Analog.com(//www.analog.com) **Engineer**Zone(//ez.analog.com) **Analog**Dialogue(//www.analog.com/en/analog-dialogue.html)

ANALOG DEVICES (/start) myAnalog (//my.analog.com/) **Log Out**(/resources/tools-software/uc-drivers/ad7124?do=logout&sectok=)

Wiki

Resources and Tools (/resources) Education Content (/university/courses/tutorials/index) Wiki Help (/wiki/help) Wiki Tools

search wiki

Analog Devices Wiki

This version (05 Apr 2023 08:46) was ***approved*** by Darius B [https://ez.analog.com/members/buha].
The Previously approved version (/resources/tools-software/uc-drivers/ad7124?rev=1563783430) (22 Jul 2019 08:17) is available.

AD7124 No-OS Software

You're looking at a legacy page. The driver API (Application Programming Interface) has since changed and does not correspond anymore to the driver for ad7124 available in no-OS (Operating System) repository. However, you may use the code in the following no-OS (Operating System) projects to see examples of ad7124 driver usage:

no-OS/tree/master/projects/ad7124-4sdz [https://github.com/analogdevicesinc/no-OS/tree/master/projects/ad7124-4sdz]

no-OS/tree/master/projects/ad7124-8pmdz [https://github.com/analogdevicesinc/no-OS/tree/master/projects/ad7124-8pmdz]

Introduction

This document describes the No-OS (Operating System) software used to control the AD7124 part and includes an example of how to initialize a AD7124 part.

Overview

The AD7124 [https://www.analog.com/AD7124-4] 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).

AD7124-4 [https://www.analog.com/AD7124-4] 4 differential / 7 single-ended inputs, 24-lead TSSOP / 32-lead LFCSP
AD7124-8 [https://www.analog.com/AD7124-8] 8 differential / 15 single-ended inputs, 32-lead LFCSP

The on-chip low gain stage ensures that signals of small amplitude can be interfaced directly to the ADC.
One of the major advantages of the AD7124 [https://www.analog.com/AD7124-4] is that it gives the user the flexibility to employ one of three integrated power modes. The current consumption, range of output data rates, and rms (root-mean-squared) noise can be tailored with the power mode selected. The device also offers a multitude of filter options, ensuring that the user has the highest degree of flexibility.
The AD7124 [https://www.analog.com/AD7124-4] can achieve simultaneous 50 Hz and 60 Hz rejection when operating at an output data rate of 25 SPS (samples per second) (single cycle settling), with rejection in excess of 80 dB (decibel) achieved at lower output data rates.
The AD7124 [https://www.analog.com/AD7124-4] establishes the highest degree of signal chain integration. The device contains a precision, low noise, low drift internal band gap reference, and also accepts an external differential reference, which can be internally buffered. Other key integrated features include programmable low drift excitation current sources, burnout currents, and a bias voltage generator, which sets the common-mode voltage of a channel to AVDD/2. The low-side power switch enables the user to power down bridge sensors between conversions, ensuring the absolute minimal power consumption of the system. The device also allows the user the option of operating with either an internal clock or an external clock.
The integrated channel sequencer allows several channels to be enabled simultaneously, and the AD7124 [https://www.analog.com/AD7124-4] sequentially converts on each enabled channel, simplifying communication with the device. As many as 16 channels can be enabled at any time; a channel being defined as an analog input or a diagnostic such as a power supply check or a reference check. This unique feature allows diagnostics to be interleaved with conversions. The AD7124 [https://www.analog.com/AD7124-4] also supports per channel configuration. The device allows eight configurations or setups. Each configuration consists of gain, filter type, output data rate, buffering, and reference source. The user can assign any of these setups on a channel by channel basis.
The AD7124 [https://www.analog.com/AD7124-4] also has extensive diagnostic functionality integrated as part of its comprehensive feature set. These diagnostics include a cyclic redundancy check (CRC), signal chain checks, and serial interface checks, which lead to a more robust solution. These diagnostics reduce the need for external components to implement diagnostics, resulting in reduced board space needs, reduced design cycle times, and cost savings. The failure modes effects and diagnostic analysis (FMEDA) of a typical application has shown a safe failure fraction (SFF) greater than 90% according to IEC 61508.
The device operates with a single analog power supply from 2.7 V (volt) to 3.6 V (volt) or a dual 1.8 V (volt) power supply. The digital supply has a range of 1.65 V (volt) to 3.6 V (volt). It is specified for a temperature range of −40°C to +105°C.

