Laboratory 4: Analog-to-digital converter (ADC) and Pulse-width modulation (PWM)

Objectives

1. Understand the concept of Digital-to-analog converter (ADC)
2. Understand the concept of Pulse-width modulation (PWM)

Digital-to-analog converter (ADC)

An analog (aka. Analog signal) is continuous signal. Unlink a digital signal which observed values are either HIGH or LOW, analog signal give an approximate level of value (e.g. 2.3 volts). In real life, everything is analog i.e. temperature (degree Celsius), brightness (Lux), sound (decibel), pressure (pascal), and speed (km/s)

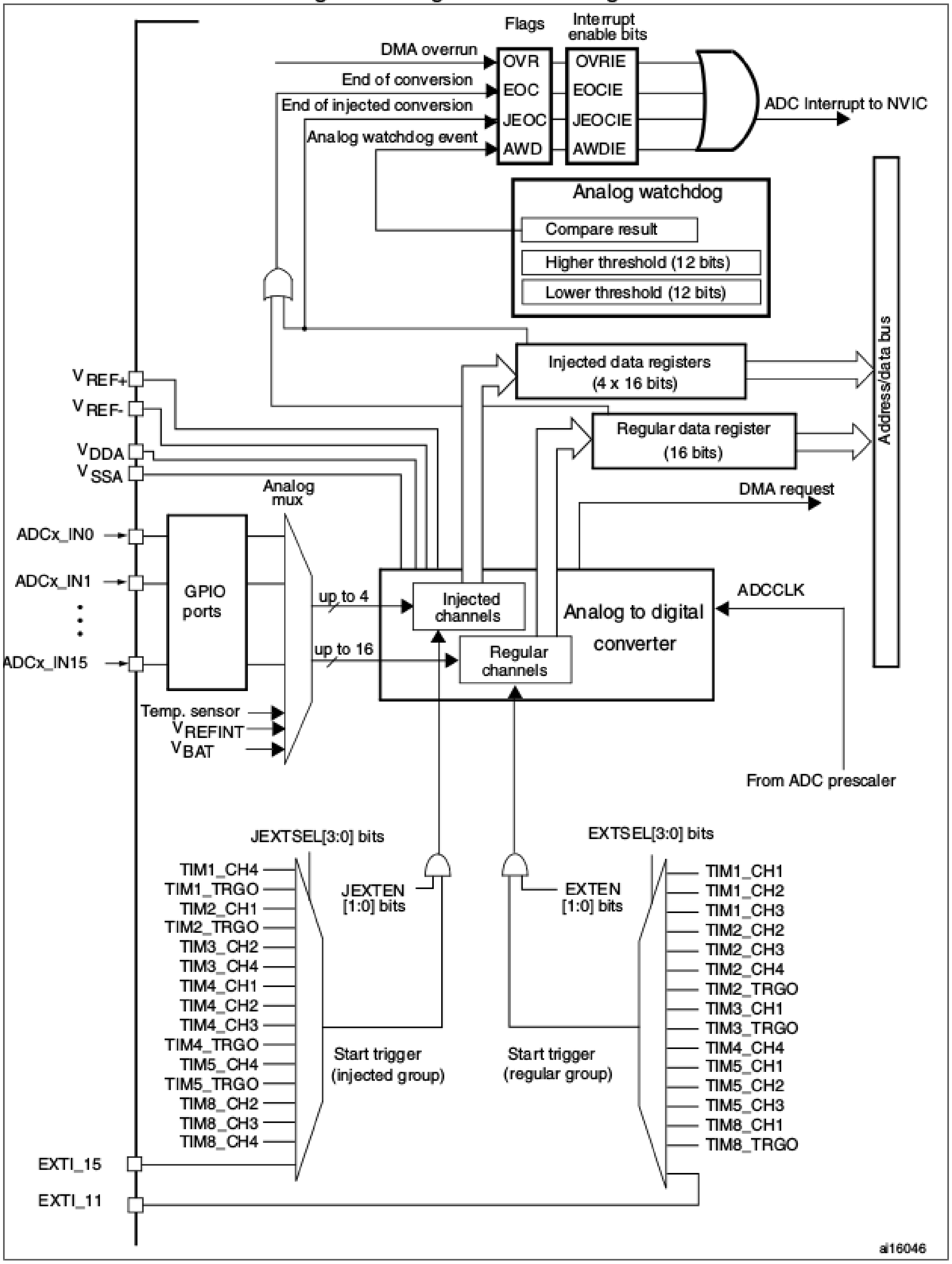
Since a computer is a digital device, any analog signal has be first converted to digital signal with a special device namely Analog-to-Digital converter (ADC). The resolution of the converter indicates the number of discrete values it can produce over the range of analog values. Thus, when we observed a value from ADC, the real voltage is a faction of value read multiplied by the reference voltage. For example, a 10-bit ADC can give a value between 0 and 1023. If a reference voltage is 3.3 volts, a 512 from ADC means 3.3*512/1023 volts.

