- Resources and Tools
  - Evaluation Boards & Kits
  - FPGA Reference Designs
  - Quick Start Guides
  - Linux Software Drivers
  - Microcontroller Software Drivers
  - ACE Software
  - Technical Guides
- Education Content
  - ADALM1000 (M1k) Active Learning Module
  - ADALM2000 (M2k) Active Learning Module
  - ADALM-PLUTO SDR Active Learning Module
  - Tutorials
  - On Line Electronics I/II
  - Labs for Electronics I/II
  - Energy/Power systems
- Wiki Help
  - Help
  - About Wiki
  - Playground
  - Wiki Site Map
- Wiki Tools
  - Recent Changes
  - Media Manager
  - Sitemap

- search wiki

This version (16 Jun 2020 14:34) was ***approved*** by AndreiD1994.

AD5770R PMOD Demo

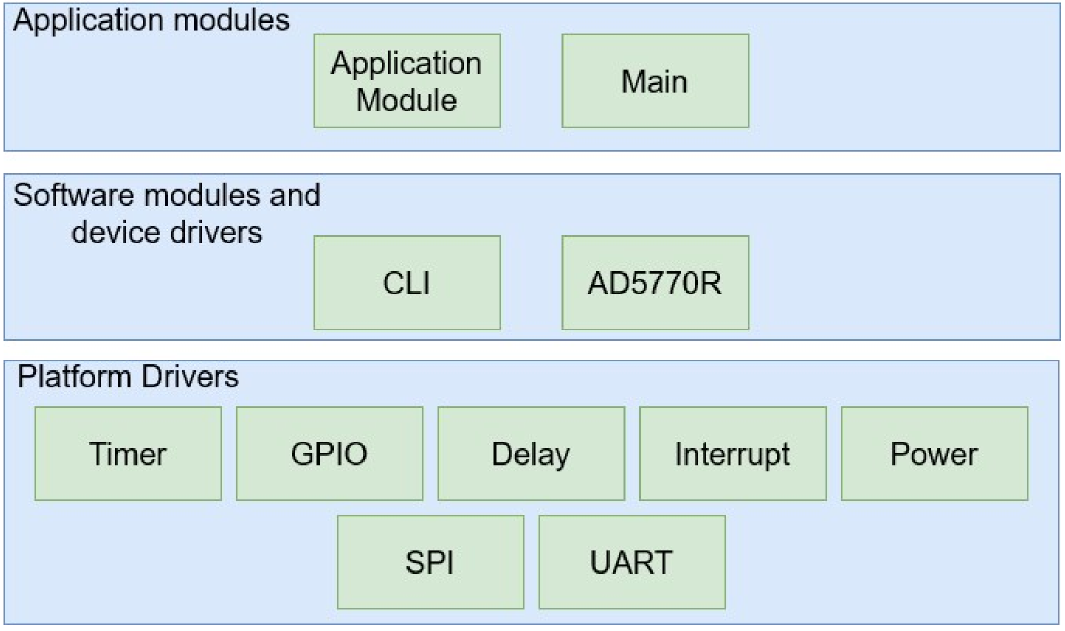
The **ADuCM3029_demo_ad5770rpmdz** project provides a solution to control the **EVAL-AD5770R-PMDZ PMOD** using a minimal **CLI** and the **no-OS** drivers for the **EVAL-ADICUP3029** platform.

General Description/Overview

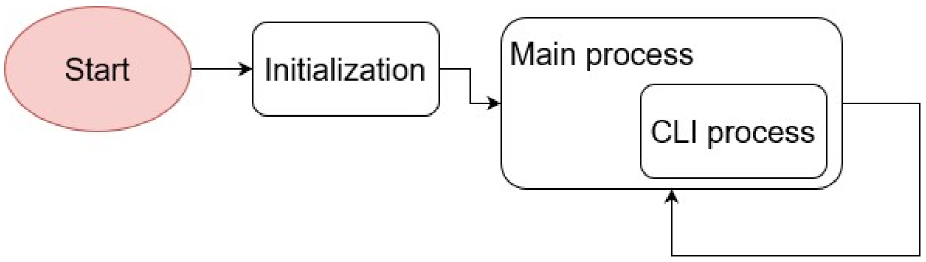
The AD5770R is a 6-channel, 14-bit, multi-range, current output **DAC** designed for use in communications systems, instrumentation and industrial applications; specifically for photonics control and current mode biasing. It has 6 programmable output channels with 1 channel capable of sinking up to 60 mA of current. The **EVAL-AD5770R-PMDZ** is a board designed to be a compact and low cost solution to evaluate the part.