Supported Devices

- AD7124-4 [https://www.analog.com/AD7124-4]
- AD7124-8 [https://www.analog.com/AD7124-8]

Evaluation Boards

- EVAL-AD7124-4 [https://www.analog.com/EVAL-AD7124-4]
- EVAL-AD7124-8 [https://www.analog.com/EVAL-AD7124-8]

Driver Description

The driver contains two parts:

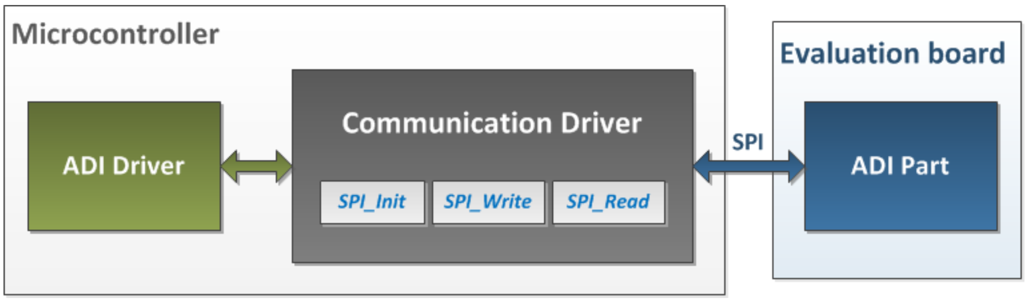
- The driver for the AD7124 part, which may be used, without modifications, with any microcontroller.

- The Communication Driver, where the specific communication functions for the desired type of processor and communication protocol have to be implemented. This driver implements the communication with the device and hides the actual details of the communication protocol to the ADI (Analog Devices, Inc.) driver.

The Communication Driver has a standard interface, so the AD7124 driver can be used exactly as it is provided.

There are three functions which are called by the AD7124 driver:

- SPI (Serial Peripheral Interface)_Init() – initializes the communication peripheral.
- SPI (Serial Peripheral Interface)_Write() – writes data to the device.
- SPI (Serial Peripheral Interface)_Read() – reads data from the device.



(/_detail/resources/tools-software/uc-drivers/spi_architecture.png?id=resources%3Atools-software%3Auc-drivers%3Aad7124)

SPI (Serial Peripheral Interface) driver architecture

The AD7124 driver contains the following:

- AD7124.h - Header file of the driver. Contains the driver function declarations, custom data types to be used by the driver and driver specific constants.
- AD7124.c - Implementation file of the driver. Contains the implementations of the driver functions.
- AD7124_regs.h - Register map of the driver. Contains the register map definitions and an array of all device registers to be used with the driver.

The following functions are implemented in this version of AD7124 driver:

Function	Description
`int32_t AD7124_ReadRegister(ad7124_device *device, ad7124_st_reg* pReg)`	Reads the value of the specified register.
`int32_t AD7124_WriteRegister(ad7124_device *device, ad7124_st_reg reg)`	Writes the value of the specified register.
`int32_t AD7124_NoCheckReadRegister(ad7124_device *device, ad7124_st_reg* pReg)`	Reads the value of the specified register without a device state check.
`int32_t AD7124_NoCheckWriteRegister(ad7124_device *device, ad7124_st_reg reg)`	Writes the value of the specified register without a device state check.
`int32_t AD7124_Reset(ad7124_device *device)`	Resets the device.
`int32_t AD7124_WaitForSpiReady(ad7124_device *device, uint32_t timeout)`	Waits until the device can accept read and write user actions.
`int32_t AD7124_WaitToPowerOn(ad7124_device *device, uint32_t timeout)`	Waits until the device finishes the power-on reset operation.
`int32_t AD7124_WaitForConvReady(ad7124_device *device, uint32_t timeout)`	Waits until a new conversion result is available.
`int32_t AD7124_ReadData(ad7124_device *device, int32_t* pData)`	Reads the conversion result from the device.
`uint8_t AD7124_ComputeCRC8(uint8_t* pBuf, uint8_t bufSize)`	Computes the CRC checksum for a data buffer.
`void AD7124_UpdateCRCSetting(ad7124_device *device)`	Updates the CRC settings.
`void AD7124_UpdateDevSpiSettings(ad7124_device *device)`	Updates the device SPI (Serial Peripheral Interface) interface settings.
`int32_t AD7124_Setup(ad7124_device *device, int slave_select, ad7124_st_reg *regs)`	Initializes the AD7124.

