

2834x External ADCs on XINTF Example Project

Contents

1	Device Support.....	1
2	Introduction.....	2
	2.1 Requirements.....	2
3	Getting Started.....	2
	3.1 Hardware Setup	2
	3.2 Software Setup.....	2
4	Example Project Overview	3
5	Design Considerations	5
	5.1 System Clock	5
	5.2 GPIOs	5
	5.3 ePWM	5
	5.4 XINTF	6
	5.5 ADS7865 ADC	6
	5.6 Timing	6
	5.6.1 XINTF Minimum Timing Requirements	6
	5.6.2 ADS7865 Timing Requirements.....	7
6	Additional Documentation.....	9

Figures

Figure 1.	Graph Settings.....	4
Figure 2.	ADC Output.....	4
Figure 3.	ADS7865 Timing Diagram.....	8

Tables

Table 1.	ADS7865 Timing Requirements	8
Table 2.	XINTF Timing	8

1 Device Support

This example project was designed to run on the Delfino C2834x DIM100 controlCARD, which features two ADS7865 ADCs connected to its external interface. It is important that the DIM100 and not the DIM168 controlCARD is used because the DIM168 controlCARD does not have ADCs connected to its external interface (XINTF). However, the example project can serve as a guide for interfacing external ADCs with the XINTF for any device that features an XINTF (C2823x, C2833x, C2834x or C281x).

2 Introduction

This example project demonstrates how to control the two external ADS7865 ADCs on the Delfino C2834x DIM100 controlCARD via the external interface (XINTF). The project also serves as a general guide for interfacing external ADCs with a C2834x device using the XINTF. Since C2823x, C2833x, and C281x devices have similar external memory interfaces, this example project provides a good reference for using an external ADC with these devices as well.

This document provides a guide to the example software, explaining the reasoning behind design choices so users can have a better grasp of how to use the external ADCs on the controlCARD as well as how to interface external ADCs with a Delfino device in custom designs.

2.1 Requirements

- It is assumed that the reader already has a 2834x controlCARD Experimenter's Kit set up and connected to a host with Code Composer Studio installed. The user should have a basic understanding of how to use Code Composer Studio to download code through JTAG and perform basic debug operations.

- The user must have controlSUITE installed with support for C2834x devices.

The project uses header files and peripheral example files located in the DSP2834x_headers and DSP2834x_common folders, respectively, located at: <controlSUITE root>\device_support\c2834x\<version>.

The project folder "adc_on_xintf" is located at:

<controlSUITE_root>\device_support\c2834x\<version>\DSP2834x_examples_ccsv4.

This project folder contains the ccsv4 project and source files.

3 Getting Started

This section describes how to set up the ADC on XINTF example project.

3.1 Hardware Setup

1. Place the controlCARD in an Experimenter's Kit docking station.
2. Connect these pins on the docking station with wires:
 - a. GPIO 01 and ADC A0
 - b. GPIO 02 and ADC B0
3. Connect the docking station to a host with Code Composer Studio 4 installed.

3.2 Software Setup

1. Open the example project from controlSUITE or use CCSv4 to import the project into the workspace.

Note: Do NOT check the "copy projects to workspace" option.

2. Ensure that the linker command file matches the device and link the correct one to the project if necessary.
3. Create or select the target configuration file appropriate for the device.
4. Make sure that the appropriate device is selected in DSP2834x_Device.h
5. Check that DSP2834x_Examples.h defines DSP28_DIVSEL = 2 and DSP28_PLLCR = 19.
This ensures 200MHz SYSCLKOUT, which is required to run the example code as is – this is further explained in section 5.1.
6. Build and load the code, and run the example, setting breakpoints and performing other debug functions as desired.

4 Example Project Overview

The example project configures the XINTF to control two ADS7865 ADCs, one connected to XINTF zone 0 and the other connected to zone 7. Each ADC is configured differently in order to demonstrate how to implement the different functionalities possible with the ADS7865. To use only one of the ADCs, change the values defined for either CE0 or CE7 to 0 to build the project without that ADC.

Both ADCs are provided with a square wave input generated by a GPIO pin that is toggled by CPU Timer0. This provides a waveform that is quickly and easily recognizable when graphing the output of the ADCs for testing. Other signals with voltages no more than 3.3V could also be provided to the ADCs for testing.

ADC1 provides the simpler example: conversions are manually triggered by pulling GPIO56 low, and the ADS7865 is configured to only perform conversions on a single channel. ADC2 conversions are triggered by the ePWM module, which outputs an external start-of-conversion signal (EXTSOC1B) to the CONVST line on the ADC. Further, ADC2 is configured to perform conversions over a sequence of channels.