The application builds upon the **no-OS** device and platform drivers and a minimal **CLI** module to provide a robust command set to set the range and output value of the channels.

Table of Contents



The program first initializes the hardware system as well as the driver handlers, then goes into the main process that just implements the CLI process and waits for user commands. If a command is received, it is executed and the program returns to the main loop.



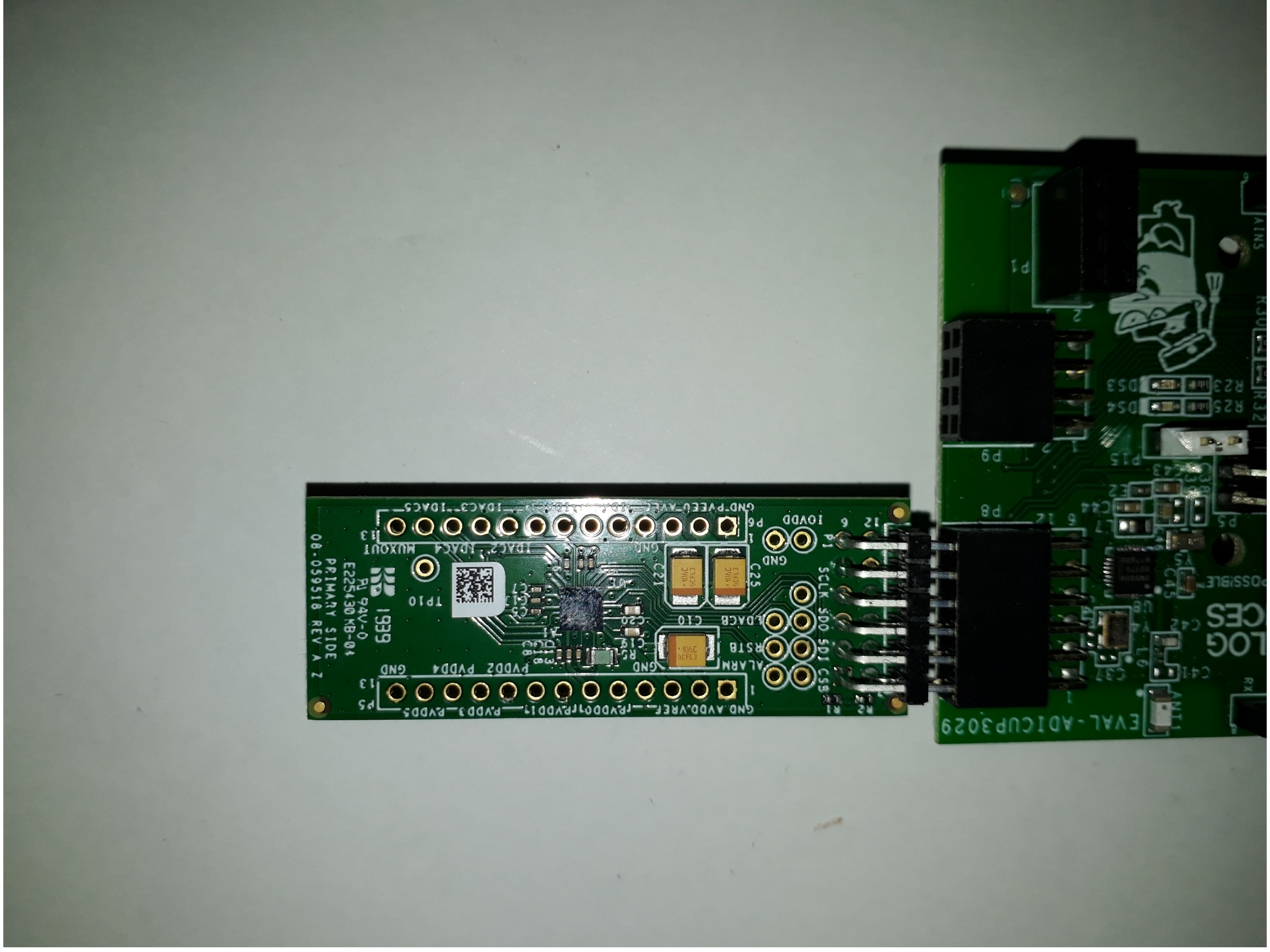
Demo Requirements

The following is a list of items needed in order to replicate this demo.

