Lesson 16
Analog-to-Digital Converter (ADC)

1. Overview

In this lesson, the Analog-to-Digital Converter (ADC) of the Cortex-M3 is introduced. For detailed description of the features and controlling options for the ADC, read chapter 29 of the *LPC17xx User Manual*.

2. Background

Most signals in the world exist as continuous functions of time in an analog format (e.g. voltage, current, speed, force, pressure, temperature, sound, colors, etc.) In order to use these signals in the digital domain (store, manipulate/edit), we must approximate the digital (binary) representations of these signals in a discrete fashion. For example, an overview of the ADC process to convert an analog audio signal to digital format is shown in the figure below.

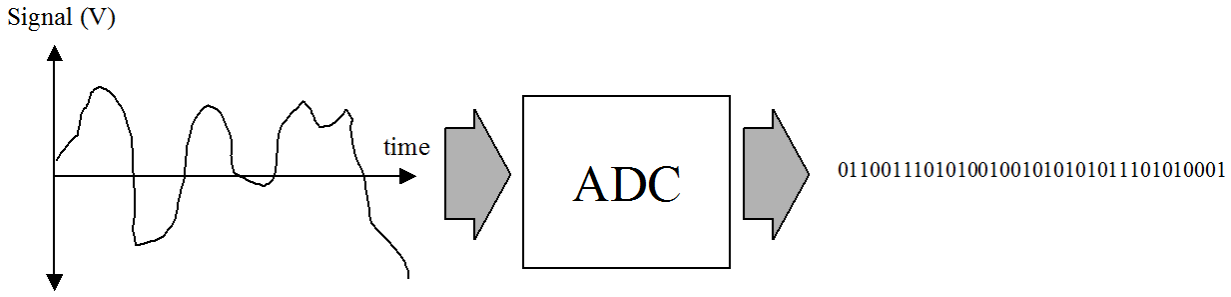

Fig. 1 Overview of the ADC process (from EE334 Thinking Digital Lesson)

A digital representation of the signal can be easily stored, manipulated (edited), transmitted, etc. When an ADC process is utilized, two things need to be considered to convert an analog signal into a digital format:

- Sampling rate: this rate would determine how often we need to record a sample value from the analog signal.
- Quantization: how to divide the analog range into discrete, measured portions. The number of quantization levels would determine the number bits that are required to represent each sample of the signal.

Sampling rate:

The sampling rate is dictated by the highest frequency component of the analog signal. The sampling rate suggested by the **Nyquist Theorem** which states that if a signal is sampled with a frequency of F_s, then the digital samples only contain the frequency components form 0 to ½ F_s. In other words,

Where F_s is the sampling frequency and F_{max} is the highest frequency component of the analog signal. If the sampling rate is too slow (F_s <2 F_{max}), aliasing error can occur. Aliasing is when the digital signal appears to have different frequency than the original analog signal. For example, consider the 7 KHz analog signal below. If the sample rate is 10 KHz (<2 F_{max} *or 14 KHz*), the analog values would be sampled every 100us (1/10 KHz) as shown. Now if we only consider the sample points and try to recreate the analog signals form these samples, we could end up with a 3 KHz signal instead of the original 7 KHz signal.