Downloads

- **AD7124 Driver:** https://github.com/analogdevicesinc/no-OS/tree/master/drivers/adc/ad7124 [https://github.com/analogdevicesinc/no-OS/tree/master/drivers/adc/ad7124]
- **Generic Platform Driver** https://github.com/analogdevicesinc/no-OS/tree/master/drivers/platform/generic [https://github.com/analogdevicesinc/no-OS/tree/master/drivers/platform/generic]

Using the API

The driver can only work together with a structure that holds the state of a device, where state means all information about the device including a copy of all register values written to the device at a certain point. This structure will henceforth be referred as an instance of a driver.
All driver functions take a handler of a driver instance as the first argument. This allows the driver to be used with multiple devices simultaneously, without the need to replicate the .c and .h files.

Before using any API (Application Programming Interface) call, an instance of the driver must first be created and then initialized using the AD7124_Setup() which has the following parameters:

- **device**: the reference of the new driver instance. A new instance can be obtained by simply declaring one:

```
struct ad7124_device my_ad7124;
```

- **slave_select**: the index of the SPI (Serial Peripheral Interface) Chip Select. It will be stored into the driver instance to be used with all SPI (Serial Peripheral Interface) calls for that particular instance.
- **regs**: must point to a register array of the device. It will be too stored into the driver and used by the driver. There is a default register array already defined in AD7124_regs.h and AD7124_regs.c files. The name of the array is ad7124_regs and can be passed here as parameter. Alternatively, a new register array can be defined by user which must be properly initialized before calling AD7124_Setup() function.

```
ad7124_st_reg my_ad7124_regs[AD7124_REG_NO];
```

A AD7124_Setup() call will also reset the part then use all register values stored in the array pointed by the **regs** parameter to configure the part (the registers flagged as “Read-only” will be skipped).

The following code snipped provides an example of driver usage:

```
#include "AD7124.h"      /* AD7124 definitions */
#include "AD7124_regs.h" /* We want to use the ad7124_regs array defined in ad7124_regs.h/.c */

void main(void)
{
    struct ad7124_device my_ad7124;                    /* A new driver instance */
    struct ad7124_device *ad7124_handler = &my_ad7124; /* A driver handle to pass around */
    enum ad7124_registers regNr;                       /* Variable to iterate through registers */
    long timeout = 1000;                               /* Number of tries before a function times out */
    long ret = 0;                                      /* Return value */
    long sample;                                       /* Stores raw value read from the ADC */


    /* Initialize AD7124 device. */
    ret = AD7124_Setup(ad7124_handler, AD7124_SLAVE_ID, (ad7124_st_reg *)&ad7124_regs);
    if (ret < 0)
    {
        /* AD7124 initialization failed, check the value of ret! */
    }
    else
    {
        /* AD7124 initialization OK */
    }

    /* Read all registers */
    for (regNr = AD7124_Status; (regNr < AD7124_REG_NO) && !(ret < 0); regNr++)
    {
        ret = AD7124_ReadRegister(ad7124_handler, &ad7124_regs[regNr]);
    }

    /* Read data from the ADC */
    ret = AD7124_WaitForConvReady(ad7124_handler, timeout);
    if (ret < 0)
    {
        /* Something went wrong, check the value of ret! */
    }

    ret = AD7124_ReadData(ad7124_handler, &sample);
    if (ret < 0)
    {
        /* Something went wrong, check the value of ret! */
    }
}
```

More information

- ask questions about the Microcontroller no-OS Drivers [https://ez.analog.com/community/linux-device-drivers/microcontroller-no-os-drivers]
- Example questions:
    - *An error occurred while fetching this feed:* http://ez.analog.com/community/feeds/allcontent/atom?community=2077 [http://ez.analog.com/community/feeds/allcontent/atom?community=2077]

01 Jun 2012 10:17 (2012-06-01T10:17:33Z)