# Function of the ADC (Analog to Digital Converter)

The ADC is a system that converts analog signals that come as inputs to the appropriate PINs, such as sound, current, light, etc., into a digital signal. Therefore, the values of the analog signals can now be read by the microcontroller and used appropriately. The ADC contained in the board accepts measurements from PIN7 of PORTD. Also, the basic operating modes of the ADC are two. The free-running mode in which the ADC continuously accepts values and converts them into digital signals. The second and default mode performs only one conversion whenever requested in the code and then stops.

In the example, when the ADC will read measurements that are below a value (for example 10) an interrupt will be triggered. At the interrupt an LED will light up for 5ms and then we will return to the original flow and wait for the ADC reading again.

To activate the ADC we need:

1. First select whether we want 10-bit or 8-bit resolution → Resolution Selection bit (RESSEL) in Control A register (ADCn.CTRLA).

2. Then, select Free-Running mode → Write '1' to the Free-Running bit (FREERUN) in ADCn.CTRLA.

3. Select which bit the ADC will be connected to → MUXPOS bit field in MUXPOS register

(ADCn.MUXPOS).

4. Enable the ADC → Write '1' to the ENABLE bit in ADCn.CTRLA.

5. We are given the option to enable Debug Mode so that the ADC never stops running and takes values.

With Window Comparator Mode all values entered by the ADC are checked against a given value, i.e. a threshold. To enable this mode the following steps must be followed:

1. First, the threshold is entered in the ADCn.WINLT and/or ADCn.WINHT register.

2. The interrupt generation function is enabled → Window Comparator Interrupt Enable bit (WCOMP) in Interrupt Control register (ADCn.INTCTRL).

3. Set the desired Mode (in our case we want interrupt when RESULT<THRESHOLD) → WINCM bit field in ADCn.CTRLE.

With Window Comparator Mode all values entered by the ADC are checked against a given value, i.e. a threshold. To enable this mode the following steps must be followed:

1. First, the threshold is entered in the ADCn.WINLT and/or ADCn.WINHT register.

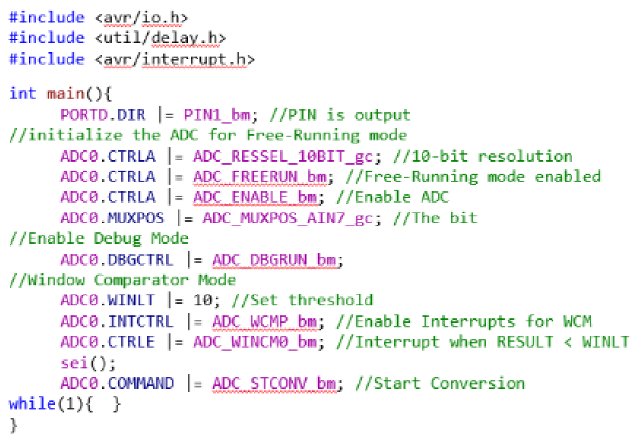
2. The interrupt generation function is enabled → Window Comparator Interrupt Enable bit (WCOMP) in Interrupt Control register (ADCn.INTCTRL).

3. Set the desired Mode (in our case we want interrupt when RESULT<THRESHOLD) → WINCM bit field in ADCn.CTRLE.

Once the ADC and its functions are properly programmed, it remains to write the logic '1' to the Start Conversion Bit (STCONV) in the Command Register (ADCn.COMMAND), which starts the conversion. The values are recorded in the RES (Result) register.

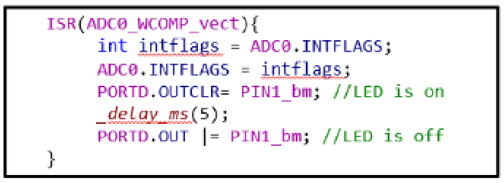
Tip: Refer to the ATmega4808 DataSheet and read pages 394-421 thoroughly. All the possible functions you can use to perform Analog to Digital Conversion and how to program them are detailed.

