

# Experiment 5

## ***Analog Input and Output***

Hazem Selmi, Ahmad Khayyat

Version 162, 19 March 2017

# Table of Contents

1. Objectives .....	1
2. Parts List .....	1
3. Background .....	1
3.1. Using LPC1769 Peripherals .....	1
3.2. ADC Configuration .....	2
3.3. Reading Digital Values .....	4
3.4. Analog Output .....	5
4. Tasks .....	5
5. Resources .....	5
6. Grading Sheet .....	5

# 1. Objectives

- Using the *Analog-to-Digital Converter (ADC)* to read analog input
- Using the *Digital-to-Analog Converter (DAC)* to write analog output

## 2. Parts List

- LPC1769 LPCXpresso board
- USB A-Type to Mini-B cable
- Breadboard
- Light sensor and/or potentiometer
- Seven-segment display or set of LEDs
- 330-Ohm Resistors
- Jumper wires

## 3. Background

Many microcontrollers have pins that can be used for *analog input*. Because the microcontroller processes digital data only, analog input must be converted to digital data. An analog-to-digital converter (ADC) is an I/O circuit often integrated into microcontrollers to allow directly connecting external analog devices, such as sensors. The ADC would convert the sensor voltage into a digital value by transforming it into a binary code with a specific number of bits.



Although not critical to conducting this experiment, it would be useful to review the three steps involved in analog-to-digital conversion: sampling, quantization, and bit encoding (COE 241).

### 3.1. Using LPC1769 Peripherals

The LPC1769 includes an integrated ADC peripheral device. In general, using any peripheral device involves three main issues:

1. Powering up the peripheral
2. Configuring the peripheral clock
3. Configuring pin functions

#### 3.1.1. Power Up

All microcontroller peripherals must be powered up before they can be used. This was not a concern in earlier experiments because we were using peripherals that are powered up by default.

Powering peripherals up and down is controlled through the *Power Control for Peripherals Register (PCONP)*.

By referring to table 46 in Chapter 4 of the [LPC1769 manual](#), you can see that the reset value (default value) is 1 for some peripherals, meaning that they are powered on by default, whereas it is 0 (OFF by default) for others.

For example, in the timer experiment, if you use a timer other than timer 0 or timer 1, your experiment wouldn't work without powering up the timer in your program.



The A/D converter (ADC) power is controlled by bit 12 of the `PCONP` register, which is 0 by default. *You must set that bit to power up your ADC.*



To save power, you can turn the power OFF for any unused peripherals that are ON by default.

### 3.1.2. Peripheral Clock

Most of the microcontroller peripherals, including timers and the ADC, require setting a peripheral clock (`PCLK`) to drive the peripheral.

You have seen in Experiment 7 (Hardware Timers) that you can configure a device's `PCLK` using the `PCLKSEL0` and `PCLKSEL1` registers.

#### Exercise

Refer to Chapter 4 in the [LPC1769 manual](#) to find out the two bits needed to configure the `PCLK` frequency for the ADC.

#### Exercise

What would happen if you skip this step?

### 3.1.3. Pin Functions

Many microcontroller pins can be configured to perform one of many functions. From Experiment 3 (Interrupts), recall that the `PINSELx` registers are used to configure a pin's function. To use the ADC, you must set the function of an appropriate pin to be analog input (`AD0.x` in the manual).

#### Exercise

Refer to Chapter 8 of the [LPC176x manual](#) to determine:

1. which `PINSELx` register should be modified
2. which bits of the register should be modified
3. what value should the bits be set to

You should connect a device that generates an analog voltage signal to the selected pin. Examples of such devices are light sensors (LDR) and potentiometers.



It is professional to correctly address the above three issues for every peripheral you plan to use, regardless of the defaults.

## 3.2. ADC Configuration

The main setup register for the ADC is the *A/D Control Register* (`AD0CR`). The [AD0CR Register Fields figure](#) illustrates the fields of the `AD0CR` register.



There is only *one* ADC in the LPC1769 microcontroller. In the `LPC17xx.h` header file, the control register is referred to as `ADCR`; while in the chip manual it is called `AD0CR`. The reason for that is that some other chips have multiple ADCs, named: `AD0CR`, `AD1CR`, etc.

