Hardware Interface with the TI TMS320F28377D and DC2100A via SPI Using Code Composer Studio v6

Andrew Hoang

August 31, 2015

Abstract

The purpose of this document is to help guide the user utilize the TMS320F28377D and DC2100A via SPI protocol. The TMS320F28377D paired with the LTC6804-2 connection utilizes SPI protocol with isoSPI as middleware between the two devices. Prior to the configuration of the SPI protocol, the TMS320F28377D requires the user to configure SPI according to the LTC6804-2 datasheet specifications i.e baud rate, clock polarity, transmit size, etc. This document will help the user understand what is necessary from the user in order to use the TMS320F28377D effectively with the DC2100A board.

This document assumes the user is familiar with C code syntax and has TI Code Composer Studio version 6.0 or higher as their IDE.

1 Setup

The TMS320F28377D utilizes the SPIA GPIO pins (SPIAMOSI, SPIAMISO, SPIASCLK) located on the control card (Pins 67, 69, and 71) as well as a GPIO pin for the SPI chip select (CS). Refer to the "TMS320F28377D Control Card Pin Out" PDF file located in the reference section of this document. However, note that in this instance the code provided uses a GPIO pin as the SPI chip select (CS) pin.

PIN CONFIGURATION SETUP:
PIN 1 - 6 left to right on the DC2100A board are as follows:

- DC2100A board 1C - 3.3V Enable, 2C SPIMOSI, 3C SPIMISO, 4C SPICLK, 5C SPICS, 6C Ground
- TMS320F28377D PIN 67 - SPIMOSI, PIN 69 SPI MISO, PIN 71 SPICLK, PIN 89 SPICS (GPIO)
- Therefore: 1C with a 3.3 Voltage Source, 2C with pin 67, 3C with pin 69, 4C with pin 71, 5C with pin 89, and 6C with an electrical GND reference.

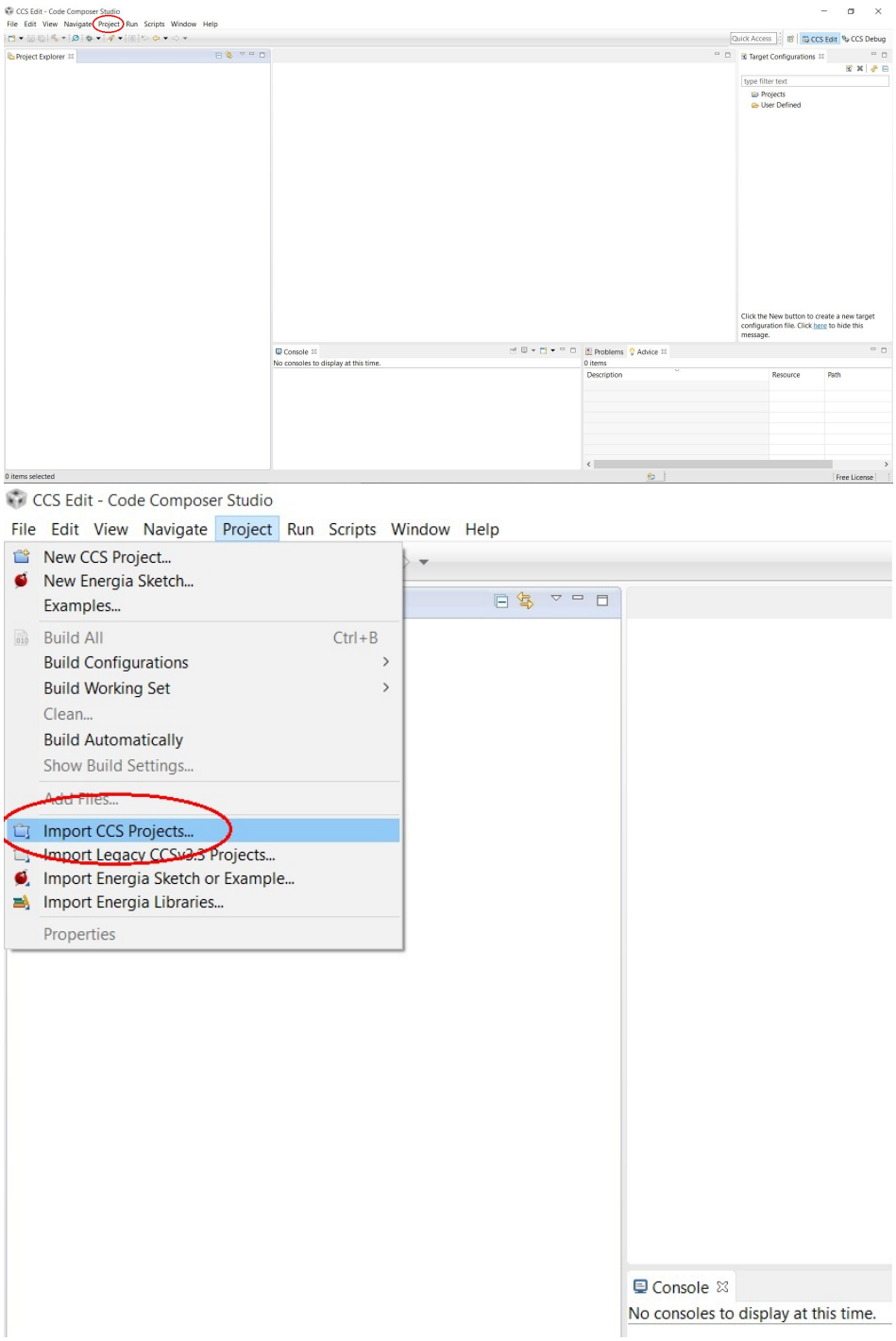
2 TI Code Composer Studio IDE configuration

