

Freescal Freedom Board with 32-bit ARM Cortex-M0+ Processor

Lab 8 Analog to Digital Conversion IR Proximity Sensor

Lab Goal

This lab covers use of the Analogue to Digital Converter (ADC) in ARM Cortex-M0+ processor and introduce the IR proximity sensor.

Principles of ADC

An ADC is a way for the MCU to read (input) a value that varies continuously over some range. This value, for example an angular position, or a temperature – must be converted to a voltage, which is read by the ADC. The ADC works by comparing the voltage with a reference voltage. The MCU uses a reference voltage of 3v so the converted voltage is in the range 0 to 3v. To do the conversion, the ADC must draw some current and take some time. The electrical characteristics of the ADC, including its input impedance, are given in the MCU datasheet. We must take care that the current drawn by the ADC does not significantly change the voltage we are trying to measure. Since the current used by the ADC is a feature of its design, this means that the resistance between the supply and the measured voltage should not be too large. If this is not the case, then the ADC loses accuracy. The time depends on the accuracy in bits and the frequency at which the ADC operates.

Using an ADC

In this part of the lab you can use an ADC to measure the voltage between two resistors.

Step 1: Circuit Principles

The ADC can be tested with a simple voltage divider in Fig. 1. The measured voltage should then be $R2 / (R1 + R2) \times 3$ volts.

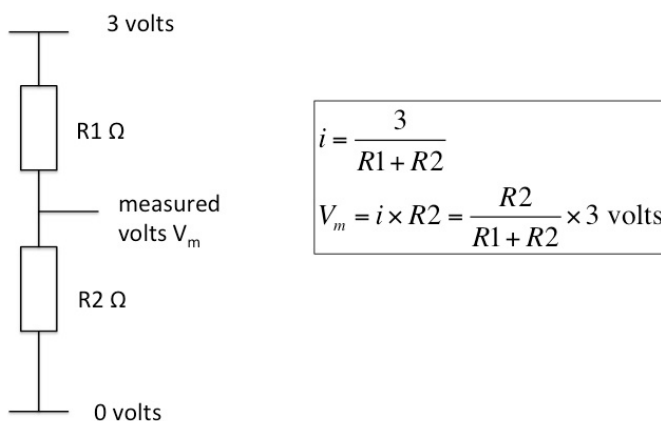
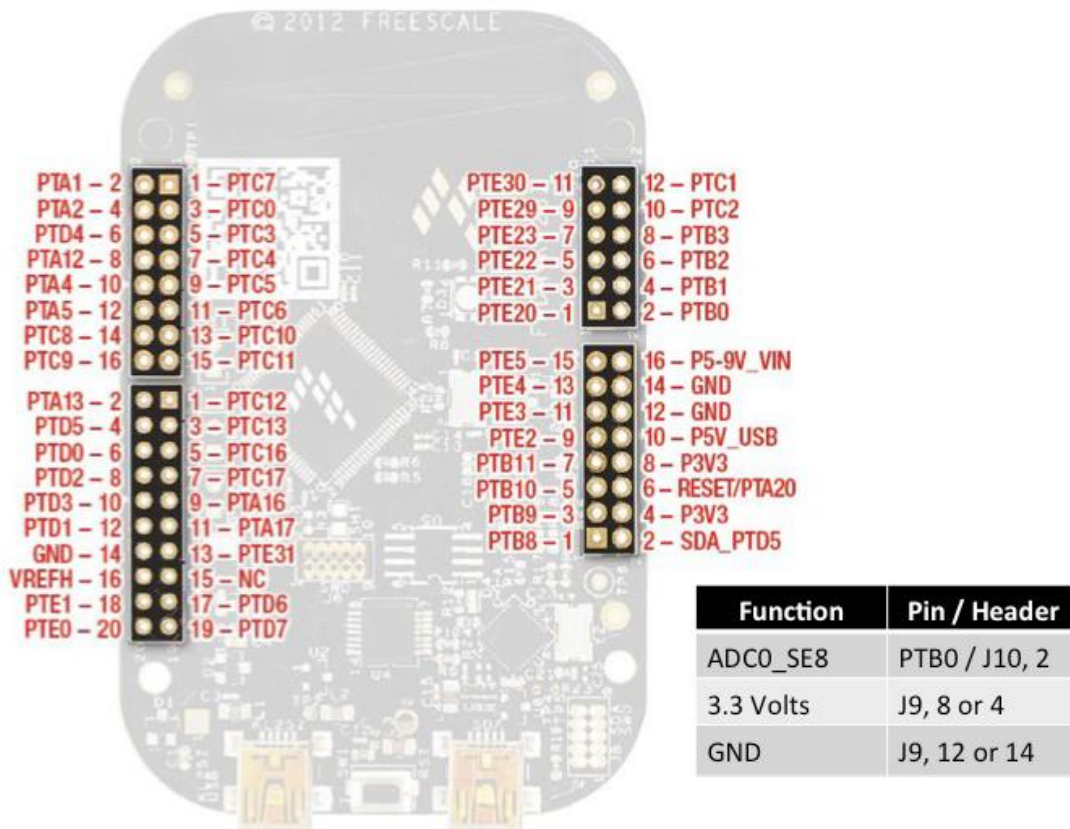


Fig. 1 Voltage divider

Step 2: Connections

The ADC measurement is on PTB0, which on J10 pin 2. Connect PTB0 pin to the V_m node in the circuit (Fig. 1) to the measured its voltage level. The following picture (from the user manual) shows the pin location of the 3.3v power supply; there are several GND (ground) connections on J9 header pin 12 and 14.