STM32F4xx family's ADC

The 12-bit ADC is a successive approximation analog-to-digital converter. It has up to 19 multiplexed channels allowing it to measure signals from 16 external sources, two internal sources, and the V_{BAT} channel. The A/D conversion of the channels can be performed in single, continuous, scan or discontinuous mode. The result of the ADC is stored into a left- or right-aligned 16-bit data register.

The analog watchdog feature allows the application to detect if the input voltage goes beyond the user-defined, higher or lower thresholds.

Figure 44. Single ADC block diagram



Source : **Reference Manuals – RM0090**: STM32F405/415, STM32F407/417, STM32F427/437 and STM32F429/439 advanced ARM®-based 32-bit MCUs

Photoresistor or light-dependent resistor (LDR)

In this laboratory, we also use LDR (together with potentiometer) as an analog signal. "LDR is a light-controlled variable resistor. The resistance of a photoresistor decreases with increasing incident light intensity; in other words, it exhibits photoconductivity. A photoresistor can be applied in light-sensitive detector circuits, and light- and dark-activated switching circuits." definition taken from wikipedia.org

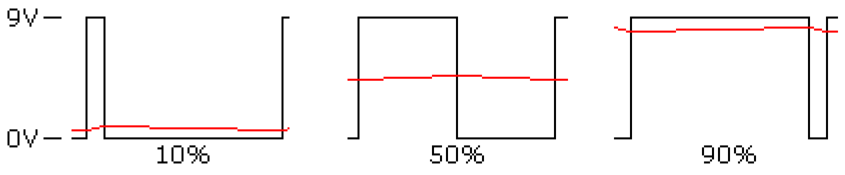
Pulse-width modulation (PWM)

Pulse-Width Modulation allows microcontroller to control the duty cycle of a pulse. It can be used to controller the brightness of LED, speed of motor, or any analog signal.

PWM can have many of the characteristics of an analog control system, in that the digital signal can be free wheeling. PWM does not have to capture data, although there are exceptions to this with higher end controllers.

One of the parameters of any square wave is duty cycle. Most square waves are 50%, this is the norm when discussing them, but they don't have to be symmetrical. The ON time can be varied completely between signal being off to being fully on, 0% to 100%, and all ranges between.

Shown below are examples of a 10%, 50%, and 90% duty cycle. While the frequency is the same for each, this is not a requirement.



Examples of PWM Waveforms

The reason PWM is popular is simple. Many loads, such as resistors, integrate the power into a number matching the percentage. Conversion into its analog equivalent value is straightforward. LEDs are very nonlinear in their response to current, give an LED half its rated current you you still get more than half the light the LED can produce. With PWM the light level produced by the LED is very linear. Motors, which will be covered later, are also very responsive to PWM.

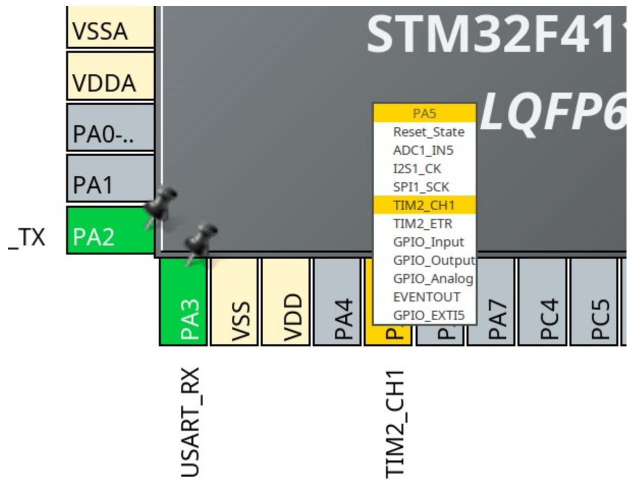
Source: http://www.allaboutcircuits.com/textbook/semiconductors/chpt-11/pulse-width-modulation/