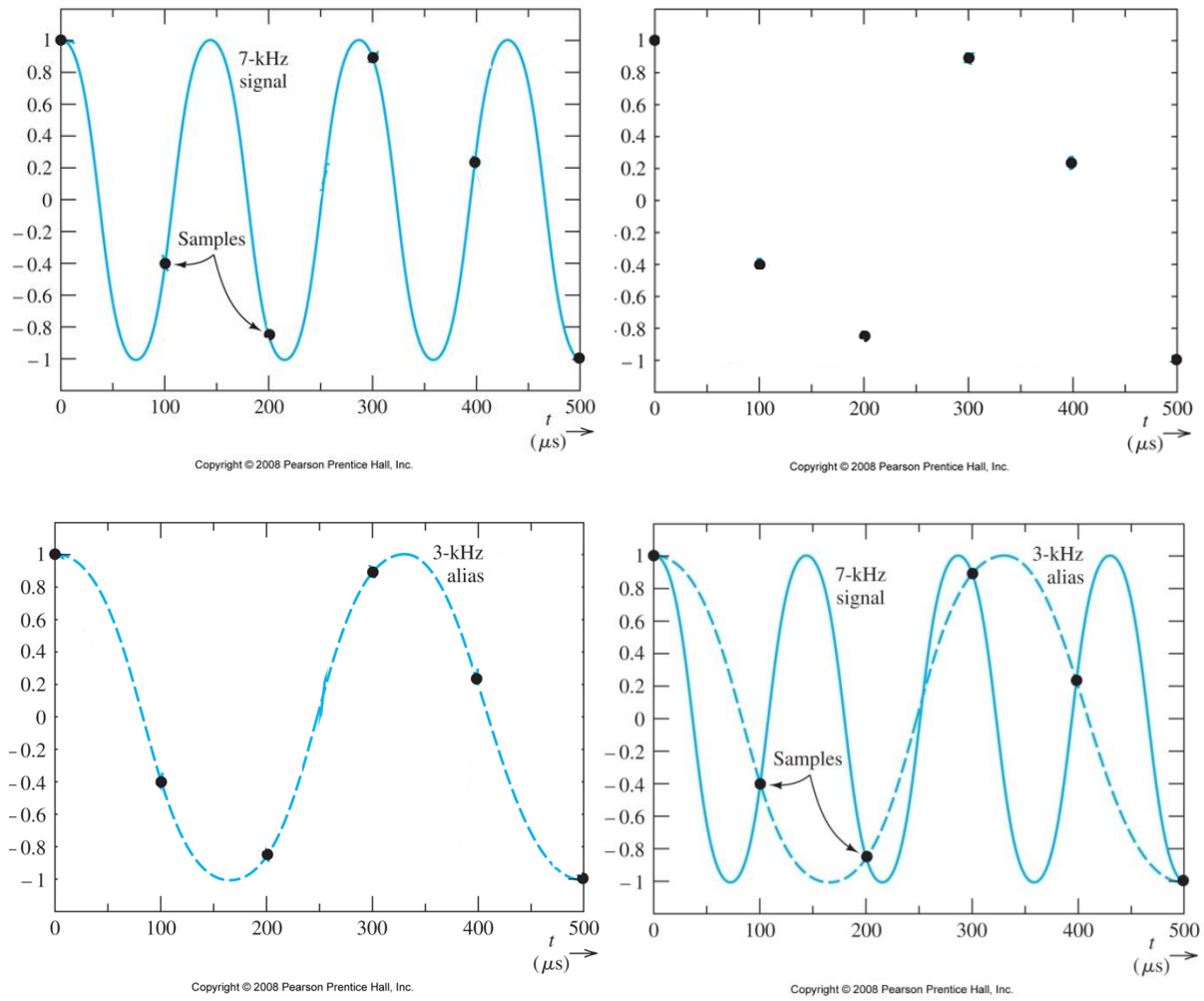

Fig. 2. Illustration of sampling and alias in the ADC process.

Quantization:

At each sampling time, the intensity level of the analog signal must be converted into a binary number. This is known as **quantization**. Because we use discrete levels to represent samples, there will be **quantization error** when an analog signal is reconstructed from its digital form (Digital-to-Analog Conversion or DAC). This error can be significant if the number of bits used to represent the digital samples is low. An illustration of quantization error is shown below.