Each group of 6 holes in the breadboard is connected and the resistors are connected across the gap. The suggested values for the resistors are $1K\Omega$ and other values can be used as well.

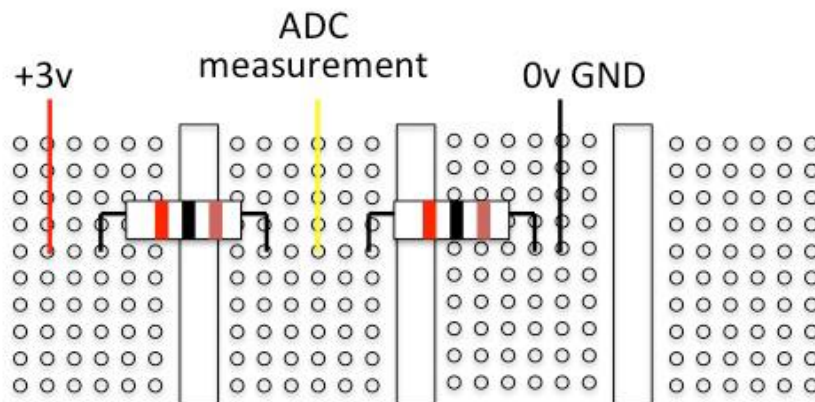


Fig. 2 wiring diagram for the voltage divider circuit

Step 3: Software and Use

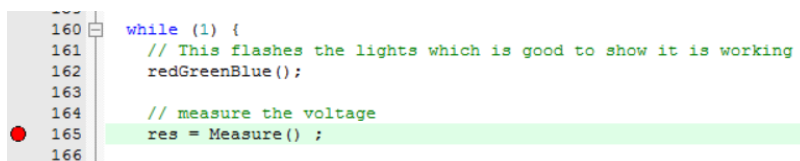
Download the sample project.

Read the initialization code. Add code to convert the value from the ADC to the correct voltage. Use the debugger to observe the value for several pairs of resistors. You may replace the simple voltage divider with a potentiometer. You may also wish to use a voltmeter to check the accuracy of the measurements.

Step 4: Use Debug tool in Keil IDE

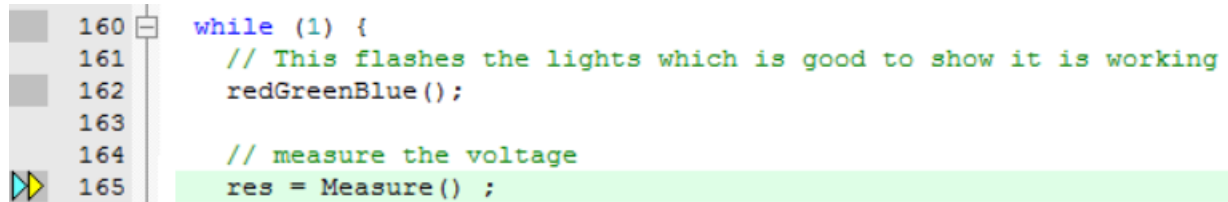
Enter debug mode. Try the following actions.

1. Insert a break point as shown in the program. Go to Keil, select Debug → select a break point. Find the line of the code you want to insert a break point.



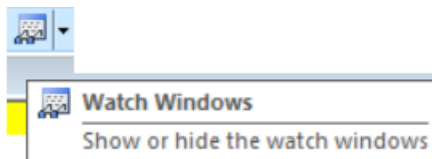
```
160 while (1) {  
161     // This flashes the lights which is good to show it is working  
162     redGreenBlue();  
163  
164     // measure the voltage  
165     res = Measure() ;  
166 }
```

2. Select Debug → start debug session → press F10 key to step over/execute each line of code. When the code step over the function redGreenBlue, it will turn on red, blue, and then green LED color.



```
160 while (1) {  
161     // This flashes the lights which is good to show it is working  
162     redGreenBlue();  
163  
164     // measure the voltage  
165     res = Measure() ;
```

3. Continue to press F10 key.
4. Select the Watch Window on top of the Keil IDE menu bar.



5. Enter the variable in the Watch Window, the Name column, with the variable that is used in your C program. The measured voltage from the bread board circuit will display in the value column. The example shows the voltage reading at 3.2983V.

Watch 1		
Name	Value	Type
res	0x000FF...	unsigned...
measured_voltage	3.29838872	float
<Enter expression>		

End of ARM Lab 2#Part 1

Infrared (IR) Proximity Sensor

Procedure:

In this part of the lab you will build a simple proximity sensor. The goal is to use an Infrared (IR) sensor to detect an object by shining an infrared light and measuring impact on ambient light level. The IR sensor consists of an IR emitter (LED) and an IR detector (phototransistor) pointing in the same direction to determine if any object is present in front of the IR emitter and it reflects the IR energy back to the phototransistor (receiver).