Lab Exercises

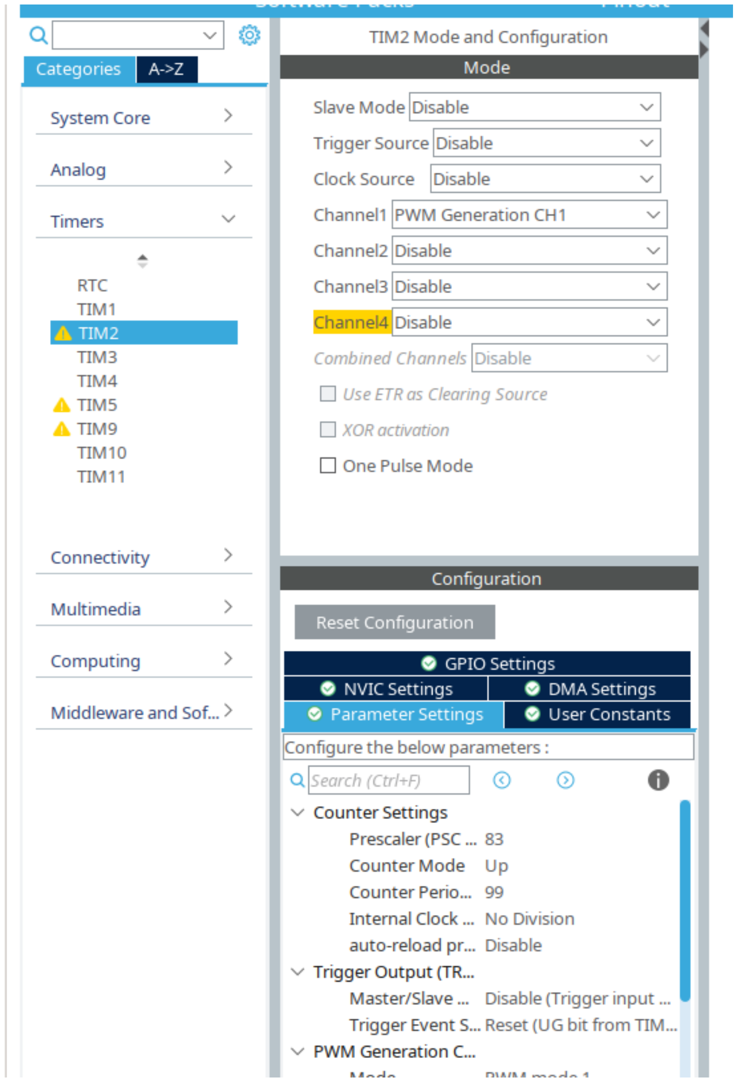
1 **Task:** Create a new STM32 Project to DIM an LED by increasing the duty cycle from 0% duty cycle by 1% every 0.01 second. When the PWM reaches the 100% duty cycle, decrease the duty cycle by 1% every 0.01 second to 0% duty cycle. Repeat the following step forever. (use 100 microseconds period for PWM)

Steps:

1.1 Set LED pin to be Timer. PC13 is corresponding to TIM2_CH1



1.2 Set Timer 2 Channel 1 parameters


TIM2 Mode and Configuration

Mode

Slave Mode: Disable
Trigger Source: Disable
Clock Source: Disable
Channel1: PWM Generation CH1
Channel2: Disable
Channel3: Disable
Channel4: Disable
Combined Channels: Disable
Use ETR as Clearing Source
XOR activation
One Pulse Mode

Configuration

Configure the below parameters :

Counter Settings
Prescaler (PSC ... 83
Counter Mode: Up
Counter Perio... 99
Internal Clock ... No Division
auto-reload pr... Disable
Trigger Output (TR...
Master/Slave ... Disable (Trigger input ...
Trigger Event S... Reset (UG bit from TIM...
PWM Generation C...