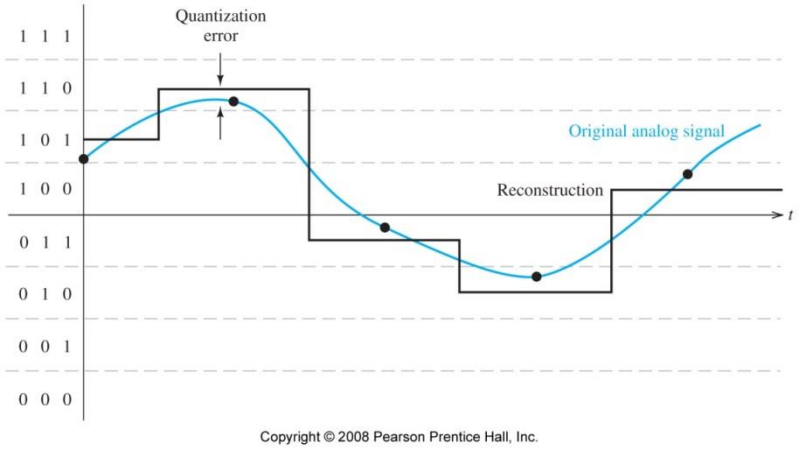

Fig. 3. Quantization error.

Question: What would be the reconstructed signal of the original 7 KHz analog signal in Fig. 1 if 1 bit is used for quantization?

Resolution:

Resolution is the smallest unit of intensity (voltage) that the ADC process can resolve. The resolution is defined as

where

- V_{max} is the maximum voltage allowed in the analog signal.
- V_{min} is the minimum voltage allowed in the analog signal.
- N is the number of bits used in the ADC process voltage.

Question: What is the resolution of the ADC if 1 bit is used to represent sample values of the original 7 KHz analog signal in Fig. 1?

Question: What is the resolution of the ADC for analog signal shown in Fig. 3? Assume that V_{max}=1V, V_{min}=0V

3. Analog-to-Digital Converter

ADC Circuit:

There are several types of circuits that can do ADC. The simplest circuit is to use an opamp comparator. An example of which is shown in Fig. 4.

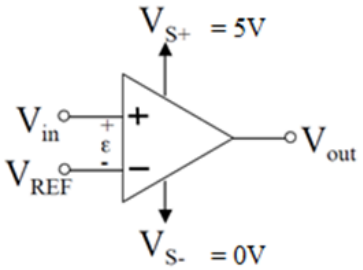

Fig. 4. 1-bit ADC circuit (from EE334 ADC Lesson).

Question: When will the opamp in Fig. 4 have an output of 5V, 0V?

If we consider the opamp output voltage of 5V as bit value '1' and 0V as bit value '0', then we have a 1-bit ADC circuit.

Successive Approximation:

Now, let's consider the block diagram below for a 2-bit ADC circuit. The control logic block sets or clears the output bits (ADC outputs) based on the successive comparison results of the opamp. The Digital-to-Analog Converter (DAC) below generates an output voltage based on the 2-bit input number. A summary of the DAC voltage (V_{DAC}) output is shown in the table below. DAC is another topic that should be discussed in another lesson. In this lesson, we will focus on the analog-to-digital conversion.

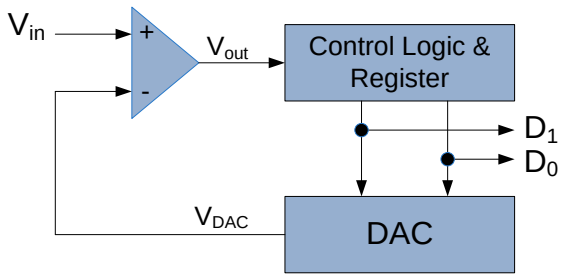

Fig. 4. 2-bit ADC circuit.

Assume that V_{max} = 5V and V_{min} = 0V. DAC output voltage:

Input D_1D_0	Output V_{DAC} (V)
11_2	
10_2	
01_2	
00_2	

Initially, all the bits are cleared. The control logic sets one bit every clock cycle (MSB to LSB). If the output of the opamp is 5V (logic 1), the bit is left unchanged as 1, otherwise the bit is cleared. For example, let V_{in} = 3.15 V, find the digital value for V_{in}.

i. First, clear D_1D_0. So, $D_1D_0 = 00_2$
ii. Set bit D_1. So, $D_1D_0 = 10_2$. Therefore, V_{DAC} =
iii. Next, set bit D_0. So, $D_1D_0 = 11_2$. Therefore, V_{DAC} =
iv. Done. The final output of the ADC is

The same idea can be repeated for an N-bit ($D_{N-1}D_{N-2}\ldots D_1D_0$) ADC circuit. This is an example of a successive approximation ADC.

