LTC 2499 API

for the B-L072Z-LRWAN1

Microcontroller

Pietro Vitagliano

University of Sannio, Benevento, Italy

A. A. 2023/2024

Index

Introduction

This document has been created to provide a comprehensive overview of the API for the LTC 2499 analog-to-digital converter used in conjunction with the B-L072Z-LRWAN1 microcontroller. Its purpose is to elaborate in detail on the architecture, operational cycle, I2C interface, and input data format of the LTC 2499, as well as to present the associated API implementation.

This document aims to serve as an essential reference for all individuals engaged in projects based on the B-L072Z-LRWAN1 microcontroller and utilizing the LTC 2499.

LTC 2499 Overview

In this chapter, the most relevant key points of the LTC 2499 will be examined, taking into account its architecture, operating cycle, I2C interface configuration and the format of input data that can be sent to the ADC.

Architecture

The LTC 2499 ADC is an analog-to-digital converter with a 24-bit resolution, featuring a total of 16 single-ended input channels or 8 when used as differential channels. This device offers an I2C interface, enabling easy communication with other system components. What sets this converter apart is its sampling scheme, which allows for automatic elimination of errors caused by input current. This feature enables the direct acquisition of high-impedance signals from external sources and signals with maximum amplitude while maintaining high precision in DC measurements. Additionally, this device is equipped with a highly precise temperature sensor and an integrated oscillator, providing it with remarkable versatility in various applications.

Operational Cycle of the Converter

The LTC 2499 operates following a specific operational cycle divided into four distinct states. The cycle begins with the conversion phase, which is automatically executed when the device is powered on. Once completed, the device enters an idle state, significantly reducing power consumption. During this phase, the conversion result is held in a shift register. It's important to note that during the latency-free conversion, the device does not respond to any external requests. The output register is 32 bits long, with the most significant 24 bits representing the conversion result. Data transmission is synchronized by the serial clock

(SCL), and data is updated on the falling edges of SCL. The operational cycle concludes with a STOP condition, marking the automatic start of a new conversion.

Input Voltage Range

The LTC 2499 ADC allows for the measurement of an input voltage in both differential and single-ended configurations within the specified range $[-0.5 \cdot V_{REF}, +0.5 \cdot V_{REF}]$.

I2C Interface

The LTC 2499 communicates through an I2C interface, supporting a master-slave connection on a single bus. This interface can achieve speeds of up to 100 kbit/s in standard mode and up to 400 kbit/s in fast mode. It's important to note that the VCC power supply should not be removed during I2C bus activity to prevent the bus lines from being taken over by internal ESD protection diodes. Each device on the I2C bus is recognized by a unique address stored in the device itself and can function as both a transmitter and a receiver, depending on the device's role.

Input Data Format

The format of the input data to the LTC2499 allows for the proper configuration of the device. The input word consists of 13 bits, distributed across two 8-bit words. Following the nomenclature provided in the datasheet, the first word, containing the bits SGL, ODD, A2, A1, and A0, is used to select the input channel, mode, and channel polarity. The second data word, comprising the bits IM, FA, FB, and SPD, determines the rejection frequency, speed mode (1x or 2x), and temperature measurement. Upon power-up, the device initiates an internal reset cycle, setting the input channel to CH0-CH1 (CH0 as positive, CH1 as negative), the rejection frequency to 50Hz/60Hz simultaneous, and an output rate of 1x (with automatic calibration enabled). The first conversion begins automatically with this default configuration.

After the conversion is complete, you can write up to two words into the device. The first input word contains three initial bits, including two preamble bits and an enable bit. Valid settings for these three bits are 000, 100, and 101, while other combinations should be avoided as they are invalid. In the case of 000 or 100 (low enable bit), the subsequent data is ignored, and the previously selected settings, including the input channel, will be used. If the first three bits are 101 (high enable bit), the following five bits select the input channel for the next conversion cycle.

The first input bit (SGL) after the sequence 101 determines whether the input selection is differential (SGL = 0) or single-ended (SGL = 1). For SGL = 0, you can select two adjacent channels to form a differential input. For SGL = 1, one of the 16 channels is chosen as the positive input, while the negative input will be ground. The first four least significant bits (ODD, A2, A1, A0) define the selected channel(s) and polarity (for differential input).