Data from channel A0+ on each ADC is stored in the arrays ADC1Data and ADC2Data. A good way to see the output of the ADC is to plot this data on a graph in CCS. After starting the Debugger go to Tools > Graph > Single Time to open up a graph window. Set the parameters according to Figure 1 below – Figure 2 shows the resulting graph.

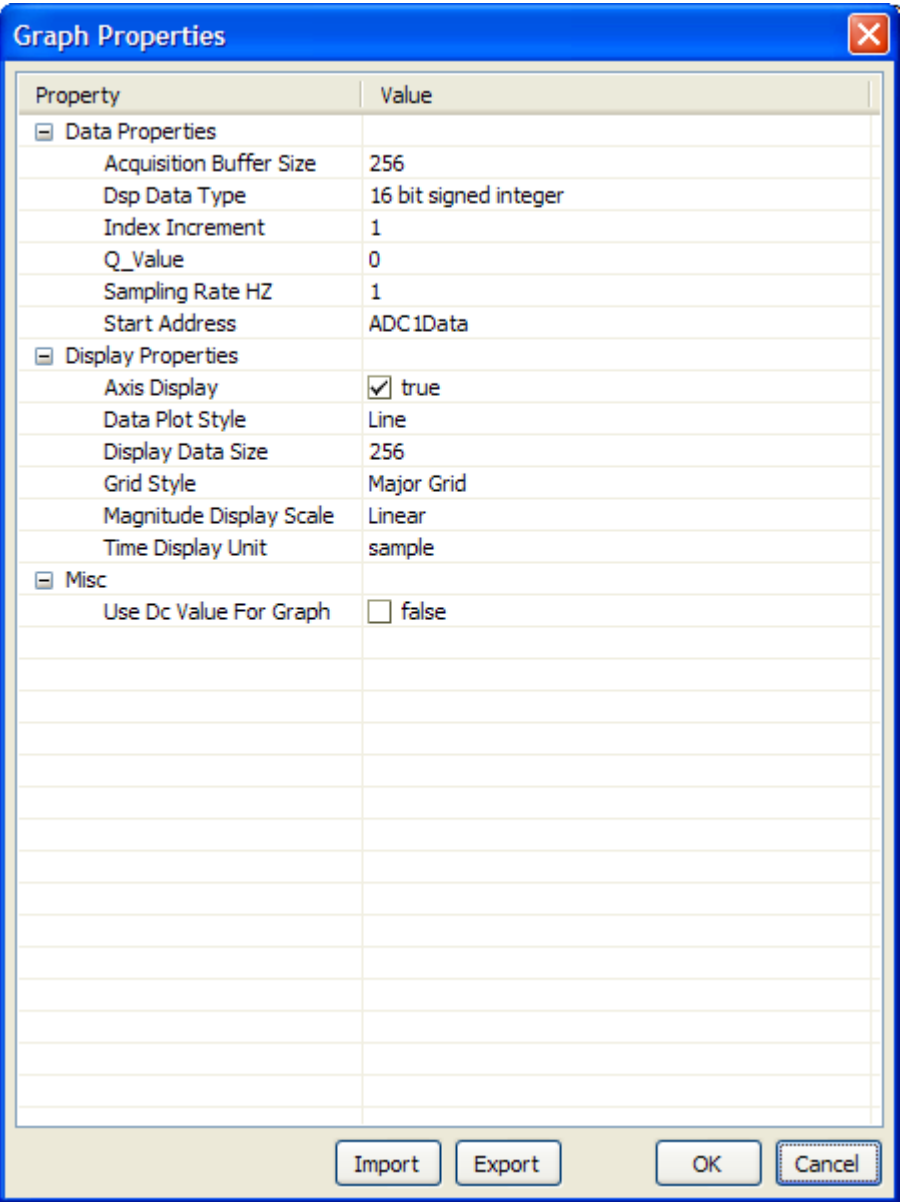


Figure 1. Graph Settings

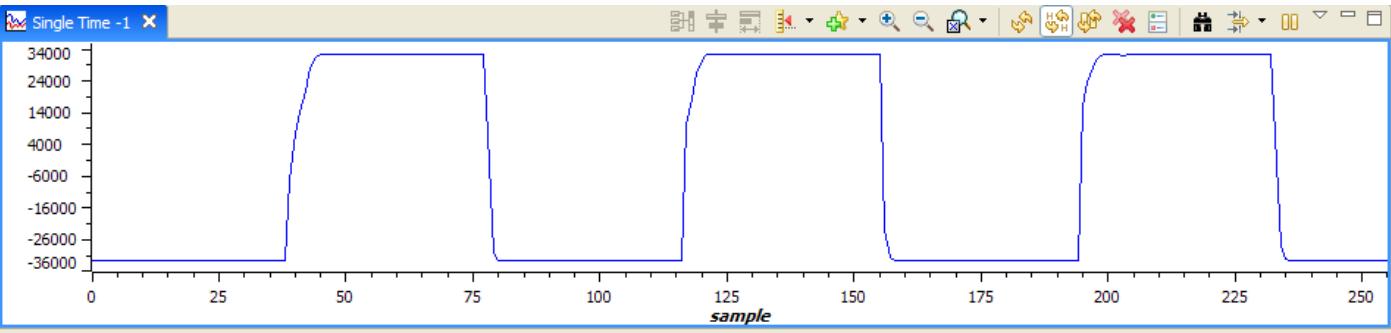


Figure 2. ADC Output

5 Design Considerations

This section explains the design considerations in the example project, and how the software should be modified for use in custom applications.

5.1 System Clock

All timing values in the example project have been calibrated for 200 MHz operation so that they will work with any 2834x device. (DSP28_PLLCR = 19, DSP28_DIVSEL = 2)

For devices capable of 300MHz interfacing with the ADS7865: The maximum frequency on the CLOCK line to ADS7865 is 32 MHz – therefore XCLKOUT cannot be greater than 32MHz. Since the smallest possible value for XCLKOUT is SYSCLKOUT/8, to use an ADS7865 the maximum allowable SYSCLKOUT possible is 250 MHz (DSP28_PLLCR = 24, DSP28_DIVSEL = 2). However, using a different SYSCLKOUT requires different timing values when configuring the external interface (see section 5.6).

5.2 GPIOs

The function GPIO_init() configures all of the GPIOs for the example project. All of the XINTF pins are configured for XINTF operation. For the ADS7865 two additional signals are needed – BUSY, and CONVST. Referring to the C2834x controlCARD schematic, the BUSY line is connected to GPIO34 – this line indicates to the C2834x that the ADC is performing a conversion. Again referring to the schematic, the CONVST signal is connected to an OR-gate over the EXTSOCs. The EXTSOCs are controlled by the ePWM module (see section 5.4).