4. LP1768 ADC

The built-in 12-bit ADC of our microcontroller uses successive approximation method. The input signal can be selected from 8 different channels (pins). Similar to other peripherals, controlling the ADC operation will be done via the memory-mapped registers. Two major tasks to be completed:

i. Configure the peripheral: This task requires setting all the associated options to enable the ADC system. Sub-tasks in this step includes:
   a. Enable the power to the ADC (default disabled).
   b. Select the appropriate clock frequency
   c. Configure the associated pin
   d. Select the operation mode and configure other conversion options
ii. Enable interrupt: This task is not required for all application. Sub-tasks in this step includes:
   a. Enable the ADC to generate an interrupt request when a conversion is completed.

b. Enable the ADC interrupt in the NVIC
c. Write the appropriate interrupt handler function for ADC. Make sure to clear the appropriate interrupt flag before returning.

Overview of ADC Registers:

Table 530. ADC registers

Generic Name	Description	Access	Reset value[1]	AD0 Name & Address
ADCR	A/D Control Register. The ADCR register must be written to select the operating mode before A/D conversion can occur.	R/W	1	AD0CR - 0x4003 4000
ADGDR	A/D Global Data Register. This register contains the ADC's DONE bit and the result of the most recent A/D conversion.	R/W	NA	AD0GDR - 0x4003 4004
ADINTEN	A/D Interrupt Enable Register. This register contains enable bits that allow the DONE flag of each A/D channel to be included or excluded from contributing to the generation of an A/D interrupt.	R/W	0x100	AD0INTEN - 0x4003 400C
ADDR0	A/D Channel 0 Data Register. This register contains the result of the most recent conversion completed on channel 0.	RO	NA	AD0DR0 - 0x4003 4010
ADDR1	A/D Channel 1 Data Register. This register contains the result of the most recent conversion completed on channel 1.	RO	NA	AD0DR1 - 0x4003 4014
ADDR2	A/D Channel 2 Data Register. This register contains the result of the most recent conversion completed on channel 2.	RO	NA	AD0DR2 - 0x4003 4018
ADDR3	A/D Channel 3 Data Register. This register contains the result of the most recent conversion completed on channel 3.	RO	NA	AD0DR3 - 0x4003 401C
ADDR4	A/D Channel 4 Data Register. This register contains the result of the most recent conversion completed on channel 4.	RO	NA	AD0DR4 - 0x4003 4020
ADDR5	A/D Channel 5 Data Register. This register contains the result of the most recent conversion completed on channel 5.	RO	NA	AD0DR5 - 0x4003 4024
ADDR6	A/D Channel 6 Data Register. This register contains the result of the most recent conversion completed on channel 6.	RO	NA	AD0DR6 - 0x4003 4028
ADDR7	A/D Channel 7 Data Register. This register contains the result of the most recent conversion completed on channel 7.	RO	NA	AD0DR7 - 0x4003 402C
ADSTAT	A/D Status Register. This register contains DONE and OVERRUN flags for all of the A/D channels, as well as the A/D interrupt/DMA flag.	RO	0	AD0STAT - 0x4003 4030
ADTRM	ADC trim register.	R/W	0x0000 0F00	AD0TRM - 0x4003 4034

From *Table 530 in the LPC17xx User manual*, NXP Semiconductors, 2010.

Configuring the Peripheral:

Step 1: Enable the ADC power by setting the PCADC bit (bit 12) of the PCONP register (0x400FC0C4).

Step 2: Select the appropriate clock frequency for the ADC by configuring the PCLK_ADC bits (bits 25:24) of the PCLKSEL0 register (0x400FC1A8).

Important notes:

- **The maximum clock frequency allowed to operate the ADC is 13 MHz.**
- **It takes 65 clock cycles to complete one conversion.**

Question: to ensure the proper operation of the ADC, what value should be assigned to PCLK_ADC bits in the PCLKSEL0 register?

It is also possible to scale down the clock further by using the CLKDIV bits in the AD0CR register (0x0x40034000). See the description of the AD0CR register in step 4.