Build the circuit in Fig.3

1. Use two small breadboards and the components in Fig. 3 to build the IR sensor circuit. One breadboard should have an IR LED emitter circuit and the other have a photodetector circuit.
2. Move the IR LED emitter toward the photodetector. The LED will be lit, and its brightness is proportional to the distance of the phototransistor to the IR LED emitter. This circuit shows a basic function of a proximity sensor and how it can be used to detect objects. A 1.2K resistor can replace the 10K resistor as shown in Fig. 3.1 if you want the LED to be brighter when the photodetector received the IR energy.
3. The circuit can be used to verify the functionality of a proximity sensor before connecting it to the Freedom board. The IR LED emits purple light when it is ON. An example can be seen by clicking on the link: <http://www.robotroom.com/Infrared555.html>

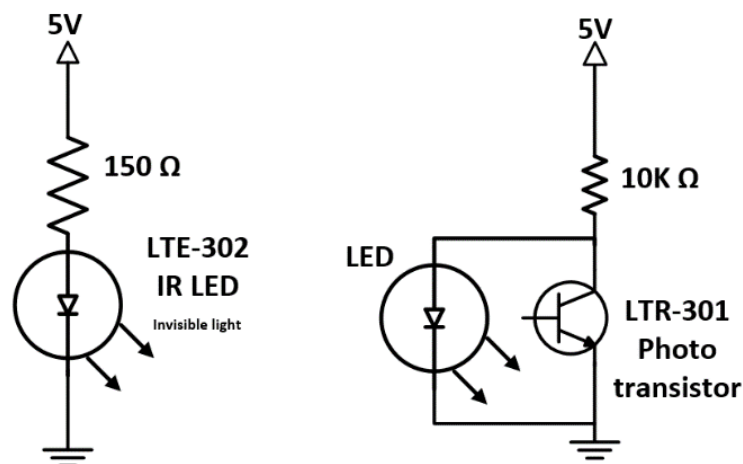


Fig. 3 wiring diagram of the Infrared proximity sensor

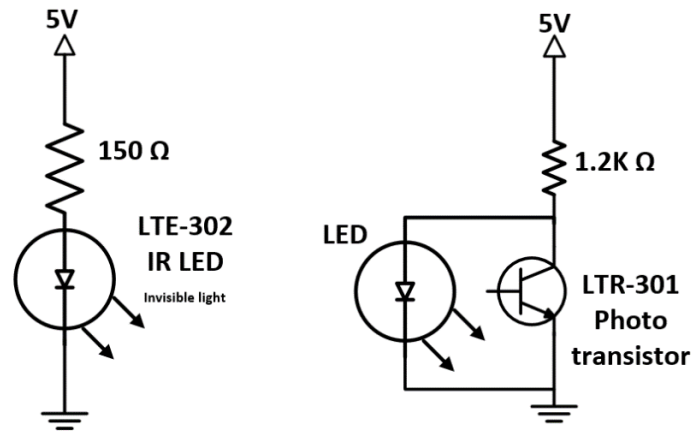


Fig. 3.1 wiring diagram of the Infarad proximity sensor with a 1.2K resistor in series with a phototransistor

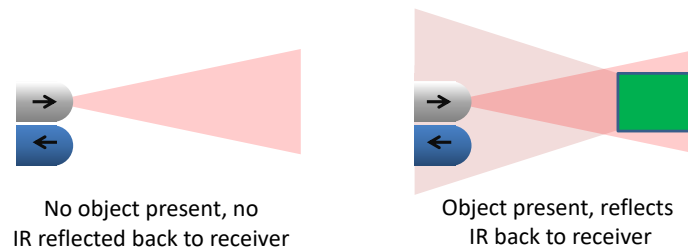


Fig. 4 The method of operation

The proximity IR sensor works with a combination of hardware and software. Sensing occurs in two steps: First, the software must measure the IR light level (using an IR-sensitive phototransistor Q1 and the analog to digital converter) when the IR-emitting LED is **turned off**. Second, the software must measure the IR light level when the IR LED is turned **on**. If the IR brightness level has increased, then there may be an object nearby reflecting the IR from D1 back to Q1 (Fig. 5). The IR signal strength at the IR emitter is to be verified with the LED in Fig. 3. You can also use a phone's camera to view the IR energy when the IR LED turn on.

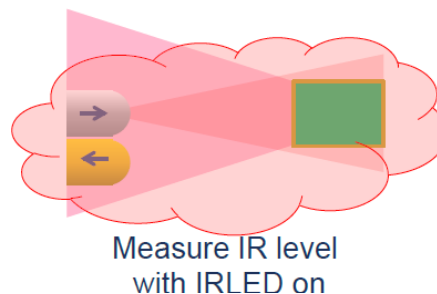
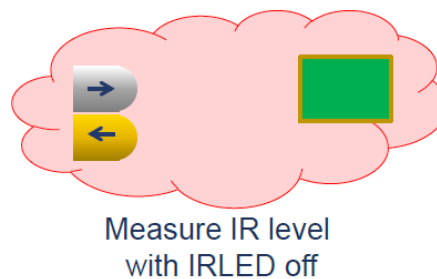
Using Differential Measurements:

- **Basic approach of measuring ambient + reflected light is unreliable**

- Vulnerable to changes in ambient light due to flicker in light sources, shadows, etc.

- **Use differential measurements instead**

- Measure brightness with IRLED off
 - Measure brightness with IRLED on
 - Difference in brightness levels indicates amount of IR reflected



Circuit and Operation

Build the circuit on the breadboard as shown below.