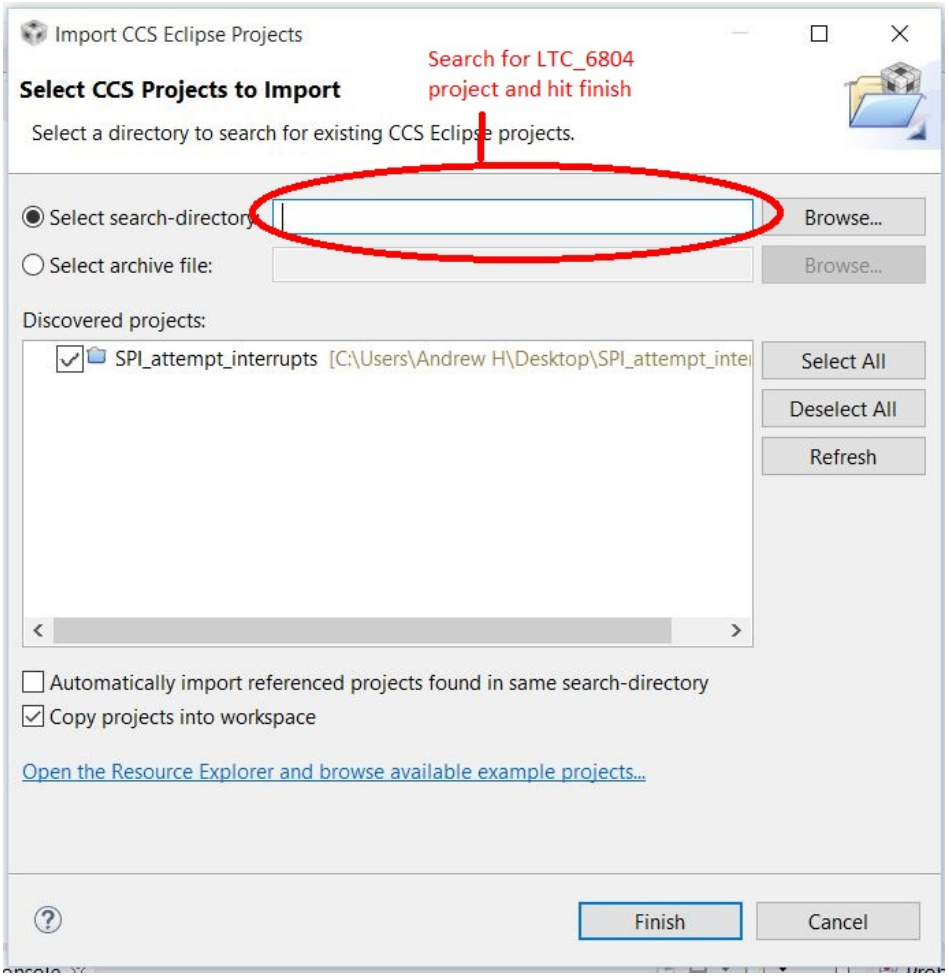
The TMS320F28377D requires the use of an IDE provided by TI called Code Composer Studio IDE. The IDE will help aid the user utilize the hardware features of the TMS320F28377D into tangible pieces. Also provided in this document contains documentation for which the user can refer to when choosing to interface with the DC2100A evaluation board as their main purpose. The code functionality is not limited to just the TMS320F28377D but to many other TI devices as long as the user configures the correct settings for each individual hardware in the Target Configurations ccxml file.