Step 3: Enable the microcontroller pins to function as the ADC input channels through the PINSEL registers. In addition the mode of input pin can also be configured via PINMODE registers (optional).

Step 4: Select the operation mode and how the ADC channel(s) should be scanned via the AD0CR register (0x40034000).

Table 531: A/D Control Register (AD0CR - address 0x4003 4000) bit description

Bit	Symbol	Value	Description	Reset value
7:0	SEL		Selects which of the AD0.7:0 pins is (are) to be sampled and converted. For AD0, bit 0 selects Pin AD0.0, and bit 7 selects pin AD0.7. In software-controlled mode, only one of these bits should be 1. In hardware scan mode, any value containing 1 to 8 ones is allowed. All zeroes is equivalent to 0x01.	0x01
15:8	CLKDIV		The APB clock (PCLK_ADC0) is divided by (this value plus one) to produce the clock for the A/D converter, which should be less than or equal to 13 MHz. Typically, software should program the smallest value in this field that yields a clock of 13 MHz or slightly less, but in certain cases (such as a high-impedance analog source) a slower clock may be desirable.	0
16	BURST	1	The AD converter does repeated conversions at up to 200 kHz, scanning (if necessary) through the pins selected by bits set to ones in the SEL field. The first conversion after the start corresponds to the least-significant 1 in the SEL field, then higher numbered 1-bits (pins) if applicable. Repeated conversions can be terminated by clearing this bit, but the conversion that's in progress when this bit is cleared will be completed. **Remark:** START bits must be 000 when BURST = 1 or conversions will not start.	0
		0	Conversions are software controlled and require 65 clocks.	
20:17	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
21	PDN	1	The A/D converter is operational.	0
		0	The A/D converter is in power-down mode.	
23:22	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
26:24	START		When the BURST bit is 0, these bits control whether and when an A/D conversion is started:	0
		000	No start (this value should be used when clearing PDN to 0).	
		001	Start conversion now.	
		010	Start conversion when the edge selected by bit 27 occurs on the P2.10 / EINT0 / NMI pin.	
		011	Start conversion when the edge selected by bit 27 occurs on the P1.27 / CLKOUT / USB_OVRCRn / CAP0.1 pin.	
		100	Start conversion when the edge selected by bit 27 occurs on MAT0.1. Note that this does not require that the MAT0.1 function appear on a device pin.	
		101	Start conversion when the edge selected by bit 27 occurs on MAT0.3. Note that it is not possible to cause the MAT0.3 function to appear on a device pin.	
		110	Start conversion when the edge selected by bit 27 occurs on MAT1.0. Note that this does not require that the MAT1.0 function appear on a device pin.	
		111	Start conversion when the edge selected by bit 27 occurs on MAT1.1. Note that this does not require that the MAT1.1 function appear on a device pin.	
27	EDGE		This bit is significant only when the START field contains 010-111. In these cases:	0
		1	Start conversion on a falling edge on the selected CAP/MAT signal.	
		0	Start conversion on a rising edge on the selected CAP/MAT signal.	
31:28	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

From *Table 531 in the LPC17xx User manual*, NXP Semiconductors, 2010.

There are 3 ways to set start an A/D conversion:

- Burst mode: continuous conversion of selected channel(s).
- Edge: conversion starts when an edge such as EINT0 is detected (as in lab 5).
- Software code: conversion starts when 001_2 are written to START bits (bits 26:24) of the AD0CR register (as in example).

Configuring ADC Interrupt:

Step 1: Select a condition that will generate an interrupt request from the ADC. This is done through the A/D Interrupt Enable register (AD0INTEN – 0x4003400C).

