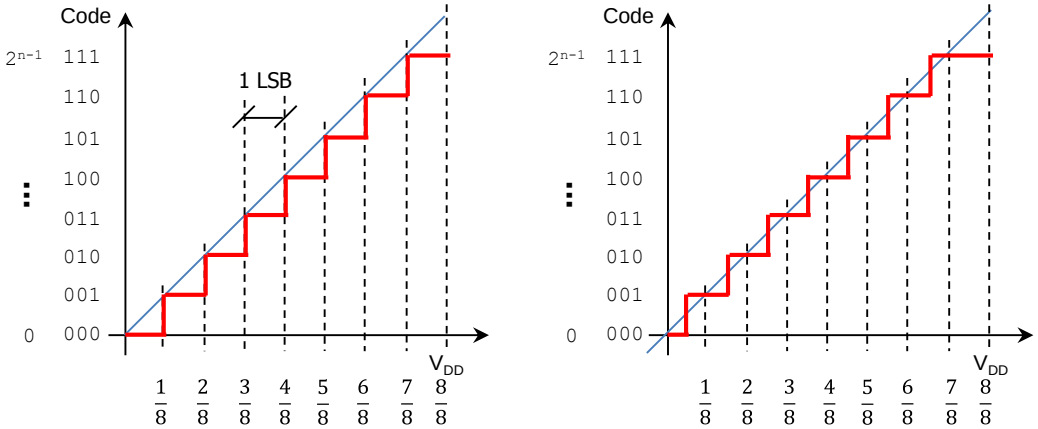
Notes - Unit 11

Analog to Digital Conversion

Process

- Transducer + Signal Conditioning (scaling, voltage translation, etc) + A/D Converter
- Linear Conversion:



- $n-bit$ converter: 2^n code values.
- The figure used values from 0 to VDD. But it can be generalized from V_{RL} (low reference voltage) to V_{RH} (high reference voltage).
- Resolution: It can be defined as the minimum change in voltage required to change the output code level, i.e., the LSB. The resolution is equal to the LSB voltage.
  $Resolution = \frac{V_{DD}}{2^n}$. The larger this value, the lower the resolution.
- Average conversion error: This is the voltage of ½ LSB: $\frac{V_{DD}}{2^{n+1}}$
- Common values for n = 8, 10, 12, 16, 24.

DETERMINE A VOLTAGE VALUE

- The Analog-to-Digital converter provides an n-bit code. We then need to determine the voltage value. There exists many ways to do so. One way is to use the voltage at which the idealized line meets the stair-case pattern for the given code. This works for both of the previous two figures.
  For example: For the figure on the left, a value of '010' resulted from a value between $\frac{2}{8}V_{DD}$ and $\frac{3}{8}V_{DD}$. For the figure on the right, a value of '010' resulted from a value between $\frac{3}{16}V_{DD}$ and $\frac{5}{16}V_{DD}$. In both cases, if we use the idealized line, we will find that '010' is equivalent to 2/8*VDD.
- In general, by using V_{RL} and V_{RH}, and by using two points in the idealized line, we have that:
$$\frac{V_k - V_{R_L}}{k} = \frac{V_{R_H} - V_{R_L}}{2^n} \rightarrow V_k = V_{R_L} + \left(\frac{k}{2^n}\right)\left(V_{R_H} - V_{R_L}\right)$$
- For the particular case in the figure, $V_{R_H} = V_{DD}$, $V_{R_L} = 0$. This results in:
$$V_k = \left(\frac{k}{2^n}\right)V_{DD}$$

A/D Conversion Algorithms

- **Parallel (Flash) A/D Converters:** It requires as many comparators as the number of possible codes (2^n). It also requires a priority encoder with 2^n inputs and n outputs. The conversion speed is high, but the hardware resources grow exponentially with 'n'. They are commonly used in high-speed and low resolution applications.
- **Slope and double-slope A/D Converters:** The charging and discharging of a capacitor is used to perform the A/D conversion. They require relatively simple hardware. They are used in low speed and high resolution applications.
- **Sigma-delta A/D Converters:** Increasingly popular in the implementation of high-resolution A/D converters. The only disadvantage is the slow conversion speed. They are used in low-speed, high resolution applications.
- **Successive approximation A/D Converters:** The analog signal is approximated in 'n' consecutive steps. It offers a good balance between speed and precision. It is one of the most popular A/D conversion methods. The HCS12 uses this technique to implement the A/D converter.