However, GPIO56 (for ADC1) and GPIO57 (for ADC2) are also OR-ed with the EXTSOCs, allowing for manual CONVST triggering. The OR-gate allows CONVST to be selected as either manual triggered or EXTSOC triggered by initializing GPIO56/57 output to 0 or 1, respectively. Since CONVST is active low, if these GPIOs are set high the EXTSOCs cannot trigger CONVST, only manual triggering is possible – this is because the OR-gate will always output 1. However, if these GPIOs are pulled low, the output of the OR-gate will match the values of the EXTSOCs, allowing for CONVST triggering from the ePWM module.

Finally, GPIO1 and GPIO2 are configured as GPIO outputs to be provided as signals for the ADC to sample. These outputs are toggled by the CPU Timer0 interrupt service routine, creating a simple square wave.

5.3 ePWM

The ePWM module in this example is used to trigger the ADC conversions for ADC2. The code in EPWM_init() configures the module to produce start-of-conversion signals and to fire EXTSOC1B. The polarity of EXTSOC1B is set to inverted since CONVST is active low for the ADS7865. The frequency of the conversions can be changed by altering the value of TBPRD.

5.4 XINTF

The `XINTF_Init()` function configures the external interface for communicating with the two ADS7865 ADCs on the controlCARD. Each ADC is connected to and controlled by a different zone of the XINTF – one on zone 0 and one on zone 7. In this example, using `SYSCCLKOUT = 200MHz`, the XINTF is configured to provide `XCLKOUT` of 25MHz (`SYSCCLKOUT/8`) to the ADCs, since the ADS7865 requires a `CLOCK` signal frequency less than 32MHz. `XSIZE` is also set for 16 bits, since the ADS7865 is a 12-bit ADC. The selection of the timing parameters for XINTF accesses depends on the ADC and is further explained in section 5.6.

5.5 ADS7865 ADC

The `ADC_init()` function configures the ADS7865 ADCs for operation by writing to them using the XINTF – necessarily this function must be called after the XINTF has been configured for proper timing and the GPIOs have been configured for communication. Setting up the ADC for operation involves writing a series of command bytes to each ADC – these commands are defined in the ADS7865 data manual. Commands for ADC1 are written to address 0x4000 (XINTF zone 0) and commands for ADC2 are written to address 0x200000 (XINTF zone 7).