Table 533: A/D Status register (AD0INTEN - address 0x4003 400C) bit description

Bit	Symbol	Value	Description	Reset value
0	ADINTEN0	0	Completion of a conversion on ADC channel 0 will not generate an interrupt.	0
		1	Completion of a conversion on ADC channel 0 will generate an interrupt.	
1	ADINTEN1	0	Completion of a conversion on ADC channel 1 will not generate an interrupt.	0
		1	Completion of a conversion on ADC channel 1 will generate an interrupt.	
2	ADINTEN2	0	Completion of a conversion on ADC channel 2 will not generate an interrupt.	0
		1	Completion of a conversion on ADC channel 2 will generate an interrupt.	
3	ADINTEN3	0	Completion of a conversion on ADC channel 3 will not generate an interrupt.	0
		1	Completion of a conversion on ADC channel 3 will generate an interrupt.	
4	ADINTEN4	0	Completion of a conversion on ADC channel 4 will not generate an interrupt.	0
		1	Completion of a conversion on ADC channel 4 will generate an interrupt.	
5	ADINTEN5	0	Completion of a conversion on ADC channel 5 will not generate an interrupt.	0
		1	Completion of a conversion on ADC channel 5 will generate an interrupt.	
6	ADINTEN6	0	Completion of a conversion on ADC channel 6 will not generate an interrupt.	0
		1	Completion of a conversion on ADC channel 6 will generate an interrupt.	
7	ADINTEN7	0	Completion of a conversion on ADC channel 7 will not generate an interrupt.	0
		1	Completion of a conversion on ADC channel 7 will generate an interrupt.	
8	ADGINTEN	0	Only the individual ADC channels enabled by ADINTEN7:0 will generate interrupts.	1
		1	Only the global DONE flag in ADDR is enabled to generate an interrupt.	
31:17	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

From *Table 533 in the LPC17xx User manual*, NXP Semiconductors, 2010.

Step 2: Enable the ADC interrupt in the NVIC by setting bit 22 of the ISER0 register (0xE000E100).

Step 3: Write the interrupt handler function for ADC. Perform the appropriate processing and make sure to clear the associated flag before returning. The flag is cleared when the data register *n* (AD0DR*n*) of the ADC channel *n* is read.

5. Example

Write a C code to continuously sample the AD0.5 channel and output the most significant 8 bits of the ADC result to 8 LEDs on the MBC1700 board. You should enable the ADC to produce a result as fast as possible. Use software code to start the conversion process.

Initialization Tasks:

Processing Tasks:

6. References

[1]. Joseph Yiu, *The Definitive Guide to ARM Cortex-M3 and Cortex-M4 Processors*, Elsevier, 3rd ed, 2014.
[2]. Jonathan Valvano, *Introduction to ARM Cortex-M Microcontroller*, 4nd ed, 2013.
[3]. *ARMv7-M Architecture Reference Manual*, ARM Limited, 2010.
[4]. *LPC17xx User manual*, NXP Semiconductors, 2010.
[5]. *Cortex-M3 Technical Reference Manual*, ARM Limited, 2010.

C code:

```c
/*-----------------------------------------------------------------------------
 Lesson 16: Configure ADC to scan AD0.5 channel and output LEDs
 *-----------------------------------------------------------------------------*/
#include "LPC17xx.h"         // Device header
#include "LED.h"
void ADC_Init(void);
unsigned int* PCONP_ptr       = (unsigned int*) 0x400FC0C4;
unsigned int* PCLKSEL0_ptr    = (unsigned int*) 0x400FC1A8;
unsigned int* PINSEL3_ptr     = (unsigned int*) 0x4002C00C;
unsigned int* AD0DR5_ptr      = (unsigned int*) 0x40034024;
unsigned int* AD0CR_ptr       = (unsigned int*) 0x40034000;
unsigned int ADC_value = 0x0;   // use this value to pass to LED_Out function
unsigned int AD0DR5_value = 0x0;// temp storage for AD0DR5 register

//main
int main (void) {
      LED_Init();
      ADC_Init();

      // main loop
      while (1){
            //Start conversion process and wait for DONE = 1




            //Extract result to ADC_value


            //Output ADC_value to 8 LEDs
            LED_Out(ADC_value);
      }
}

//ADC_Init: set up ADC to scan channel AD0.5
void ADC_Init(void)
{
      //Enable ADC power first


      //Set PCLK for ADC


      //Enable pin as AD0.5 channel


      //Select channel AD0.5
      //A/D is operational

}
```