After writing the first word into the device, a second word can be sent, in order to set other options. The first bit of the second word enables these new options (EN2). If this bit is 0, the next conversion will be executed using the previously selected configuration.
Furthermore, the settings configurable through the second word are optional. Thus, if they are not needed, it is possible to provide a sequence of 8 low bits as the second word, in order to ensure that EN2 remains low, allowing the previous settings to continue to be used.

LTC 2499 Configuration

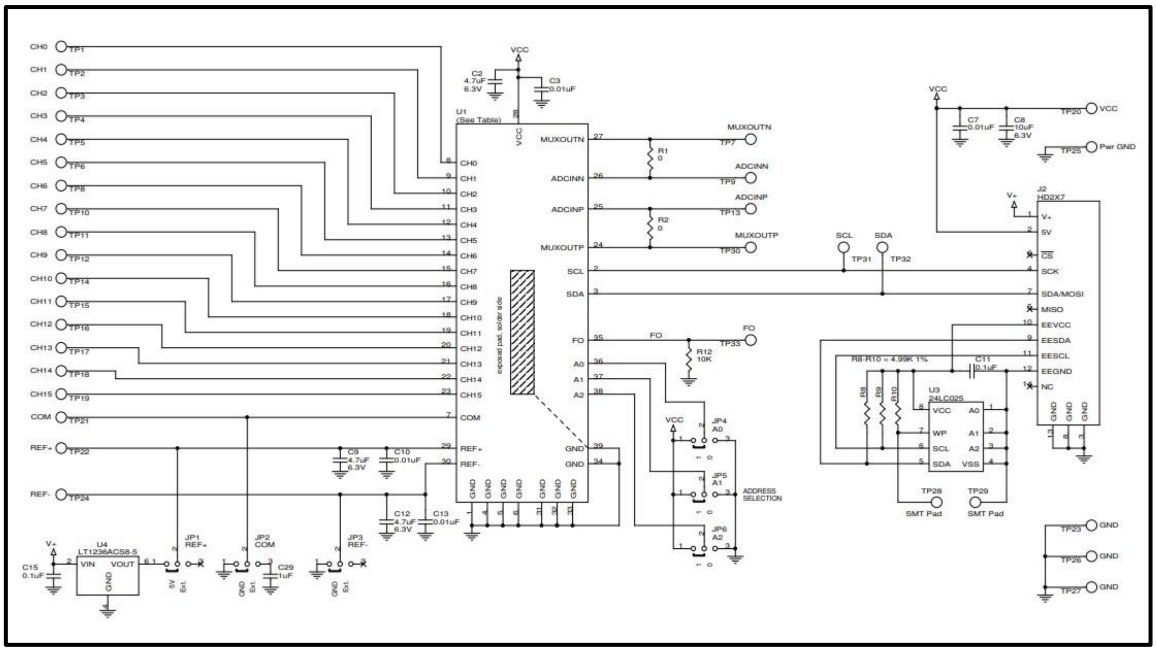

Figure 1: Diagram of the LTC 2499 pins.

