Analog Devices Wiki

This version (24 Jan 2020 09:48) was ***approved*** by Michael Bradley [https://ez.analog.com/members/mbradley].
The Previously approved version (/resources/tools-software/product-support-software/ad7124-stm32?rev=1564159884) (26 Jul 2019 16:51) is available.

AD7124 Example on STM32 Processors

Introduction

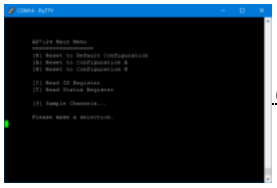
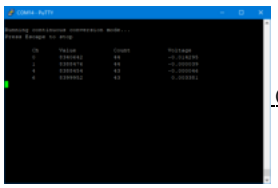
This describes how to take the AD7124 example code and integrate it with STM32 firmware libraries in a suitable development environment to produce a complete program. The IDE (Integrated Drive Electronics (hard drives!)) used here is the STM32CubeIDE, but the general procedure can be applied to other IDEs.

Useful links

- AD7124 No-OS Software (/resources/tools-software/uc-drivers/ad7124)
- AD7124-8 Product Page [https://www.analog.com/AD7124-8]
- AD7124-4 Product Page [https://www.analog.com/AD7124-4]
- AD7124-8 Evaluation Board [https://www.analog.com/EVAL-AD7124-8]
- AD7124-4 Evaluation Board [https://www.analog.com/EVAL-AD7124-4]

Overview

The AD7124 [https://www.analog.com/AD7124-8] is a low power, low noise, completely integrated analog front end for high precision measurement applications. The device contains a low noise, 24-bit Σ-Δ analog-to-digital converter (ADC). The AD7124 example application provides a terminal based console interface that allows a user to select between different configurations, and to sample data in single or continuous conversion modes.


(/_detail/resources/tools-software/product-support-software/ad7124_example/ad7124_main_menu.png?id=resources%3Atools-software%3Aproduct-support-software%3Aad7124-stm32)

(/_detail/resources/tools-software/product-support-software/ad7124_example/ad7124_continuous_conversion.png?id=resources%3Atools-software%3Aproduct-support-software%3Aad7124-stm32)

The example makes use of the AD7124 No-OS (/resources/tools-software/uc-drivers/ad7124) software drivers and platform drivers that are using the STM32 HAL firmware libraries.

The example code was developed and tested using the Nucleo-L476RG with version 1.14.0 of the STM32 firmware libraries, and STMCube32IDE v1.0.0. However, it may be re-targeted to other STM32 processors/boards, through the use of the appropriate ST firmware HAL libraries.

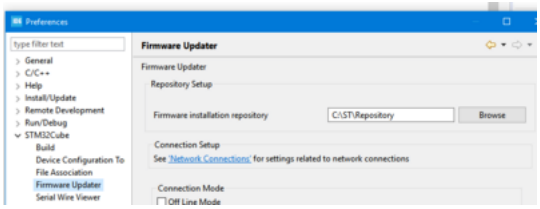
Software Integration Guide

Downloads

- ad7124_stm32_example.zip (/_media/resources/tools-software/product-support-software/ad7124_example/ad7124_stm32_example.zip)

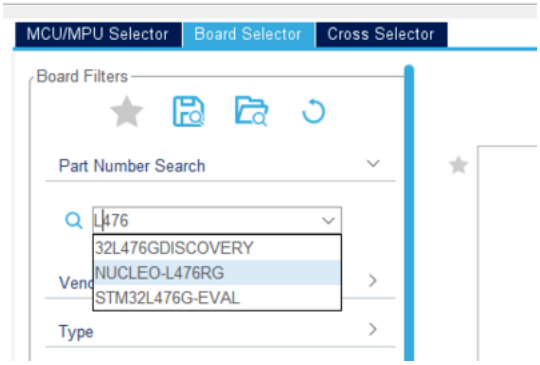
Project Creation

- If you have not already done so, install the STM32CubeIDE available from www.st.com [http://www.st.com].
- In the Firmware Update section of the STM32CubeIDE preferences, set the location to where the firmware package is going to be stored.
  - You may want to place this in a common location, e.g. (for example) 'C:\ST\Repository' and not in a directory located under your home directory, to avoid user specific paths in any shared configuration files