Algorithm:

```
SAR = 00…00
i = n-1
while (i >=0)
    SAR[i] = 1
    Convert SAR value to voltage
    if voltage > Vin then
           SAR[i] = 0
    else
           SAR[i] = 1
    end

    i = i-1
end
```

Example: n=3, V_{DD}=5V. Assume that Vin=3.2
First iteration: b_2=1 → Code: `100`. '100' is equal to 2.5. 2.5 < 3.2 then b_2=1 (keeps value)
Second iteration: b_1=1 → Code: `110`. '110' is equal to 3.75. 3.75 > 3.2 then b_1=0
Third iteration: b_0=1 → Code: `101`. '101' is equal to 3.125. 3.125 < 3.2 then b_0=1 (keeps value)
Final code for 3.2 v = `101`

HCS12: ADC

The HCS12DG256 device includes two A/D modules: ATD0, ATD1. Each A/D module includes:

- Four non-reserved control registers: `ATDnCTL2, ATDnCTL3, ATDnCTL4, ATnCTL5`, n=0,1
- Two status registers: `ATDnSTAT0, ATDnSTAT1`, n=0,1
- One input enable register: `ATDnDIEN` that enables analog pins to be used as digital input
- One port data register: `PORTADn`
- Eight 16-bit result registers: `ATDnDR0,ATDnDR1, ATDnDR2, ATDnDR3, ATDnDR4, ATDnDR5, ATDnDR6, ATDnDR7`. For example: `ATDnDR0`: it is actually `ATD0DR0H|ATDnDR0L`. But we can ready the whole 16-bit register at once.

ATDnCTL2: It controls power down, fast clear, stopping in Wait Mode, Interrupt, and the external trigger.

- External Trigger: Allows the user to synchronize the ATD conversion with external events (low, high levels, edges).

ATDnCTL3: It controls the conversion sequence length, enables/disables FIFO mode for result registers, and controls the ATD behavior in freeze mode.

- Conversion sequence length: Each time we execute an A/D conversion, we can execute up to 8 consecutive conversions (to be stored in `ATDnDR0, …, ATDnDR7`). The bits `ATDnCTL3(6..3)` control the conversion sequence length (from 1 to 8, see Figure 12.10 in textbook).

ATDnCTL4: It controls the ATD clock frequency, the length of the second phase of the sample time, and the resolution of the A/D conversion.

- ATD Resolution: We can pick between 10 bits and 8 bits using `ATDnCTL4(7)`.
- ATD Clock Pre-scaler: The following formula determines the ATD conversion clock.

$$ATDnclock = \frac{E - clock}{PRS + 1} \times 0.5, \qquad PRS = ATDnCTL4(4..0), \qquad n = 0,1$$

The maximum ATD conversion clock is 2MHz and minimum ATD conversion clock is 500 KHz. The value of PRS must be such that the ATDnclock is between 500KHz and 2 MHz.

- The sampling process consists of two phases:
  - ✓ A sample amplifier buffers the input analog signal for two cycles to charge the sample capacitor almost to the input voltage.
  - ✓ The sample buffer is then disconnected and the input signal is directly connected to the storage node for a number of programmable cycles (2,4,8, or 16 using `ADTnCTL4(6..5)`). This is for final charging and high accuracy.
- The conversion time then results from: i) the sampling process time, and ii) the fact that each bit takes one ATD clock cycle to compute:

$$Conversion\ Time = \frac{Resolution\ bits + 2 + programmed\ sample\ clocks}{ATD\ clock\ frequency}$$

ATDnCTL5: It selects the type of conversion sequence (unsigned/signed), justification on the 16-bit result register (left/right), the analog input channels sampled. A write to this register starts an ATD conversion sequence.