1.3 Setup code to turn on the LED according to TIM2 channel 1.

```
/* USER CODE BEGIN 2 */
HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);
// HAL has a function to config the output of channel using
// HAL_TIM_OC_ConfigChannel but it is harder to use
// For all intent purpose, you can access the pulse parameter here
TIM2->CCR1 = 50;
/* USER CODE END 2 */
```

Try to run, you should see the led on at about 50%. If you change 50 to 10, you should see the led on, but at a much less brightness.

1.4 Make it so that it changes brightness in time.

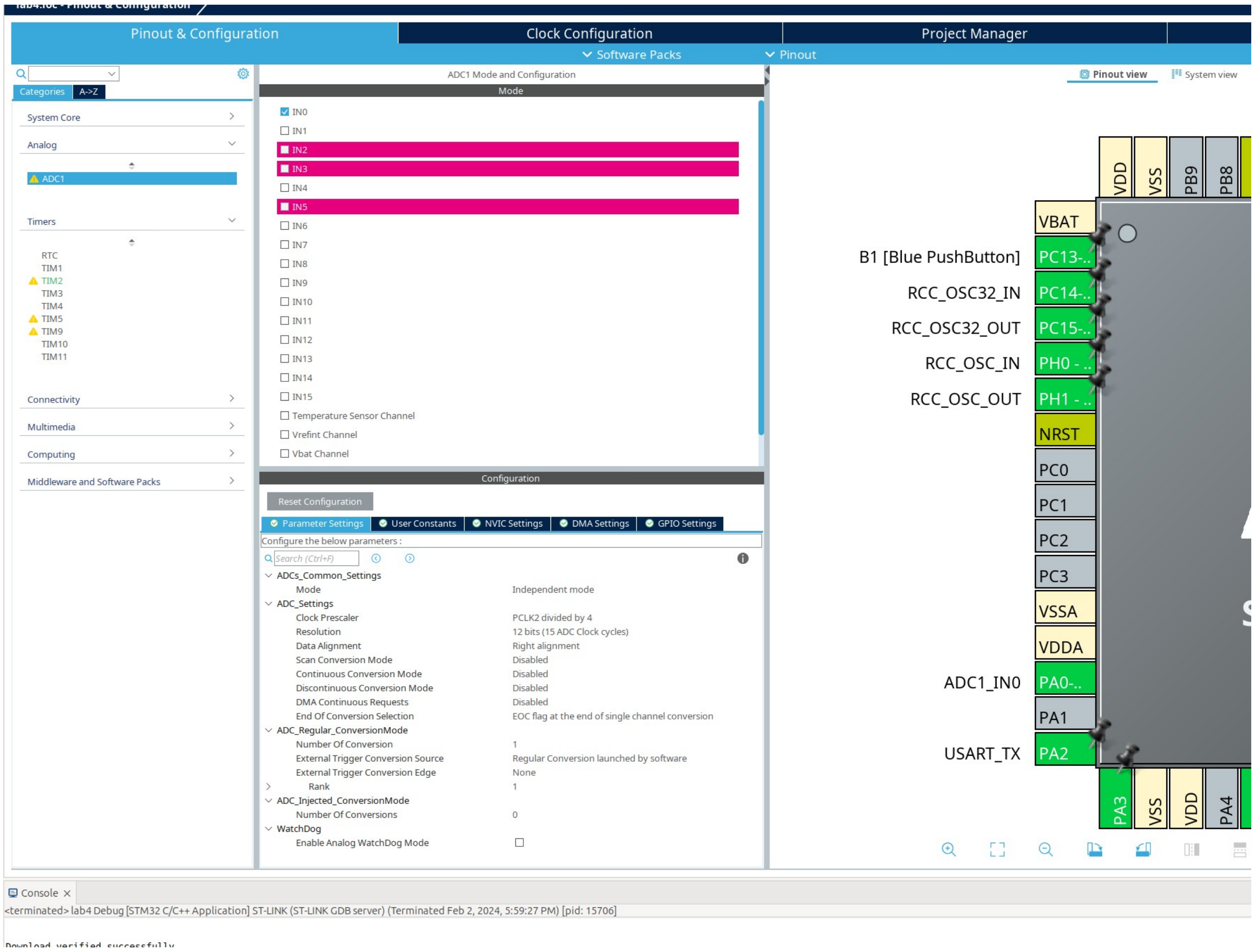
```
/* USER CODE BEGIN WHILE */
while (1)
{
      for(int i=0;i<100;i++) {
            TIM2->CCR1 = i;
            HAL_Delay(10);
      }
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
```