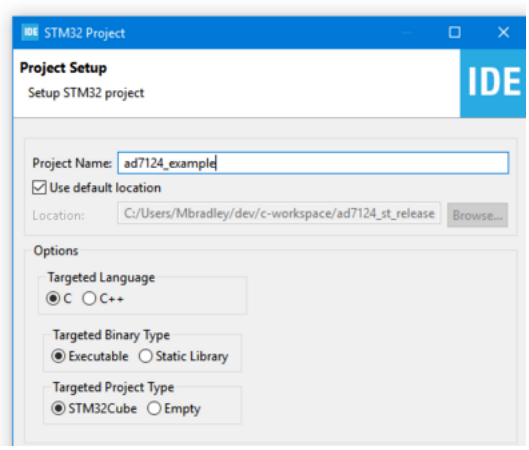
(/_detail/resources/tools-software/product-support-software/ad7124_example/ad7124_firmware_repo.png?id=resources%3Atools-software%3Aproduct-support-software%3Aad7124-stm32)

- Select the File » New » STM32 project menu option.
- Select the MCU part number or Board being used



(/_detail/resources/tools-software/product-support-software/ad7124_example/ad7124_select_l476.png?id=resources%3Atools-software%3Aproduct-support-software%3Aad7124-stm32)

- Give the Project a name, select target language, and project type of STM32Cube



(/_detail/resources/tools-software/product-support-software/ad7124_example/ad7124_project_setup.png?id=resources%3Atools-software%3Aproduct-support-software%3Aad7124-stm32)

- Make sure the Target Reference is correct as it can't be changed once the project is created, and the firmware package repository path is as expected. Set the Code Generator Options to reference library files, or copy library files in your project, depending on how you want to structure your project.
- When you click Finish, the STM32CudeIDE will download the firmware library if required and unzip it to the repository location specified above. This file is typically 100's MB (megabyte) in size so this will take several minutes to complete.

Configuring the Project

The Device Configuration Tool with automatic code generation is used to define pin usage and other default modes of operation for the NUCLEO-L476. In addition there are some build and linker settings that may be required depending on the default project build configuration.

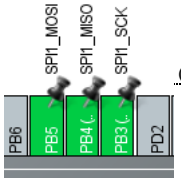
The pins and configuration provided here need to be tailored to the specific board/processor being used.

Device Configuration Tool

A 4-wire SPI (Serial Peripheral Interface) bus is used to connect AD7124 to the NUCLEO-L476RG board, and a UART (universal asynchronous receiver/transmitter) is used to provide the serial I/O for the console interface. An LED is also used to indicate activity. The following sections detail the configuration settings that need to be made for each of these.

SPI
SPI1 port on the processor is used to communicate with the AD7124, with the pin assigned to each function as shown here, with the corresponding label.

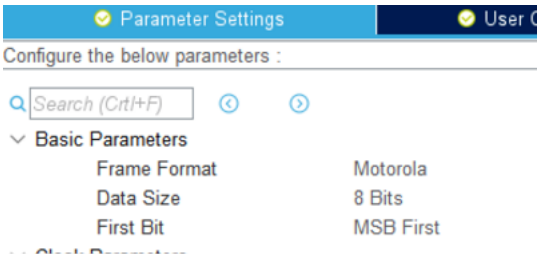
It is recommended that a pull-up resistor be used on the SPI (Serial Peripheral Interface) MOSI to ensure it is never floating in an undefined logic state. This can be a resistor on the board to the logic supply, or internal to the processor if available.



(/_detail/resources/tools-software/product-support-software/ad7124_example/ad7124_spi_l476.png?id=resources%3Atools-software%3Aproduct-support-software%3Aad7124-stm32)

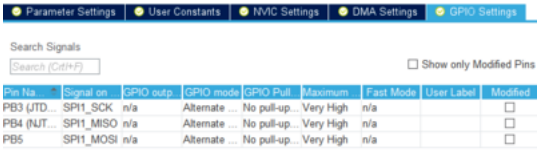
(/_detail/resources/tools-software/product-support-software/ad7124_example/ad7124_csb_l476.png?id=resources%3Atools-software%3Aproduct-support-software%3Aad7124-stm32)

The pin PB10 is used as a software controlled chips select for SP1, and so its mode must be set to GPIO (General Purpose Input/Output) _output, and set the user label to SPI1_NSS to match what is used in the platform driver file. The Connectivity » SPI1 configuration settings are shown here.