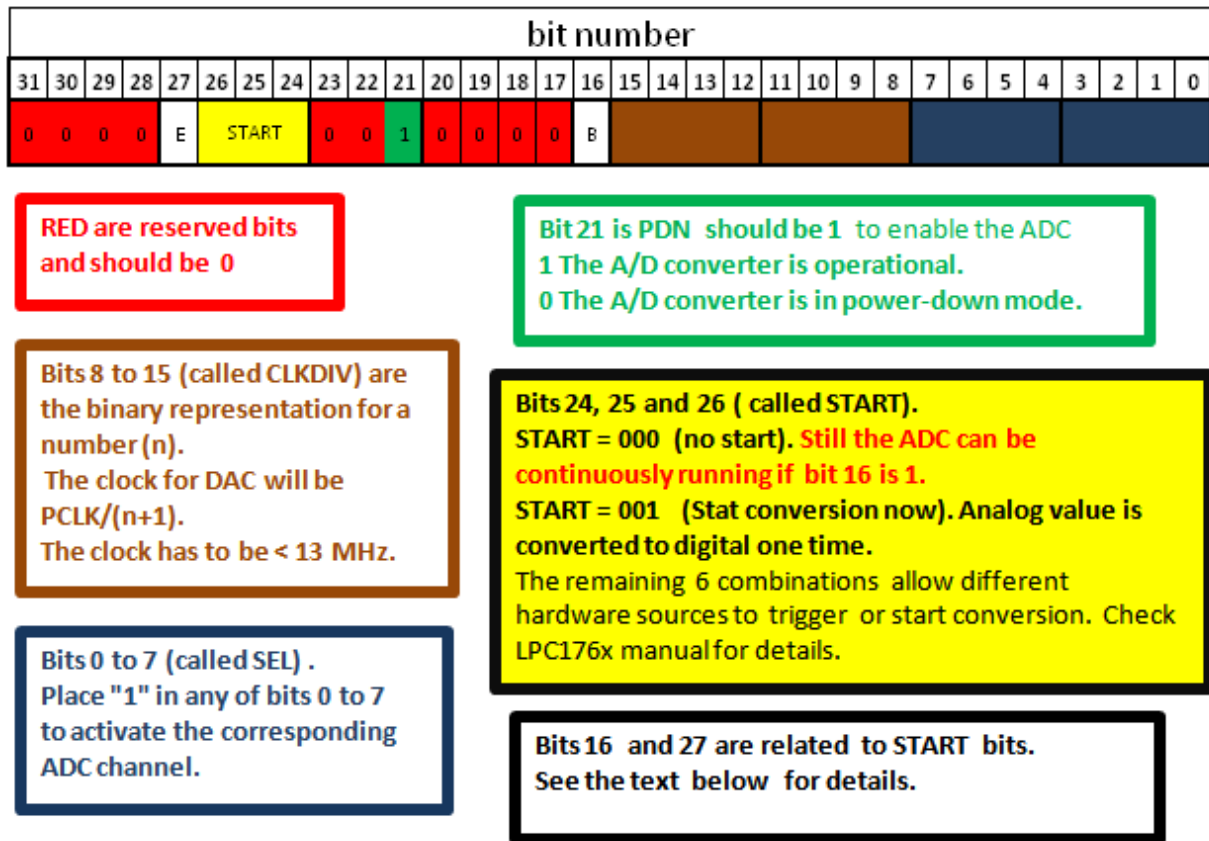


Figure 1. A/D Control Register (`AD0CR`) Fields

The following table explains the function of the `B` (*Burst*) and `E` (*Edge*) bits of the `AD0CR` register.



Bit 27 (`E`) works only if `B` = 0 and `START`  $\geq$  2. When  $2 \leq \text{START} \leq 7$ , the conversion starts when the state of a specific pin is changed. The `E` bit decides whether the ADC is triggered on the positive edge or the negative edge of that pin specified by `START`.

Bit	Label	Value	Effect
16	<code>B</code>	0	The <code>START</code> bits control when the ADC starts the conversion
		1	The ADC is continuously running ( <code>START</code> should be <code>000</code> )
27	<code>E</code>	0	Start conversion on a falling edge
		1	Start conversion on a rising edge

### 3.2.1. `START` vs. `BURST`

Using `START` will perform the conversion only once.

If you want the analog value to be repeatedly converted, you have two options:

1. Set the `B` bit (*Burst*) of the `AD0CR` register to 1; or
2. Set the `START` bits to `001` repeatedly, i.e. in a loop. The analog value is read every time such a statement is executed.

### 3.2.2. Using ADC Interrupt

In simple ADC applications, you don't need interrupts. You can simply read the digitized value from the proper register whenever needed and take some action. However, in some applications, such as real time applications, you may need to interrupt the CPU to take an action *only when* the conversion is completed. To do that, you can use the `ADGINTEN` register.



See Table 534 in Chapter 29 of the [LPC1769 manual](#) for details.

