADC value matches the comparison, only if the previous value didn't.

    3. **Hysteresis-Always Mode**:
    Interrupt/trigger keeps asserting while ADC value is in the comparison range until it moves into the opposite band.

    4. **Hysteresis-Once Mode**:
    Interrupt/trigger asserts once when ADC value matches, only if the previous value didn't, and the opposite band was crossed.

12. ADCTSSEL register is programmed to identify in which PWM module

---

FIFO:

- **ADC0_SSFIFO3_R (Sample Sequencer FIFO)**
  - Stores **conversion results** for sequencer 3. The result is 12 bits wide.

Steps to Set Up ADC:

1. **Enable ADC clock**: Set bit 0 in SYSCTL_RCGCADC_R for ADC0.
2. **Enable port clock**: Set the clock for the GPIO port used for the ADC input.
3. **Stabilize clocks**: Wait for clock stabilization.
4. **Configure the input pin**:
   - Set as input by writing 0 to the DIR register.

- Enable the alternative function via AFSEL.
- Disable digital I/O using DEN.
- Enable the analog function via AMSEL.

5. **Set maximum sampling rate**: Use ADC0_PC_R. Example: Set for 125 kHz for high accuracy.
6. **Set sequencer priorities**: Use ADC0_SSPRI_R to ensure unique priorities.
7. **Disable sequencer**: Temporarily disable with ADC0_ACTSS_R.
8. **Set trigger event**: Use ADC0_EMUX_R. Example: Software trigger (0x0).
9. **Configure input source**: Write channel number to ADC0_SSMUX3_R.
10. **Set sample control bits**: Use ADC0_SSCTL3_R. Ensure END bit is set for the last sample.
11. **Disable interrupts**: Clear bit 3 in ADC0_IM_R for sequencer 3.
12. **Enable sequencer**: Use ADC0_ACTSS_R. Set bit 3 for sequencer 3.
13. **Initiate conversion**: Write 8 to ADC0_PSSI_R to start conversion.
14. **Read result**: Access the result from ADC0_SSFIFO3_R.
15. **Clear flag**: Write 8 to ADC0_ISC_R to clear the completion flag.