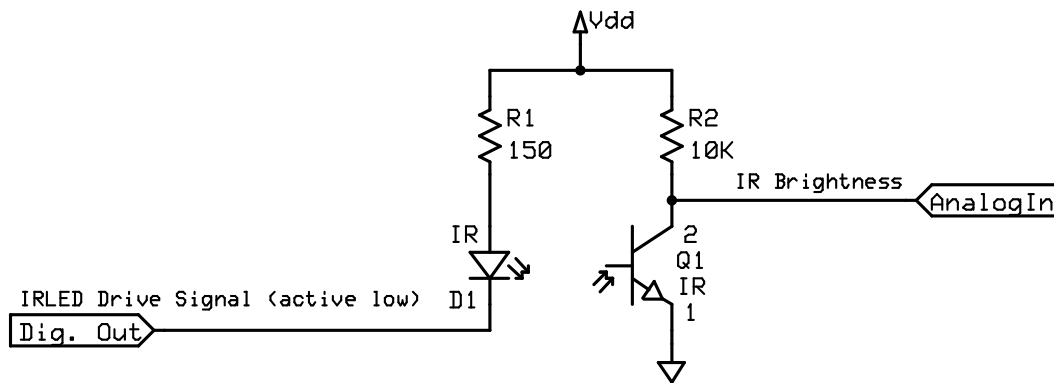
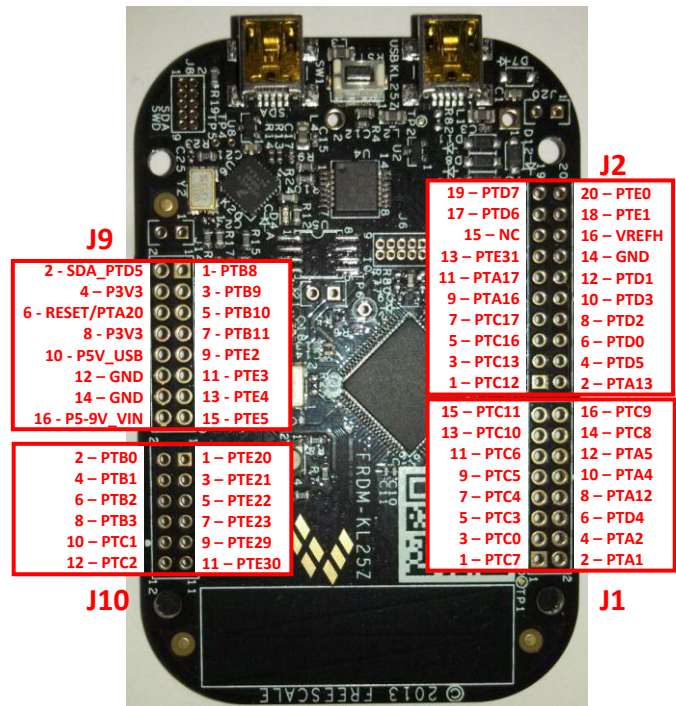
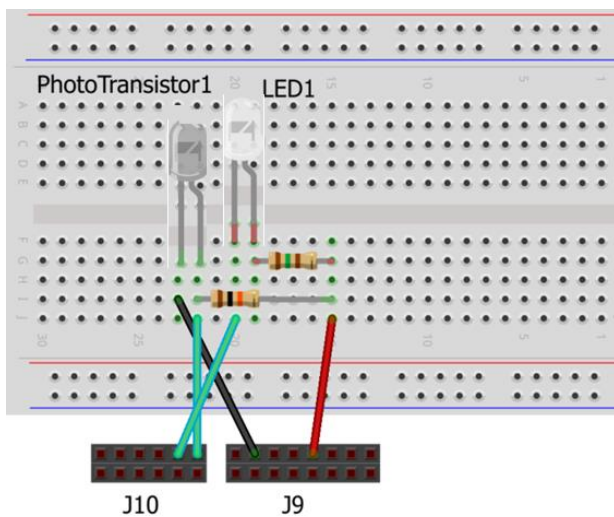


Fig. 5 wiring diagram

1. Infrared LED emitter –emits IR light.
2. Infrared phototransistor –conducts more current as IR increases.
3. The IR LED emitter has a clear plastic package and the phototransistor has a dark package.



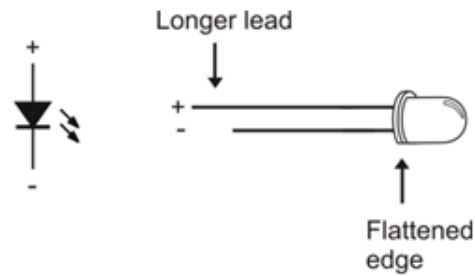
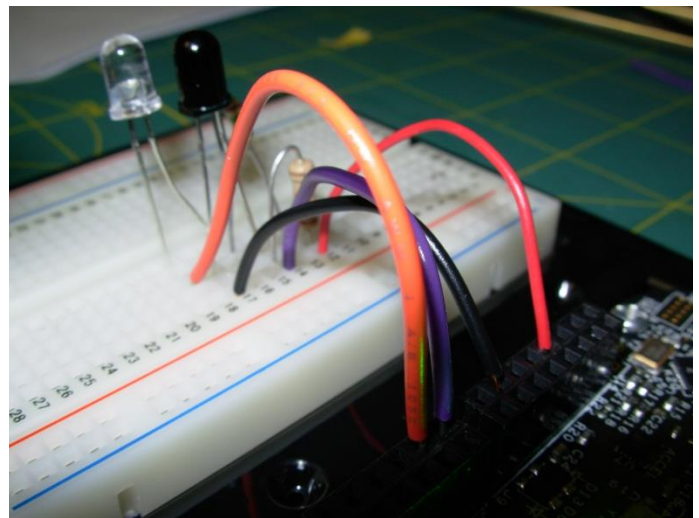
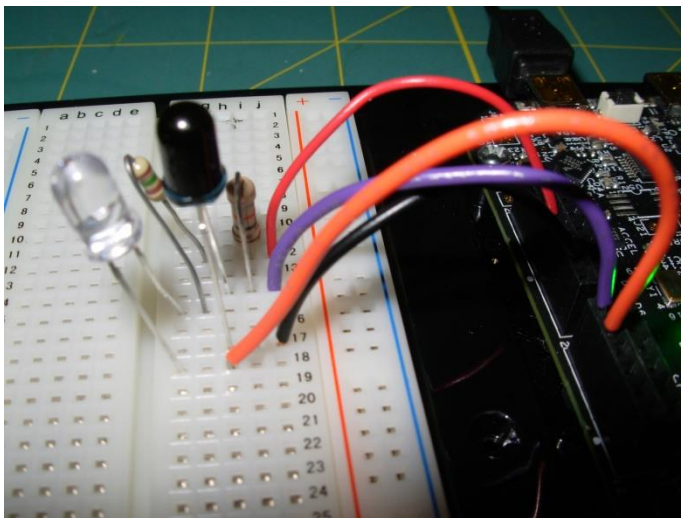