3 Procedure to use TMS320F28377D from an Empty Project

The TMS320F28377D requires the software ControlSuite provided freely on TI's webpage: ControlSUITE. The ControlSuite software includes all the necessary peripheral support for SPI, interrupt, and general interfacing for use on the TMS320F28x family boards. Once ControlSuite has been downloaded and installed, TI Code Composer Studio needs to link to various files found inside "Installation Directory/controlSUITE/device_support/F2837xD/v150/F2837xD_common/include" and "Installation Directory/controlSUITE/device_support/F2837xD/v150/F2837xD_headers/include".
In order to initialize a new Code Composer Studio project, the user would follow the depicted images below step-by-step.
Click Finish.
Congratulations, you have created your very first project within CCS!
Note, in order to load existing CCS projects, the user may use the import function found under "Projects → Import CCS Projects" and locate the CCS project for direct usage. Note that this document has already prepared a CCS project for the user to use the TMS320F28377D to DC2100A board with SPI as a peripheral interface.

4 Procedure to use TMS320F28377D from Imported Project





Note that the LTC_6804-2_SPI project has included all the necessary files required in order to immediately work with the DC2100A board from the TMS320F28377D. The only thing the user must do in order to obtain necessary function between software and TMS320F28377D is to correctly configure the proper target configuration file (included in the project as well).

5 LTC6804-2.c Function Explanations

Main functions to be utilized as follows:

- void LTC_wakeup()

  Description: The LTC_wakeup() function corresponds to the GPIO 89 found on the control card of the TMS320F28377D. It is responsible for acting as a pseudo-CS (SPI Chip Select) wire for SPI communications.

  If the user wishes to use the default SPISTE CS on the TMS320F28377D as opposed to a GPIO pin, the SPI_init() function must be reconfigured. In addition, the user still needs to submit "wake-up" pulses to the DC2100A board in order to operate the system as intended.

- void LTC_refon_set()

  Description: The LTC_refon_set() function sends 4 bytes of data through the SPI bus and instantiates the LTC6804-2 write configuration command (WRCFG). The LTC_refon_set() then sends 6 bytes of preset information. The 6 bytes of preset information is followed by 2 bytes of PEC (Packet Error Code). Refer to the LTC6804-2 WRCFG command datasheet for a more detailed analysis of all the specific commands available.

- void LTC_adc_clear()

  Description: Sends 4 bytes to the LTC6804-2 to clear any previous ADC conversion readings.

- void LTC_ADSTAT_clear()

  Description: The LTC_ADSTAT_clear() function clears any data stored in the LTC6804-2 status registers.

- void LTC_ADC_conversion()

  Description: The LTC_ADC_conversion() sends 4 bytes of data to the SPI bus to initialize the LTC6804-2 to begin a calculation of the voltage readings. The LTC adc conversion time may vary depending on how the user has setup the configuration registers of the ADC. However, the current preset conversion time has been regulated to a 6000ms buffer region time. Note that simply executing this function will not output information, but serve as a preempt to the LTC ADC readings.

- void LTC_ADSTAT_conversion()

  Description: Similar to the ADC conversion command, the ADSTAT conversion function informs the LTC to initialize ADC status register readings from it's other storage blocks.Note that simply executing this function will not output information, but serve as a preempt to the LTC ADC readings.

- void LTC_read_voltages_123()

  Description: The LTC_read_voltages_123() reads back cells 1,2 and 3 voltages found in the voltage readings register of the LTC6804-2 ADC. Note however that in order obtain the most up to date readings from the LTC, the ADC conversion function (LTC_ADC_conversion()) should be instantiated first before running this command. Voltage readings are stored under a global array as cell_voltage with voltage 1 starting at 0.

- void LTC_read_voltages_456()

  Description: The LTC_read_voltages_123() reads back cells 1,2 and 3 voltages found in the voltage readings register of the LTC6804-2 ADC. Note however that in order obtain the most up to date readings from the LTC, the ADC conversion function (LTC_ADC_conversion()) should be instantiated first before running this command. Voltage readings are stored under a global array as cell_voltage with voltage 1 starting at 0.