1.5 Complete the task.

2 **Task:** Connect LDR to STM32 Board and create a new STM32 Project to read the value from LDR and display to serial terminal via UART.

Steps:

2.1 Setup a PIN to accept an analog input: You can do this by going to IOC, then Choose Analog then select a pin in INx for example, we show the result of IN0 enable here.



[✓ IN0]

Configure the below parameters :

```
ADCs_Common_Settings
    Mode                                  Independent mode
ADC_Settings
    Clock Prescaler                       PCLK2 divided by 4
    Resolution                            12 bits (15 ADC Clock cycles)
    Data Alignment                        Right alignment
    Scan Conversion Mode                  Disabled
    Continuous Conversion Mode            Disabled
    Discontinuous Conversion Mode         Disabled
    DMA Continuous Requests               Disabled
    End Of Conversion Selection           EOC flag at the end of single channel conversion
ADC_Regular_ConversionMode
    Number Of Conversion                  1
    External Trigger Conversion Source    Regular Conversion launched by software
    External Trigger Conversion Edge      None
      Rank                                1
ADC_Injected_ConversionMode
    Number Of Conversions                 0
WatchDog
    Enable Analog WatchDog Mode           ☐
```

```
<terminated> lab4 Debug [STM32 C/C++ Application] ST-LINK (ST-LINK GDB server) (Terminated Feb 2, 2024, 5:59:27 PM) [pid: 15706]

Download verified successfully
```

Note that ADC1_IN0 is corresponding to PIN PA0.

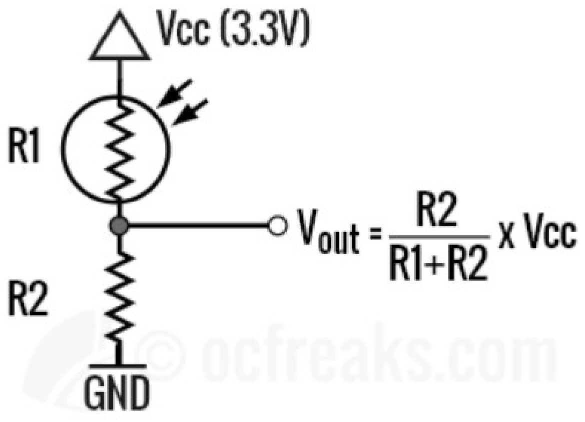
2.2 Setup analog reading: Here is an example how to

```
/* USER CODE BEGIN 2 */
int adcval = 0;
char buf[256];
/* USER CODE END 2 */
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
      // Start ADC conversion
      HAL_ADC_Start(&hadc1);
      // Wait for 1000ms or when the conversion is finished.
      if (HAL_ADC_PollForConversion(&hadc1, 1000) == HAL_OK) {
            // Read the ADC value
            adcval = HAL_ADC_GetValue(&hadc1);
            // Write integer to buffer
            sprintf(buf, "%d\r\n", adcval);
            // Transmitted with UART
            HAL_UART_Transmit(&huart2, buf, strlen(buf), 1000);
        }
    /* USER CODE END WHILE */
```

With this code, it should printout value from the ADC readout to UART terminal.

2.3 Connect LDR to Nucleo board. Make sure you unplug the USB before connecting anything

Wire the LDR as following



Note that Vout is connected to PA0 in our example (Pin A0 on Nucleo board)

If everything works, you should see the values of ADC readout in **UART** as high when there is a lot of light and low when there is no light.

3 Create a new STM32 Project to control a green LED brightness with PWM from the value of LDR with this equation.

PWM Duty Cycle = (1 – (C / P)) * 100 %

C = (max environment brightness) – (current environment brightness)

P = (max environment brightness) - (min environment brightness)