Fig. 6 wiring diagram

Signals and connections

Signal Name	Description	Direction	MCU				Freedom KL25Z Board	
			Port	Bit	Connector	Pin		
Dig. Out	Drive signal for IR LED D1	Output	B	1	J10	4		
Analog In	IR brightness	Input	B	0	J10	2		
Vdd	3.3 V power supply	Power	-	-	J9	4 or 8		
GND	Ground	Power	-	-	J9	12 or 14		



As IR energy increases, the conductivity of the phototransistor increases and lowers the output voltage. However, the phototransistor has a slow response. (You may try measuring this with an oscilloscope).

4. Replace R2 with a 1.2 resistor (Fig. 5) and verify if the IR brightness level has increased. Describe the differences when you use 10K and 1.2K in series with the phototransistor.

Have the instructor check off your work when you are done.

Reference# 1:

Software Requirements and Design

The requirements for the range measurement software are:

Take measurements by (i) turning the IR LED on, and waiting for a short while before measuring the voltage on the phototransistor, then (ii) turning the IR LED off, again waiting then measuring. The difference between the two voltage measurements indicates the presence of a reflecting object and the greater the difference the closer the object.

Delays need to be of the order of 1ms or more. It is best to repeat the measurement multiple (e.g. 10) times and take the average. The resulting range can be monitored using the debugger. Alternatively, we can use the week 1 lab code to set the color of the multi-color LED on the Freedom board. A project is provided with an outline of the code. The following software functions are suggested:

- Initialization functions: configures GPIO pins and ADC input.
- 'Control_IR_LED' function: turns on or off IR LED.
- 'Measure_IR' function: ruse the ADC to read the voltage level.
- **[Included]** 'Delay' function: performs delay loop based on function argument.
- **[Included]** 'Display_Range' function: display RGB LED based on range.

The main function, which is not included, initializes the system and then repeatedly measures difference in brightness caused by lighting LED, averages this difference over several measurements and displays the range using the RGB LED.

Ambient Light & Shielding:

With the IR emitter connected, monitor the IR difference reading. Shield the phototransistor from IR energy emitted from the side of the LED. Does this change the difference signal strength?

Delay

The sensitivity of the proximity sensor increases as you wait longer to sample the phototransistor's voltage after changing the IR LED. Try different values (up to a few milliseconds. Note: delay function gives only an approximate delay. Use an oscilloscope if you want the exact value.

Range

What is the maximum distance at which your sensor can reliably detect a sheet of paper? Does it vary for other objects (e.g. your hand)? Choose an object to work with.

Calibration

Calibrate your code so that the RGB LED is lit according to object distance and color, shown below. Calibrate by adjusting the values in the array called Threshold, declared in main.c. To ensure consistency, use the same object for all calibration and testing.

Color	Distance %Max Range	Example Distance
	Out of range	No object within 20 cm
Blue	80-100% max range	16-20 cm
Yellow	60-80%	12-16 cm
Red	40-60%	8-12 cm

White	< 20% max range	0-4 cm
-------	-----------------	--------

Reference# 2:

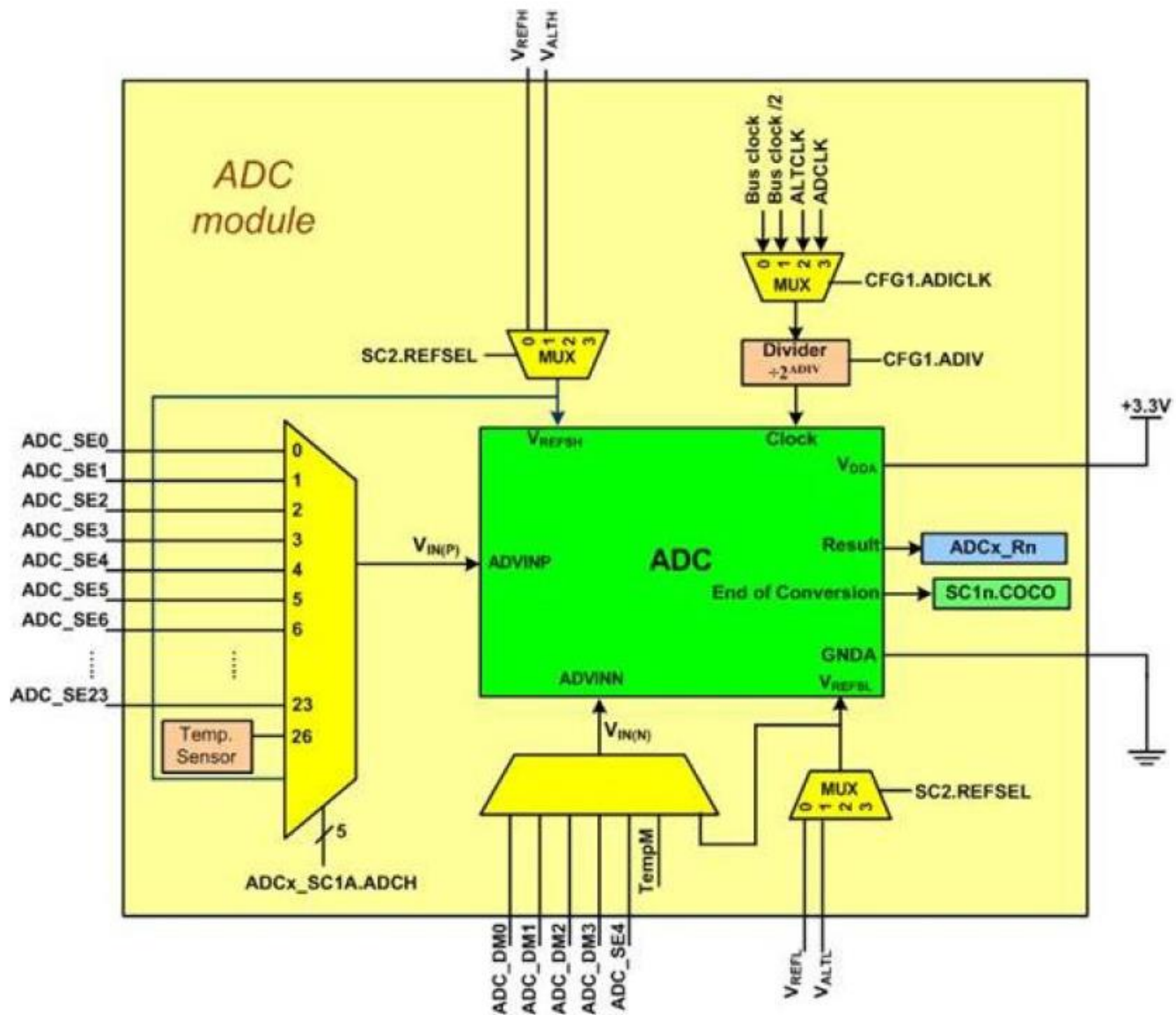
The Freescale KL25Z ARM chip has a single ADC module which can support up to 31 ADC channels. These ADC channels have 16-bit resolution.

ADC Conversion:

The conversion for the ADC has three parts. They are as follows:

1. In the first phase, a sample amplifier of unity gain samples the analog input for a total of n clock cycles. This buffering of the analog input charges the sample capacitor up to the input potential.
2. In the second phase, the sample buffer is disconnected and connected to the storage node for a certain number of clock cycles. The number of clock cycles can be 4, 6, 10, 16, or 24. We program this number via the ADLSMP bit in ADCx_CFG1 register and the ADLSTS bits in ADCx_CFG2 register. Longer sampling time ensures that the voltage of the sample capacitor is brought closer to the input voltage. This is important when the input voltage differs significantly from sample to sample. But it prolongs the conversion time of each sample.
3. In the third phase, the analog input is converted to binary numbers using the successive approximation method. In this phase, the number of clocks used depends on how many bits are in the binary output. For each bit, we need one clock. That means we need 8 clocks for the 8-bit output, 10 clocks for the 10-bit output, and so on. We choose the n -bit resolution option using the MODE bits of ADCx_CFG1 register.

To program them, we need to understand some of the major registers. The picture shows a simplified block diagram of a KL25Z chip.

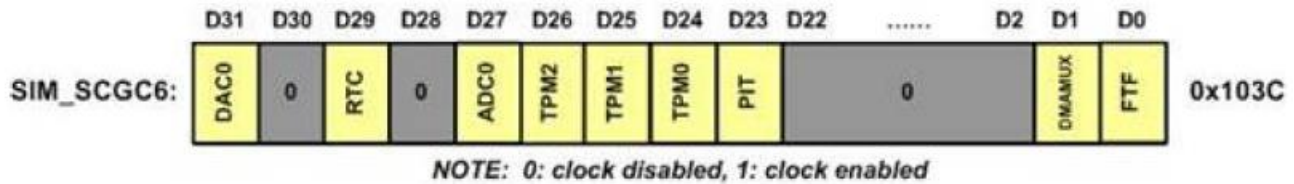


Below are some major registers of KL25Z ADC from KL25Z reference manual.