(/_detail/resources/tools-software/product-support-software/ad7124_example/ad7124_spi_config.png?id=resources%3Atools-software%3Aproduct-support-software%3Aad7124-stm32)

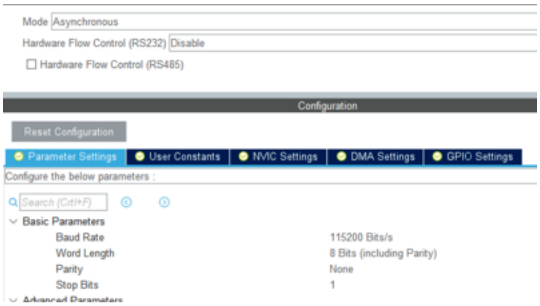
DMA (Direct Memory Access) and interrupts are not used and don't need to be configured. The SPI1 GPIO (General Purpose Input/Output) Settings are as follows.



(/_detail/resources/tools-software/product-support-software/ad7124_example/ad7124_spi_gpios.png?id=resources%3Atools-software%3Aproduct-support-software%3Aad7124-stm32)

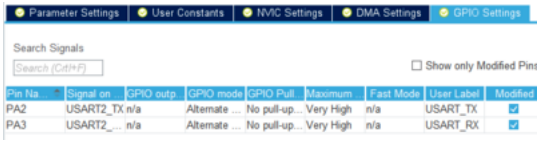
Serial UART

The serial port uses USART2, and no DMA (Direct Memory Access) or interrupts need to be configured.



(/_detail/resources/tools-software/product-support-software/ad7124_example/ad7124_uart_config.png?id=resources%3Atools-software%3Aproduct-support-software%3Aad7124-stm32)

The USART2 GPIO (General Purpose Input/Output) settings are as shown.



(/_detail/resources/tools-software/product-support-software/ad7124_example/ad7124_uart_gpios.png?id=resources%3Atools-software%3Aproduct-support-software%3Aad7124-stm32)
GPIO
An LED is toggled on the NUCLEO-L476 board to indicate sampling and other activity. The activity LED is controlled by Port A, Pin 5, and needs to enabled as digital output to support this function.

Build Settings

The printf(…) function is used to print numbers formatted as floating point values in the terminal view. As this feature is often disabled by default due to the additional memory requirement, floating point support in printf(…) must be enabled. In the Project properties window, under 'C/C++ Build » Settings » MCU GCC (GNU Compiler Collection) linker » Miscellaneous > Other Flags' add a '-u _printf_float' option.

> There is also a checkbox option to enable this in the MCU Settings view as well, but doing it in the linker section, seems to prevent the code analysis feature in the SMT32CubeIDE reporting '%f' as not supported in the code editor window. There is no problem enabling it in both locations in STM32CubeIDE.

If there are other source or include directories that need to be added to support the project build, they should also be added to the relevant 'Include paths' in the MCU GCC (GNU Compiler Collection) Compiler section as required.

Linker Files

The default value for _estack may be incorrect in the *.ld files. This can cause problems when calling into certain library functions. In particular this can prevent the %f format specifier working with the floating point version of printf(…). If instead of a value like '1.23', the terminal output is '0.00', this can indicate a need to update the linker *.ld files. For the NUCLEO-L476RG, the RAM and the FLASH versions of the ld files contain the following:

```
/* Highest address of the user mode stack */
_estack = 0x20017fff; /* end of "RAM" Ram type memory */
```

Changing this as follows fixes the issues related to floating point support in printf:

```
/* Highest address of the user mode stack */
_estack = 0x20018000; /* end of "RAM" Ram type memory */
```

Source File Edits

When using the Device Configuration Tool, the code generator produces a two of these source files, main.c and main.h need minor edits to integrate the AD7124 example code. There may be an edit required to the _read(…) function in syscalls.c to work around an issue, but whether this is required, will depend on the specific library and build environment.

main.c

To keep the integration of the AD7124 example application with other user and platform specific code, there are only two functions that a user needs to call from their own code, typically as part of the main function.

- ad7124_app_initialize(..) that does all the one-time initialization work required by the app, mainly AD7124 device setup
  - It is strongly recommended to test the return value from this function to determine if the initialization was successful or not
  - A value less than 0 indicate failure.

```
  /* Initialize the AD7124 application before the main loop */
int32_t setupResult;
if ((setupResult = ad7124_app_initialize(AD7124_CONFIG_A)) < 0 ) {
    // Handle error setting up AD7124 here
}
```

- adi_do_console_menu(…) displays the user menu to interact with the application features
  - This can be called in a while(1) loop so that it is always displayed.

```
  while(1) {
    // display the console menu for the AD7124 application
    adi_do_console_menu(&ad7124_main_menu);
}
```

Both are defined in the “ad7124_console_app.h” header which needs to be added as #include file.