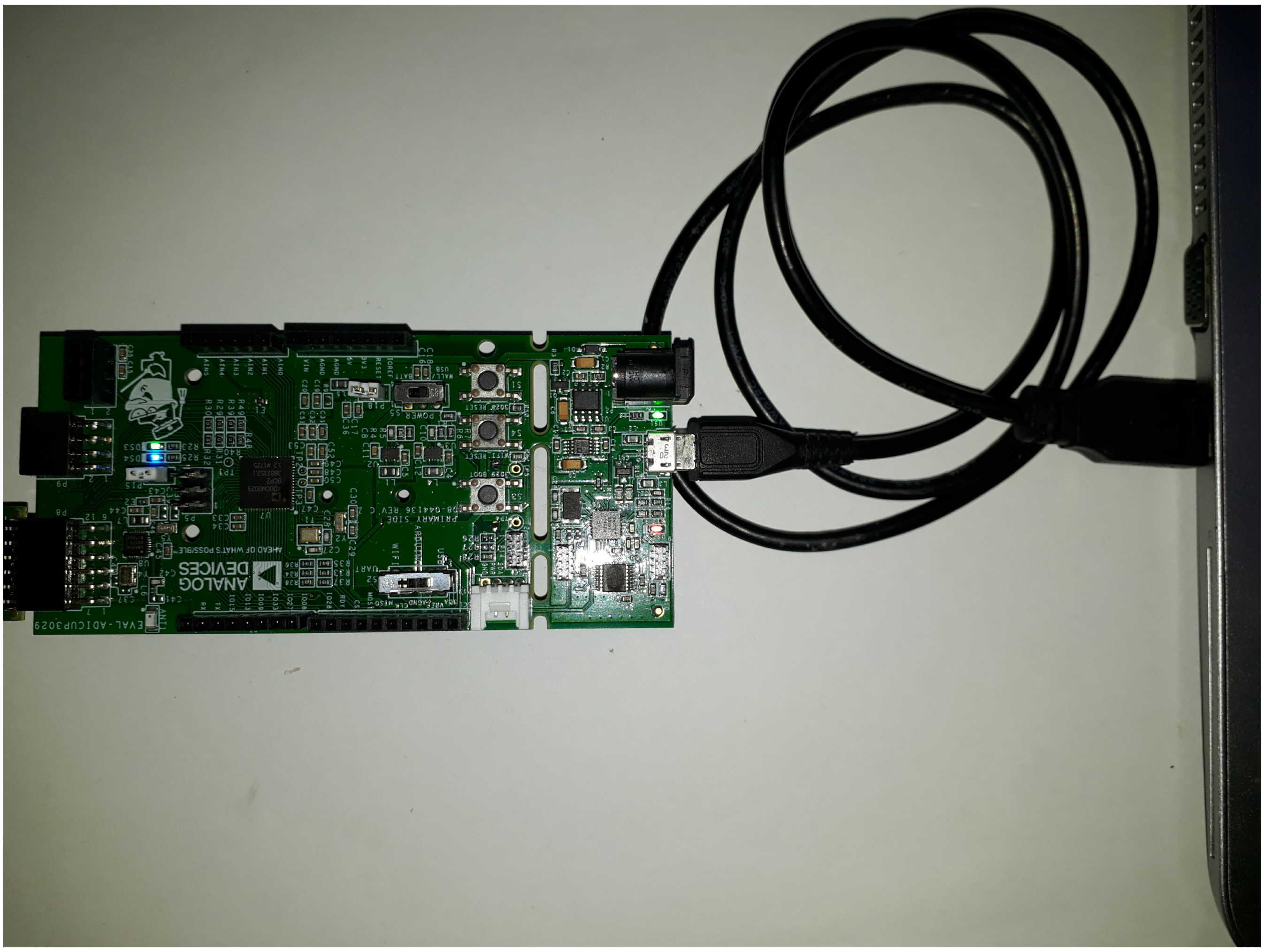
- Hardware
  - EVAL-ADICUP3029
  - EVAL-AD5770R-PMDZ
  - Mirco USB to USB cable
  - PC or Laptop with a USB port
- Software
  - ADuCM3029_demo_ad5770rpmdz
  - CrossCore Embedded Studio (2.9.1 or higher)
  - ADuCM302x DFP (3.2.0 or higher)
  - ADICUP3029 BSP (1.1.0 or higher)
  - Serial Terminal Program
    - Such as Putty or Tera Term

Setting up the Hardware

1. Connect **EVAL-AD5770R-PMDZ** board to the **EVAL-ADICUP3029** using connector **P8**.



2. Connect a micro- USB cable to P10 connector of the EVAL-ADICUP3029 and connect it to a computer. The final setup should look similar to the picture below.