- Once the Control Registers are set up, we just need to wait for the SCF (bit 7) of the `ATD0STAT0` register to be '1'. This indicates that the conversion sequence has completed. Also `ATD0STAT1` will let us know which conversion number has completed (can be 1 channel or all the 8 channels).

Example: Using the Trimmer Pot VR2 connected on AN7 (PAD07).
As PAD07 is being used, this is the ATD0 module.

- `ATD0CTL2 = 0xE0`:
  - ✓ Enable ATD0 by setting bit 7 of ATD0CTL2 to '1'.
  - ✓ Enable fast flag clear all: Any access to a result register will the cause the associated CCF flag (I n ATD0STAT1 register) to clear automatically. This is done by setting bit 6 of ATD0CTL2 to '1'. Otherwise, to clear we would need to read ATDnSTAT1 and then read the corresponding result register.
  - ✓ Set ATD0 to stop when in wait mode. This is done by setting bit 5 of ATD0CTL2 to '1'.
  - ✓ Disable external trigger on channel 7. The external trigger allows to synchronize sample and ATD conversions processes with external events. Bits 4,3,2 of ATD0CTL2 to '0'.
  - ✓ Disable ATD0 interrupt. Bits 1,0 of ATD0CTL2 to '0'.

- `ATD0CTL3 = 0x0A`:
  - ✓ Perform 1 conversion per sequence (up to 8 conversions can be performed per sequence (a sequence defined at every time we direct the ATD module to start conversion)): Bits 6-3 are set to "0001".
  - ✓ Disable FIFO mode: bit 2 is 0
  - ✓ When debugging, it is useful to have ATD pause when a breakpoint (freeze) is encountered. We finish the current conversion and then freeze (this is to not to compromise accuracy): bits 1, 0 are "10".

- `ATD0CTL4 = 0x25`
  - ✓ Select 10-bit operation. Bit 7 is '0'.
  - ✓ Sample time set to four A/D clock periods (ATD clock): bit6=0, bit5=1.
  - ✓ Pre-scale value set to 12. Bits 4 to 0 equal to 00101.

- Delay for 20 us: this allows for the ATD to stabilize.

- `ATD0CTL5 = 0x87`: By writing here, we start the A/D conversion sequence.
  - ✓ Result register (16-bits) ATD0DR0 is right-justified. Set bit 7 to '1'.
  - ✓ Result is unsigned. Bit6 =0
  - ✓ Single conversion sequence: bit5=0 (whether conversion sequences are performed continuously or only once).
  - ✓ Single channel mode: bit 4=0
  - ✓ Select channel 7: bits 2 to 0 = 111

- Wait until the SCF flag of `ATD0STAT0` is '1', then get the conversion result.
- Conversion result: k (10-bit number) $\rightarrow$ Quantized voltage: $V_k = \left(\frac{k}{2^{10}}\right) V_{DD},\ V_{DD} = 5v$

Code: `unit11a.c`

Example: Using the Temperature Sensor LM45 on AN5 (PAD05).
As PAD05 is being used, this is the ATD0 module.

- `ATD0CTL2 = 0xE0`
- `ATD0CTL3 = 0x0A`
- `ATD0CTL4 = 0x25`

- Delay for 20 us: this allows for the ATD to stabilize.

- `ATD0CTL5 = 0x85`: By writing here, we start the A/D conversion sequence.
  - ✓ Select channel 5: bits 2 to 0 = 101

- Wait until the SCF flag of `ATD0STAT0` is '1', then get the conversion result.
- On-board (Dragon-Light 12) LM45: 0°C to 100°C, 10mv per °C, thus the output voltage V_o is between 0 and 1 v. And the temperature (°C) is given by: $T = 100 \times V_o$.
- Conversion result: k (10-bit number), $V_{DD} = 5v \rightarrow$ Quantized voltage: $V_k = \left(\frac{k}{2^{10}}\right) V_{DD}$, Temperature (°C): $T = 100 \times V_k$

Code: `unit11b.c`