Absolute Address	Register
4003 B000	ADC Status and Control Registers 1 (ADC0_SC1A)
4003 B004	ADC Status and Control Registers 1 (ADC0_SC1B)
4003 B008	ADC Configuration Register 1 (ADC0_CFG1)
4003 B00C	ADC Configuration Register 2 (ADC0_CFG2)
4003 B010	ADC Data Result Register (ADC0_RA)
4003 B014	ADC Data Result Register (ADC0_RB)

Enabling Clock to ADC

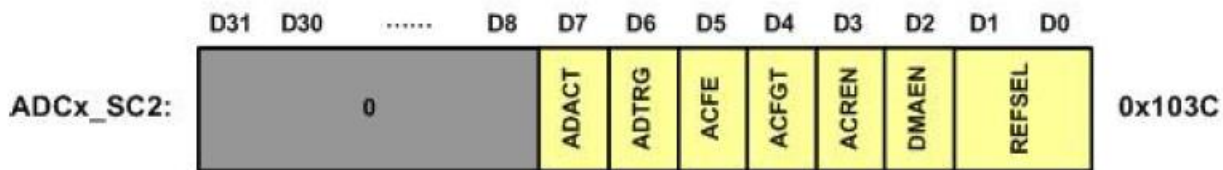
First thing we need to do is to enable the clock to the ADC0 module. Bit D27 of SIM_SCGC6 register is used to enable the clock to ADC0 module. The SIM_SCGC6 is part of the System Integration Module and located at physical address $0x4004\ 7000 + 103C = 0x4000\ 803C$.



SIM_SCGC6 register t for Enableing to ADC0

Start Conversion trigger options

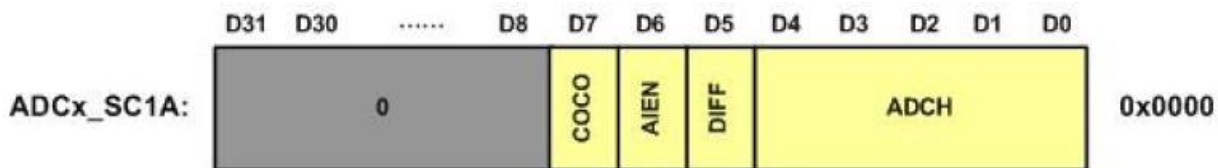
There are two start-conversion (trigger) options. They are hardware trigger and software trigger. The selection of hardware or software trigger for conversion is done via the bit D6 (ADTRG, ADC Trigger) of ADC0_SC2 (ADC0 Status Control 2) register. The hardware trigger may be external pin, comparator, or timers (TPMx, LPTMR0, PIT, or RTC). The selection of hardware trigger is done in SIM_SOPT7 register. The default trigger is software and that is what we will use in this section.



ADCx_SC2 Register

Choosing Vin input channel

The channel selection is done through the ADC0_SC1A (ADC Status and Control 1A) register. (There are more ADC0_SC1n registers but only ADC0_SC1A can be used for software trigger. The other registers are for hardware trigger only and will not be discussed here.) The lowest 5 bits of ADC0_SC1A register are used to select one of the 31 single-ended channels to be converted.



ADCx_SC1A Register

Bit	Field	Descriptions
7	COCO	Conversion Complete Flag: (0: Conversion is not completed, 1: Conversion is completed) The COCO is cleared when the ADCx_SC1n register is written or the ADCx_Rn register is read.
6	AIEN	Interrupt Enable: The ADC interrupt is enabled by setting the bit to HIGH. If the interrupt enable is set, an interrupt is triggered when the COCO flag is set.
5	DIFF	Differential mode (0: Single-ended mode, 1: Differential mode)
4-0	ADCH	ADC input channel: The field selects the input channel as shown in Figure 7-7. When DIFF = 0 (single-ended mode), values 0 to 23 choose between the 24 input channels (ADC_SE0 to ADC_SE23). When DIFF = 1 (Differential mode), values 0 to 3 select between the 4 differential channels. See the reference manual for more information. When ADCH = 11111, the module is disabled.

ADCx_SC1 Register

Not all the channels are connected to the input pins. The number of available channels in the Freescale KL25Z varies among the family members. In the case of KL25Z128VLK4 ARM chip used in FRDM board, there are 14 channels connected to the input pins and additional 4 channels are connected internally. See table below.

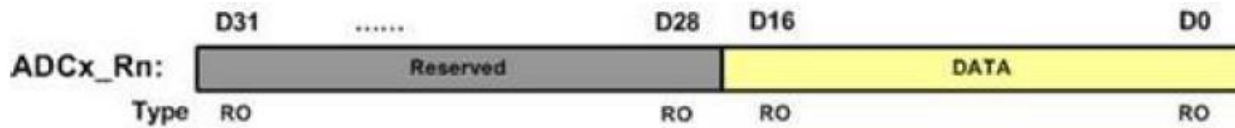
ADC Inputs and their pin names on the Freedom Board:

Pin Name	Description	Pin
ADC_SE0	ADC input 0	PTE20
ADC_SE3	ADC input 3	PTE22
ADC_SE4	ADC input 4	PTE21, PTE29
ADC_SE5	ADC input 5	PTD1
ADC_SE6	ADC input 6	PTD5
ADC_SE7	ADC input 7	PTD6, PTE23
ADC_SE8	ADC input 8	PTB0
ADC_SE9	ADC input 9	PTB1
ADC_SE11	ADC input 11	PTC2
ADC_SE12	ADC input 12	PTB2
ADC_SE13	ADC input 13	PTB3
ADC_SE14	ADC input 14	PTC0
ADC_SE15	ADC input 15	PTC1
ADC_SE23	ADC input 23, DAC0 output	PTE30
ADC_SE26	Temperature sensor	
ADC_SE27	Bandgap reference	
ADC_SE29	V_{REFH}	
ADC_SE30	V_{REFL}	
ADC_SE31	Module disabled	

Analog input pin assignment in Freescale KL25Z

ADC Data result

Upon the completion of conversion, the binary result is placed in the ADC0_RA register. (There are many ADC0_Rn registers corresponding to the ADC0_SC1n registers. Because we can only use ADC0_SC1A for software trigger, the data will be in ADC0_RA register. This is a 32-bit register but only the lower 16 bits are used. For the ADC, we have the options of 8-, 10-, 12-, and 16-bit for single-ended unsigned result. In any case, always the result is right-justified and the rest of the bits up to bit D15 are unused. If the result is in 2's complement for differential, then it is signed-extended to bit D15.