Outputting Data

A serial terminal is an application that runs on a PC or laptop that is used to display data and interact with a connected device (including many of the Circuits from the Lab reference designs). The device's UART peripheral is most often connected to a UART to USB interface IC, which appears as a traditional COM port on the host PC/ laptop. (Traditionally, the device's UART port would have been connected to an RS-232 line driver / receiver and connected to the PC via a 9-pin or 25-pin serial port.) There are many open-source applications, and while there are many choices, typically we use one of the following:

- Tera Term
- Putty
- Real Term

Before continuing, please make sure you download and install one of the above programs.

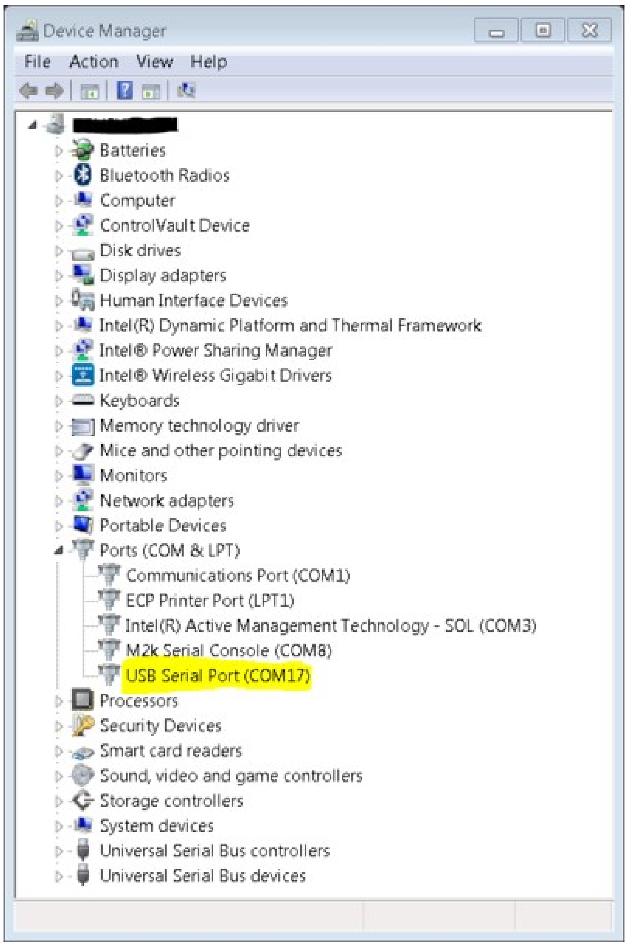
There are several parameters on all serial terminal programs that must be setup properly in order for the PC and the connected device to communicate. Below are the common settings that must match on both the PC side and the connected UART device.

1. **COM Port** - This is the physical connection made to your PC or Laptop, typically made through a USB cable but can be any serial communications cable. You can determine the COM port assigned to your device by visiting the device manager on your computer. Another method for identifying which COM port is associated with a USB-based device is to look at which COM ports are present before plugging in your device, then plug in your device, and look for a new COM port.
2. **Baud Rate** - This is the speed at which data is being transferred from the connected device to your PC. These parameters must be the same on both devices or data will be corrupted. The default setting for most of the reference designs in 115200.
3. **Data Bits** - The number of data bits per transfer. Typically UART transmits ASCII codes back to the serial port so by default this is almost always set to 8-Bits.
4. **Stop Bits** - The number of “stop” conditions per transmission. This usually set to 1, but can be set to 2 for redundancy.
5. **Parity** - Is a way to check for errors during the UART transmission. Unless otherwise specified, set parity to “none”.
6. **Flow Control** - Is a way to ensure that data lose between fast and slow devices on the same UART bus are not lost during transmission. This is typically not implemented in a simple system, and unless otherwise specified, set to “none”.