In this example, both ADCs are configured to use a `Vref DAC` value of 0x2A3, which corresponds to 3.3V/2 since our GPIOs are going to provide at maximum 3.3V inputs to the ADC. See the ADS7865 data manual for more information on `Vref DAC` codes.

ADC1 is configured to convert a single channel `A0+/B0+` (`A0+` is connected to pin ADC A0 on the docking station) with a single conversion start and busy for the entire sequence – since the sequence has a length of one, changing the command to have individual conversion start and busy signals has no functional difference.

ADC2 is configured to convert three channels with an individual conversion start and busy for each channel in the sequence. It also specifies the order of the channel sequence: channel `A0+/B0+` (pin B0), `A1+/B1+` (pin B1), and finally `A1-/B1-` (pin B2).

5.6 Timing

The timing of the XINTF signals must be tuned to the particular peripheral being accessed, in this case, the timing requirements of the ADS7865. To maximize efficiency, the fastest accesses without violating the worst case timing of the device should be used. However, the XINTF also has some minimum timing requirements, which can be found in the device data manual. The minimum XINTF required access times, the access times of the ADC, and the resulting register settings are found in Table 2, with the calculations for obtaining these values explained in sections 5.6.1 and 5.6.2.

5.6.1 XINTF Minimum Timing Requirements

The minimum lead, active, and trail periods for read and write accesses are pulse durations based on `XTIMCLK` cycles. In this example, `SYSCCLKOUT = 200MHz` and `XTIMCLK = SYSCCLKOUT/2 = 100MHz`. Therefore, one `XTIMCLK` cycle ($t_{c(XTIM)}$) = 10ns. Also note that `X2TIMING` is set to 0 (meaning wait-state values are not doubled). The following calculations were used to obtain the minimum XINTF timing requirements for this example:

$$LR = XRDLEAD \times t_{c(XTIM)} = 2 \times 10 \text{ ns} = 20 \text{ ns}$$

$$AR = XRDACTIVE \times t_{c(XTIM)} = 6 \times 10 \text{ ns} = 60 \text{ ns}$$

$$TR = XRDTRAIL \times t_{c(XTIM)} = 0 \times 10 \text{ ns} = 0 \text{ ns}$$

$$LW = XWRLEAD \times t_{c(XTIM)} = 3 \times 10 \text{ ns} = 30 \text{ ns}$$

$$AW = XWRACTIVE \times t_{c(XTIM)} = 1 \times 10 \text{ ns} = 10 \text{ ns}$$

$$TW = XWRTRAIL \times t_{c(XTIM)} = 3 \times 10 \text{ ns} = 30 \text{ ns}$$

Since the minimum XINTF timing requirements are defined in numbers of cycles, when a different XTIMCLK is used it is important to remember that even though the minimum register values (number of cycles) do not change, the minimum time does change (since the clock cycles are longer or shorter). For example, if SYSCLKOUT = 250MHz instead of 200MHz (with all other settings the same), the minimum XINTF settings may now be outweighed by the peripheral device timing requirements even if they weren't before.

CAUTION:

No internal device hardware is included to detect illegal XINTF timing settings, so care must be taken when setting timing requirements.

5.6.2 ADS7865 Timing Requirements

The ADS7865 read and write timing diagram is shown in Figure 3. The corresponding relevant timing values can be found in Table 1. The calculations below were used to obtain the ADS7865 timing values in Table 2:

$$LR = t_6 = 0 \text{ ns}$$

$$AR = t_{10} = 20 = 20 \text{ ns}$$

$$TR = \max(t_7, t_{11}, t_9) = \max(0, 5, 10) = 10 \text{ ns}$$

$$LW = t_6 = 0 \text{ ns}$$

$$AW = \max(t_8, t_{12}) = \max(10, 10) = 10 \text{ ns}$$

$$TW = \max(t_7, t_{13}) = \max(0, 5) = 5 \text{ ns}$$

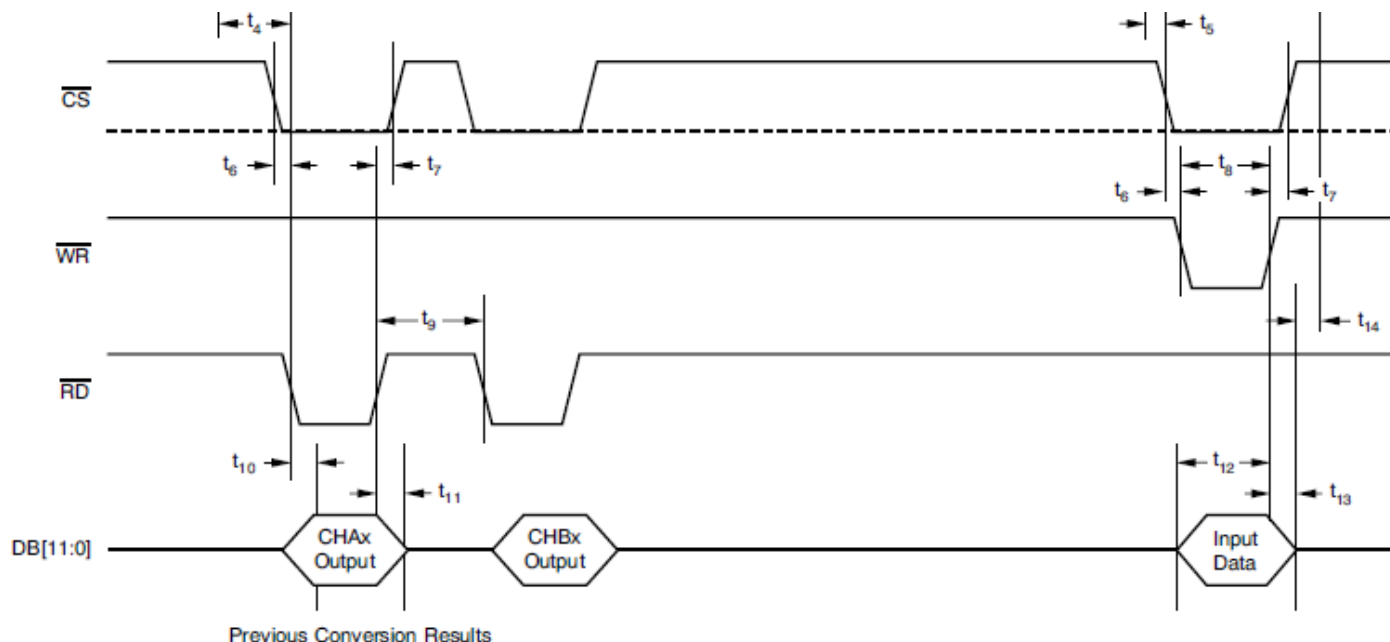


Figure 3. ADS7865 Timing Diagram

Table 1. ADS7865 Timing Requirements

	Parameter	Min	Max
t_6	CS falling edge to RD or WR falling edge setup time	0 ns	
t_7	CS rising edge to RD or WR rising edge hold time	0 ns	
t_8	WR low time	10 ns	
t_9	RD high time between two read accesses	10 ns	
t_{10}	RD falling edge to output data valid delay		20 ns
t_{11}	Output data hold time	5 ns	
t_{12}	Input data setup time	10 ns	
t_{13}	Input data hold time	5 ns	

Table 2. XINTF Timing

	Description	XINTF minimum	ADS7865	Register setting
LR	Lead period, read access	20 ns	0 ns	XRDLEAD = 2
AR	Active period, read access	60 ns	20 ns	XRDACTIVE = 6
TR	Trail period, read access	0 ns	10 ns	XRDTRAIL = 1
LW	Lead period, write access	30 ns	0 ns	XWRLEAD = 3
AW	Active period, write access	10 ns	10 ns	XWRACTIVE = 1
TW	Trail period, write access	30 ns	0 ns	XWRTRAIL = 3

Note: Highlighted boxes indicate the stricter requirement, showing which values determine the register setting for each row.

6 Additional Documentation

This guide references information about the ADS7865 ADC, the TMS320C2834x XINTF, and the Delfino C2834x DIM100 controlCARD schematic. It is strongly recommended that the user reference the following documentation for more detailed information:

1. *Dual 12-Bit, 3+3 or 2+2 Channel, Simultaneous Sampling ADC* (SBAS441B)
2. *C2834x C/C++ Header Files and Peripheral Examples Quick Start* (SPRCA74)
3. *TMS320x2834x Delfino External Interface (XINTF)* (SPRUFN4A)
4. *TMS320x2834x Delfino System Control and Interrupts* (SPRUFN1B)
5. *TMS320x2834x Delfino Enhanced Pulse Width Modulator (ePWM) Module* (SPRUZF6B)
6. *Delfino C2834x DIM100 controlCARD Schematic* - in TMDXCNCDD28343 Hardware Developer's Package (SPRC905)