The code of the example implementation is shown below.



```
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

int main(){
      PORTD.DIR |= PIN1_bm; //PIN is output
//initialize the ADC for Free-Running mode
      ADC0.CTRLA |= ADC_RESSEL_10BIT_gc; //10-bit resolution
      ADC0.CTRLA |= ADC_FREERUN_bm; //Free-Running mode enabled
      ADC0.CTRLA |= ADC_ENABLE_bm; //Enable ADC
      ADC0.MUXPOS |= ADC_MUXPOS_AIN7_gc; //The bit
//Enable Debug Mode
      ADC0.DBGCTRL |= ADC_DBGRUN_bm;
//Window Comparator Mode
      ADC0.WINLT |= 10; //Set threshold
      ADC0.INTCTRL |= ADC_WCMP_bm; //Enable Interrupts for WCM
      ADC0.CTRLE |= ADC_WINCM0_bm; //Interrupt when RESULT < WINLT
      sei();
      ADC0.COMMAND |= ADC_STCONV_bm; //Start Conversion
while(1){  }
}
```



```
ISR(ADC0_WCOMP_vect){
      int intflags = ADC0.INTFLAGS;
      ADC0.INTFLAGS = intflags;
      PORTD.OUTCLR= PIN1_bm; //LED is on
      _delay_ms(5);
      PORTD.OUT |= PIN1_bm; //LED is off
}
```

Note: To simulate the operation of the ADC you must write the input value to the RES (Result) register manually.

## Laboratory Exercise 02:

## Smart Home Appliance, with Motion in Space

The purpose of this lab exercise is to simulate the operation of a smart home appliance that moves in the space of an empty room. It starts from a corner of the room and its purpose is to draw the outline of the room. As it moves through the space it will take values from a sensor that measures the distance of the device from an obstacle in front of it (here the sensor values will be measured by the ADC). If the value is below the allowable value (you specify any value between 1-254) the device should stop and move left. The device will choose to move right if the right sensor shows us that there is no wall to the right of the device (also simulated with the ADC), i.e. the value is above the allowed value. When it reaches the corner from which it started we want it to stop. We also want if a switch is pressed we want it to be able to return to its position following the path it has taken so far but in reverse.

Some assumptions to be used are the following:

- The movement will be simulated with an LED (when moving straight the LED is on otherwise it is off).

- There is only one ADC available on the board, so to use it in two different modes (check if it is approaching a wall and check if there is a wall on the right) you have to use free-running mode for one mode and for the other only one conversion is performed when requested (i.e. the second mode does not need to be checked continuously).

- The left and right movement will be simulated with two different LEDs which are on until a timer reaches a certain value (put in values that suit you and explain how you found the values).

- When a switch is pressed a reverse movement will be activated.

As for the reverse course, we want the device to turn 180 degrees. This will be simulated with the three LEDs on simultaneously for a period of time (use a timer). Then, it will perform the exact same process but in reverse. Specifically, as it moves forward and hits a wall we want it to turn right (corresponding LED on). When it realizes that there is no wall on the left we want it to move left (corresponding LED on). Finally, when it realizes that it has returned to its original position, it finishes.

## Laboratory Exercise Questions 2

1) Implement the home appliance motion code when the room is square (so the right sensor and the second ADC function are not needed).

2) Implement the motion code in the room shown in figure 2.2 below, here the second function of the ADC is also introduced.

3) Implement the reverse operation.

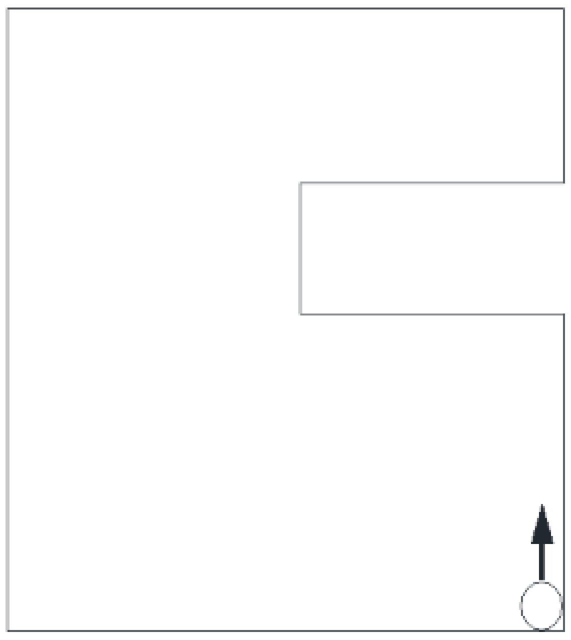


Figure: Shape of room