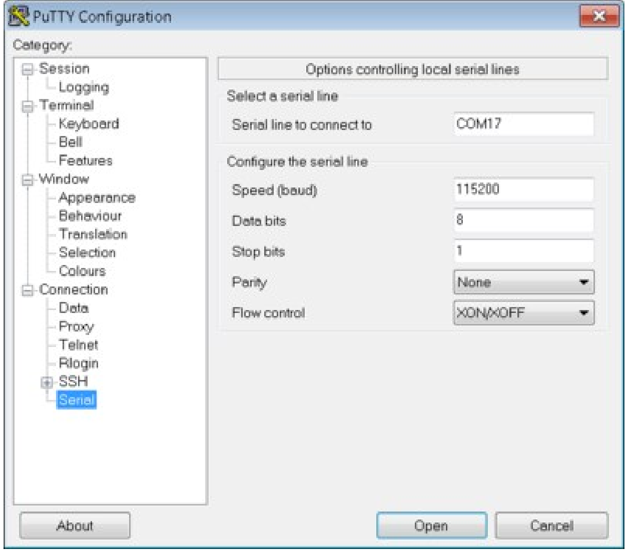
In many instances there are other options that each of the different serial terminal applications provide, such as **local line echo** or **local line editing**, and features like this can be turned on or off depending on your preferences. This setup guide will not go over all the options of each tool, but just the minor features that will make it easier to read back data from the connected devices.

Example setup using Putty

1. Plug in your connected device using a USB cable or other serial cable.
2. Wait for the device driver of the connected device to install on your PC or Laptop.
3. Open your device manager, and find out which COM port was assigned to your device.



4. Open up your serial terminal program (Putty for this example)
5. Click on the serial configuration tab or window, and input the settings to match the requirements of your connected device. The default baud rate for most of the reference designs is 115200. Make sure that is the selected baud rate as well.



Options controlling local serial lines

Select a serial line

Serial line to connect to: COM17

Configure the serial line

Speed (baud): 115200

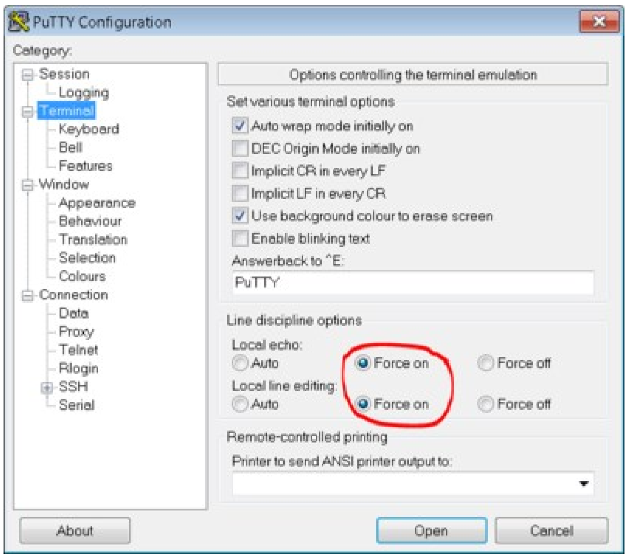
Data bits: 8

Stop bits: 1

Parity: None

Flow control: XON/XOFF

6. Ensure that local echo and line editing are enabled, so that you can see what you type and are able to correct mistakes. (Some devices may echo typed characters - if so, you will see each typed character twice. If this happens, turn off local echo.)



Options controlling the terminal emulation

Set various terminal options

- ✓ Auto wrap mode initially on
- DEC Origin Mode initially on
- Implicit CR in every LF
- Implicit LF in every CR
- ✓ Use background colour to erase screen
- Enable blinking text

Answerback to ^E: PuTTY

Line discipline options

Local echo: Auto [✓ Force on] Force off

Local line editing: Auto [✓ Force on] Force off

Remote-controlled printing

Printer to send ANSI printer output to:

7. Click on the open button, and as long as your connected device and serial terminal program are setup the same, than you should see data displaying.

Hint: If you see nothing in the serial terminal, try hitting the reset button on the embedded development board.

Available commands

Typing **help** or **h** after initial calibration sequence will display the list of commands and their short versions. Bellow is the short command list:

Function	Command	Description	Example
General commands			
	h	Display available commands.	
	t	Set channels at production test levels.	
DAC commands			
	sir	Set the input register of the chosen DAC channel. *<chan>* = channel to update; values are: c0, c1, c2, c3, c4, c5. *<value>* = update value in decimal; between 0 and 16383.	sir c0 8192
	uo	Update the DAC output with the channel input registers value. This is done with the nLDAC GPIO if it's available and with the SW register otherwise.	
	sc	Set the value of the chosen DAC channel. *<chan>* = channel to update; values are: c0, c1, c2, c3, c4, c5; *<value>* = update value in decimal; between 0 and 16383.	sc c0 8192
	sr	Set the range of the DAC output channels. *<chan>* = channel to update; values are: c0, c1, c2, c3, c4, c5; *<opt>* = option chosen for the channel. values are: opt1 - only for channels 0 and 1, opt2, opt3. Those correspond to the options in the datasheet.	sr c0 opt1

Obtaining the Source Code

We recommend not opening the project directly, but rather import it into CrossCore Embedded Studios and make a local copy in your workspace.