main.h

The AD7124 example assumes that all the STM32 hardware is initialized and appropriate SPI (Serial Peripheral Interface) and UART (universal asynchronous receiver/transmitter) port handles are available, and are used in platform_drivers.c and platform_support.c. The following extern declarations for the SPI (Serial Peripheral Interface) and serial port handles are required in main.h to make them available to the platform specific code.

```
extern SPI_HandleTypeDef hspi1;
extern UART_HandleTypeDef huart2;
```

> The names of the port handles are defined by the Device Configuration Tool based on the selected processor and pin choices. If using a different processor or pins, these may need to be changed in the platform_drivers.c and platform_support.c files so they match up.

syscalls.c

In the _read(…) function, the 'len' parameter passed in was found to always be '1024' for the library and build environment used to develop the example code.

```
int _read(int file, char *ptr, int len)
```

In order to support the use of getchar(), the expression 'len = 1;' was added immediately before the for loop in the _read(…) function. While this is sufficient for getchar() to work, it does not support use of other stdio.h functions such as scanf(…).

Adding AD7124 Example Files

The distribution of the AD7124 source and header files can be added to the project that has been created. The files can be added in a dedicated 'adi' directory, or in the main 'src' directory, or split as appropriate between 'src' and 'inc' directories, according to the file structure being used. If adding new source and header file locations, then these will need to be added to the build settings as necessary, in the relevant 'Include paths' in the MCU GCC (GNU Compiler Collection) Compiler configuration.

The platform_support.c/.h files provide the necessary definitions and declarations of io_getchar(…) and io_putchar(…) to read/write characters over the serial port to the connected terminal. If using a compiler other than GCC (GNU Compiler Collection), or a different serial port than USART2 then there may be additional changes required to get serial I/O working. Examples included in the ST Firmware download can provide guidance.

At this point, assuming that any necessary changes, pin names, port usage/configuration have been made, the project should compile cleanly.

Hardware Connections

Power & USB

A 9V DC supply (barrel jack, center pin positive) is required to power the EVAl-AD7124-8SDZ evaluation board. The NUCLEO-476RG is powered via the USB (Universal Serial Bus) connection to the PC, which also provides the serial UART (universal asynchronous receiver/transmitter) connection back to the PC. The NUCLEO-476 creates a COM port that can be connected to by a terminal emulator, e.g. (for example) putty.

If you are unsure what COM port to use to communicate with the board, open Device Manager, and look under the 'Ports (COM & LPT)' node.

SPI Interface

SPI (Serial Peripheral Interface) connections to the host processor board can be made to the relevant test points on the eval board, or more easily with an SDP Breakout Board [https://www.analog.com/SDP-BREAKOUT-BOARD].

(/_detail/resources/tools-software/product-support-software/ad7124_example/ad7124_digital_interface.png?id=resources%3Atools-software%3Aproduct-support-software%3Aad7124-stm32)

AD7124 SPI (Serial Peripheral Interface) Signal	SDP Breakout Board	NUCLEO-L476
GND	81	GND on CN5.7
SCLK (System Clock)	82	D3 (PB3) on CN9.4
DOUT/RDYB	83	D5 (PB4) on CN9.6
DIN	84	D4 (PB5) on CN9.5
CSB	85	D6 (PB10) on CN9.7

Analog Input

The screw terminal connections to J6 and J11 can be used to connect appropriate analog input signals to provide test stimulus to the AD7124.

In Configuration A

- AIN0/AIN1 are used for channel 0, simple voltage measurement

In Configuration B

- AIN2/AIN3 go to the A2 thermocouple connector on the evaluation board, and are captured on channel 0. This uses an internal reference and has a bias voltage enabled on AIN2. A suitable thermocouple should connected to A2 for this measurement.
- AIN4/AIN5 are an RTD1000 measurement on channel 1. Excitation is provided from AIN1 for this. This requires an external RTD and reference resistor connected as show in the figure below.

(/_detail/resources/tools-software/product-support-software/ad7124_example/ad7124_rtd_thermocouple_connections.png?id=resources%3Atools-software%3Aproduct-support-software%3Aad7124-stm32)

Console Application

Once the hardware connections are made, and the compiled code programmed into the board, open the terminal program, and reset the hardware to see the AD7124 menu that allows a user to perform a variety of functions. These include reset the device, program one of the pre-defined configurations, and sample data that is displayed on screen or streamed so it can captured by the console.

(/_detail/resources/tools-software/product-support-software/ad7124_example/ad7124_main_menu.png?id=resources%3Atools-software%3Aproduct-support-software%3Aad7124-stm32)