## 3.3. Reading Digital Values

There are 8 ADC channels, each corresponding to an analog pin. The digitized value corresponding to an input analog voltage is stored in 12 bits in one of the *A/D Data Registers*: `ADDR0` to `ADDR7`, where each register corresponds to an analog pin.

The [ADDR Register Fields figure](#) illustrates the fields of the `ADDRx` registers.

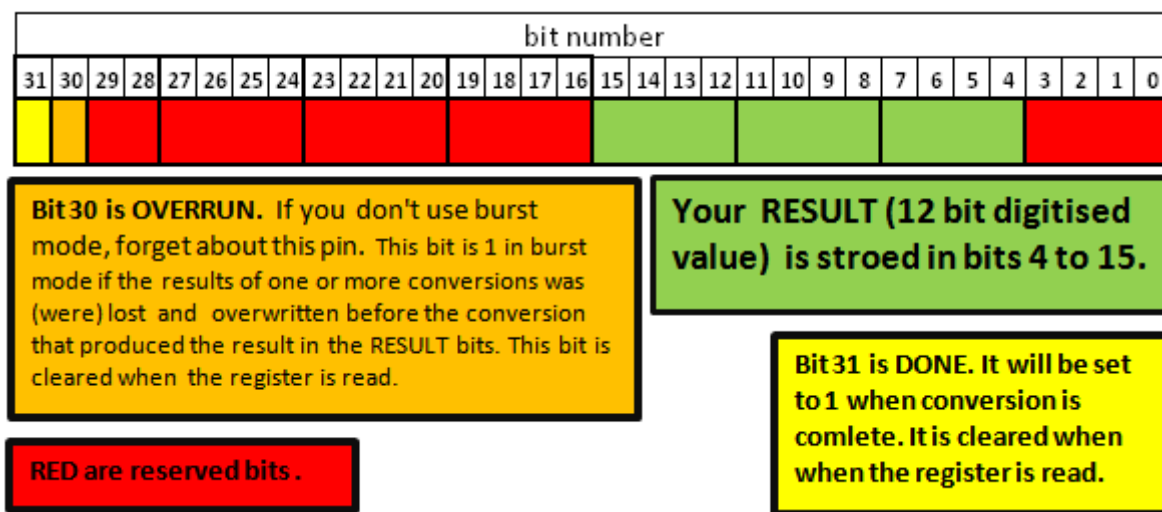


Figure 2. A/D Data Register (`ADDR`) Fields



Using proper shifting and bitwise operations, you should be able to get the proper value representing the analog voltage.



The `DONE` and `OVERRUN` bits are less important (may not be needed) in `BURST` mode. However in `START` mode, you may need to check them to avoid reading an old or unintended value.

#### Example 1. Using the `DONE` Bit

To wait until the conversion of the ADC channel 3 is over, you may use:

```
while ((LPC_ADC->ADDR3 & (1 << 31)) == 0); // Check the DONE bit for ADC channel #3
```

The 12-bit digital value generated by the ADC ranges from 0 to 4095. The way to process this value depends on your application.

You may want to divide this range to a number of sub-ranges, and assign different actions for each sub-range. In this case, you can use an if-else block.

In many applications, however, you will want to map this range to a another range using a mathematical

formula. For example, if you are reading from an analog temperature sensor, you would want to map the 0-to-4095 range to the range of temperatures supported by the sensor, as specified in the sensor's data sheet. In most cases, a linear relationship is sufficient.

## 3.4. Analog Output

To write analog values to an analog output device, use the LPC1769's digital-to-analog converter (DAC) as follows:

1. Use `PINSELx` to configure P0.26 to function as analog output (AOUT).
2. Use the *D/A Converter Register (DACR)* to set the digital value to be converted to analog.

Refer to Chapter 30 in the [LPC1769 manual](#) for details.

## 4. Tasks

1. Use the ADC in LPC1769 to read an analog input device, such as the LDR (light sensor) or the potentiometer.

The output can be any thing you want. The seven-segment display is a good option. You can simply display the analog level. If you use one seven-segment display, you have 10 different levels (0 to 9).

It is recommended to use a formula to map the readings to sensible values, instead of using an if-else block.

2. Use the DAC in LPC1769 to output analog values to an analog device.

## 5. Resources

*[lpc1769-manual]*

NXP Semiconductors. 'UM10360 LPC176x/5x User manual'. Rev. 3.1. 2 April 2014.  
[http://www.nxp.com/documents/user\\_manual/UM10360.pdf](http://www.nxp.com/documents/user_manual/UM10360.pdf)

## 6. Grading Sheet

Task	Points
Analog Input	5
Analog Output	2
Discussion	3