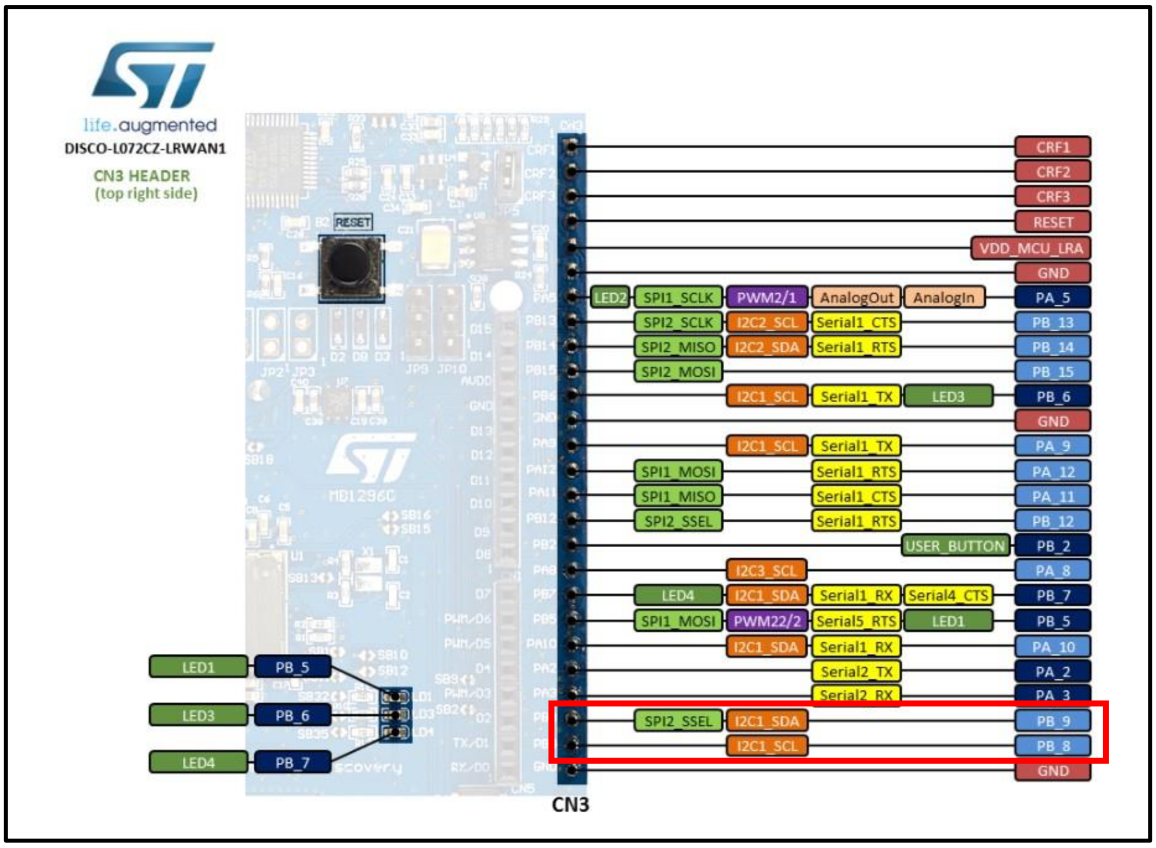

Figure 2: Pins utilized to exploit the type 1 I2C interface on the microcontroller.

To configure the LTC 2499, it is necessary to connect it to the microcontroller and provide it with power and ground.

This was achieved by utilizing the following pins of the ADC (Figure 1):

- Pin 2 for 5 V power supply
- Pin 13, 8 or 3 for ground
- Pin 4 for the data line
- Pin 7 for the clock

Specifically, the data line and clock line of the ADC were connected to the PB_9 and PB_8 pins of the microcontroller (Figure 2), as these pins enables the use of the I2C Type 1 interface, upon which the API is built.

API Implementation

Now, the implementation of the API for the LTC 2499 will be presented. It consists of two files: "ltc_2499.h" and "ltc_2499.c".

Header File "ltc_2499.h"

The "ltc_2499.h" file contains all the constants, enumerations, and function definitions that make up the API. The key elements of this file include:

- The maximum time interval within which read and write operations must be completed.
- The duration of the delay used at each step of a wait cycle.
- The number of ADC levels as 2^{24}, since it has a 24-bit resolution.
- The reference voltage provided to power the ADC.
- Enumerations for enable bits, ADC channel mode, ADC channel polarity, ADC channel selection, and ADC status.

Source File "ltc_2499.c"

The "ltc_2499.c" file consists of the implementation of the API, which is composed of the following functions:

- **ADC_Init**: Initializes the ADC with the necessary parameters, including the 7-bit I2C address, channel mode, channel polarity, and channel selection. It is required to enable read operations.
- **ADC_DeInit**: This function deinitializes the ADC by clearing its registers and resetting its configuration.
- **ADC_Write**: Writes a data buffer to the ADC's input register, 16 bits long and divided into 2 bytes, with the former being mandatory and the latter being optional. Since the

bits written to the input register are used to configure the ADC, this function is used within ADC_Init, but it can also be used to modify ADC settings at runtime after the initial setup.

- **ADC_Read**: Reads a data buffer from the ADC's output register, 32 bits long and divided into 4 bytes. The data is stored in the buffer provided as a parameter but needs to be decoded before it can be used.
- **ADC_Decode_Voltage**: Decodes the data in the ADC's output register into a voltage, represented as a 32-bit floating-point number.