The source code and include files of the **ADuCM3029_demo_ad5770rpmdz** can be found here:

ADuCM3029_demo_ad5770rpmdz at Github

How to use the Tools

The official tool we promote for use with the EVAL-ADICUP3029 is CrossCore Embedded Studio. For more information on downloading the tools and a quick start guide on how to use the tool basics, please check out the Tools Overview page.

Importing

For more detailed instructions on importing this application/demo example into the CrossCore Embedded Studios tools, please view our How to import existing projects into your workspace section.

Debugging

For more detailed instructions on importing this application/demo example into the CrossCore Embedded Studios tools, please view our How to configure the debug session section.

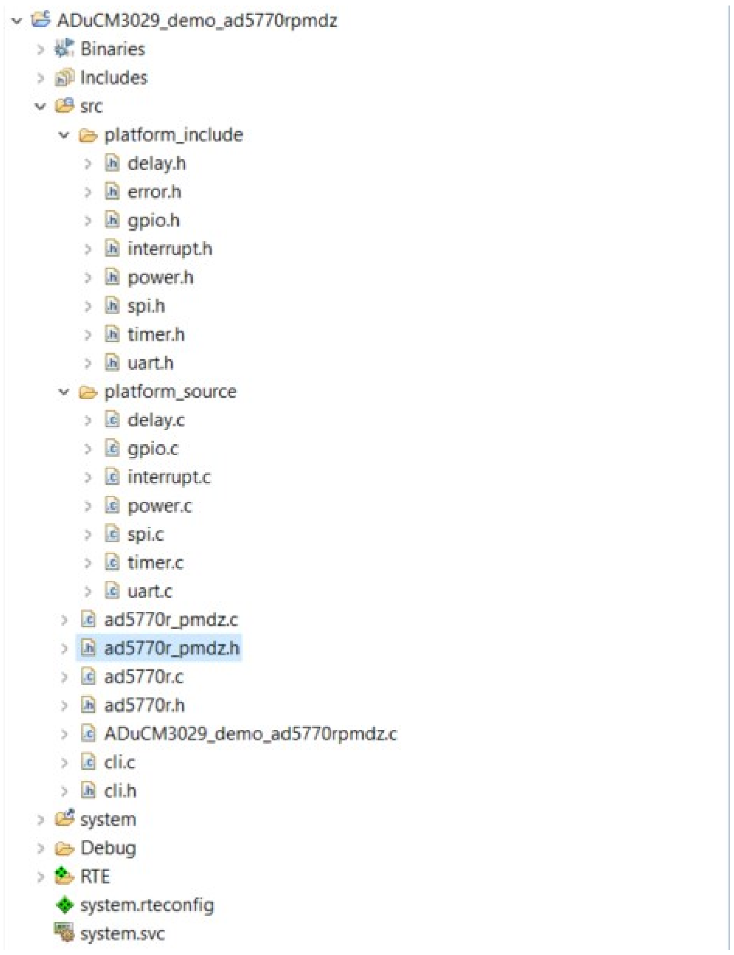
Project Structure

Beside the IDE generated sources the project structure is divided into high level software modules and low level software modules.

The high level modules are in the **src** folder and are:

- AD5770R device driver;
- CLI module;
- AD5770R_PMDZ module (application source)
- ADuCM3029_demo_ad5770rpmdz.c (main file)

The low level modules are the platform drivers and are included in the **platform_source** and **platform_include** folders.



```
ADuCM3029_demo_ad5770rpmdz
  Binaries
  Includes
  src
    platform_include
      delay.h
      error.h
      gpio.h
      interrupt.h
      power.h
      spi.h
      timer.h
      uart.h
    platform_source
      delay.c
      gpio.c
      interrupt.c
      power.c
      spi.c
      timer.c
      uart.c
    ad5770r_pmdz.c
    ad5770r_pmdz.h
    ad5770r.c
    ad5770r.h
    ADuCM3029_demo_ad5770rpmdz.c
    cli.c
    cli.h
  system
  Debug
  RTE
  system.rteconfig
  system.svc
```

End of Document