ADC Result Register (ADCx_Rn) See Table 28-43 in KL25Z Ref man

Clearing conversion complete flag

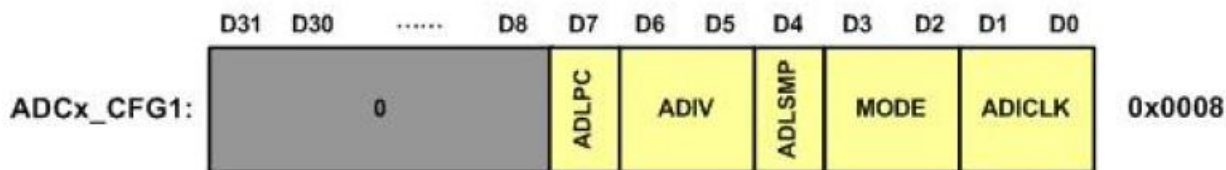
The conversion complete (COCO) flag bit in ADCx_SC1n register is cleared automatically when the data from the respective ADCx_Rn register is read.

Differential versus Single-Ended

In some applications, our interest is in the differences between two analog signal voltages (the differential voltages). Rather than converting two channels and calculate the differences between them, the KL25Z has the option of converting the differential voltages of two analog channels. The bit D5 (DIFF) of ADC0_SC1A register allows us to enable the differential option. Upon Reset, the default is the single-ended input and we will leave it at that for the discussion here. See the KL25Z reference manual for further information on differential options.

Selection Bit Resolution

We use ADCx_CFG1 (ADC configuration 1) register to select 8, 10, 12, or 16-bit ADC resolution. This register is also used to select the speed of the clock source to the ADC.



ADCx_CFG1 Register

Bit	Field	Descriptions		
7	ADLPC	Low-Power Configuration		
6-5	ADIV	Clock Divide Select: The clock is divided by 2^{ADIV} as shown in Figure 7-7.		
4	ADLSMP	Sample time configuration (0: Short sample time, 1: Long sample time)		
3-2	MODE	Conversion mode selection		
		MODE	In single-ended mode (DIFF = 0)	In differential mode (DIFF = 1)
		00	8-bit conversion	9-bit conversion with 2's complement output
		01	12-bit conversion	13-bit conversion with 2's complement output
		10	10-bit conversion	11-bit conversion with 2's complement output
		11	16-bit conversion	16-bit conversion with 2's complement output
1-0	ADICLK	Input Clock Select		
		ADICLK	Clock source	
		00	Bus clock	
		01	(Bus clock)/2	
		10	Alternate clock (ALTCLK)	
		11	Asynchronous clock (ADACK)	

ADCx_CFG1 Register

Notice in ADCx_CFG1, the MODE bits (D3:D2) selects the resolution. If we use the Low Power option with D7 bit, then the conversion speed is limited.

Vref in FRDM board

In the Freescale ARM KL25Z chip, the pin for Vref (+) is called VREFH (Vref High) and Vref (-) pin is called VREFL (Vref Low). In the FRDM board, the VREFH pin is connected to 3.3V, the same supply voltage as the digital part of the chip. The circuit may be altered to use the AREF pin for an external reference voltage. Even if we connect the VREFH to an external reference other than the VDD of the chip, it cannot go beyond the VDD voltage. With VREFH=3.3V, we have the step size of $3.3\text{V} / 65,536 = 0.05\text{ mV}$ since the maximum ADC resolution for KL25Z is 16 bits. See Example below.

Find the digital converted output if the analog input voltage is 1.2V for the Freescale FRDM board.

Solution:

Since the step size is $3.3\text{V} / 65,536 = 0.05\text{ mV}$, we have $1.2\text{V} / 0.05\text{ mV} = 23,831 = 0x5D17$ as ADC output.

Configuring ADC and reading ADC channel

In using ADC, we must also configure the GPIO pins to allow the connection of an analog signal through the input pin. In this regard, it is the same as all other peripherals. We need to take the following steps to configure the ADC:

1. Enable the clock to I/O pin used by the ADC channel. Table 7-7 in the textbook or page 13 of this document shows the I/O pins used by various ADC channels.
2. Set the PORTX_PCRn MUX bit for ADC input pin to 0 to use the pin for analog input channel. This is the power-on default.
3. Enable the clock to ADC0 modules using SIM_SCGC6 register.
4. Choose the software trigger using the ADC0_SC2 register.
5. Choose clock rate and resolution using ADC0_CFG1 register.
6. Select the ADC input channel using the ADC0_SC1A register. Make sure to choose the right pin and channel. Also, make sure interrupt is not enabled and single-ended option is used when you select the channel with this register.
7. Keep monitoring the end-of-conversion COCO flag in ADC0_SC1A register.
8. When the COCO flag goes HIGH, read the ADC result from the ADC0_RA and save it.
9. Repeat steps 6 through 8 for the next conversion.