- void LTC_read_voltages_789()

  Description: The LTC_read_voltages_789() reads back cells 7, 8, and 9 voltages found in the voltage readings register of the LTC6804-2 ADC. Note however that in order obtain the most up to date readings from the LTC, the ADC conversion function (LTC_ADC_conversion()) should be instantiated first before running this command. Voltage readings are stored under a global array as cell_voltage with voltage 1 starting at 0.

- void LTC_read_voltages_10_11_12()

  Description: The LTC_read_voltages_10_11_12() reads back cells 10,11 and 12 voltages found in the voltage readings register of the LTC6804-2 ADC. Note however that in order obtain the most up to date readings from the LTC, the ADC conversion function (LTC_ADC_conversion()) should be instantiated first before running this command. Voltage readings are stored under a global array as cell_voltage with voltage 1 starting at 0.

- void ready_rxbuf()

  Description: The ready_rxbuf() function grabs all data off of the SPIRXBUF until there are only 8 bytes remaining. This function was designed to ensure data transmission from the buffer line is

accurate from byte to byte. This function can be used intermittently with other functions in order to obtain actual readings.

- void LTC_spi_terminate_charge_or_discharge()

  Description: Terminates any LTC3300 balancing that may be going on the DC2100A board by sending a parity bit to the SPI bus through the LTC6804-2 IC.

- void LTC_UVOV_get_flags()

  Description: This function calls the LTC6804-2 to check for any under voltage or overvoltage flags in the cell readings. The ADC flag readings are stored in the status register and can be read from by utilizing the LTC_read_UVOV_flag() command.

- void LTC_read_UVOV_flags()

  Description: This function pulls the under voltage and overvoltage flags from the status registers. Note that it does not do any preemptive readings and pulls what is found on the status registers alone. A refresh to the status register requires the use of the get_flags function command(). The cell flags are stored under a global array that may be used to read out under/over/normal flags in the main function. The array is notified as cell_flags[] and each array may contain a value of 0 (normal), 1 (undervoltage), 2 (overvoltage), or 3 (under and over voltage, due to error most likely).

- Uint16 LTC_pec_calc(Uint16 *data, Uint16 length)

  Description: Initializes a pec calculation for the TMS320F28377D and LTC6804-2. This code is to be only used internally with the LTC6804-2_functions.c source code but may be externalized if necessary. The function returns a 4 hexadecimal value that can be split into 2 bytes as PEC values. The PEC values are calculated by inputting an array and size of the array as arguments into the function.

- Uint16 ltc3300_crc_calc(Uint16 nibble1, Uint16 nibble2, Uint16 nibble3)

  Description: This function does a nibble PEC calculation designed for the LTC3300. Note however that all communications are to be completed on the SPI bus line through the LTC6804-2 before targeting the LTC3300 channel directly.

- void LTC3300_execute_balance(Uint16 cell1, Uint16 cell2, Uint16 cell3, Uint16 cell4, Uint16 cell5, Uint16 cell6)

  Description: This function executes any write commands previously written to it (see write balance command description) and runs the LTC3300 active balancing protocol.

- void LTC3300_write_balance(Uint16 cell1, Uint16 cell2, Uint16 cell3, Uint16 cell4, Uint16 cell5, Uint16 cell6)

  Description: The function takes in 6 arguments and depending on the value of the 6 arguments, the user may charge, discharge or do nothing to each of the 6 cells the LTC3300 controls. This command only instantiates a write command to the LTC3300 and does not actually execute an active balance protocol.

6 main.c Function descriptions

- cpu_timer0_isr
- cpu_timer1_isr

- cpu_timer2_isr

  Description: These 3 CPU timers have a clock rate of 200MHZ and may be configured depending on the user preference and specifications required. The configurations for these CPU timers can be found in the main() function where all code instantiation is to be started. Furthermore, most functions should be incorporated into the CPU timer ISRs as they will be hosts for all function/methods. Currently, CPU timer 1 and 3 has been configured to trigger an interrupt every second and CPU timer 2 has been configured to trigger every 10 seconds. Also, the CPU timer priority corresponds to the lowest CPU timer (ie. CPU timer 0 will always try to preempt CPU timer 1).

- output_high(void)

  Description: Instantiates the GPIO pin to enact as an SPI chip select pin.

- spi_fifo(void)

  Description: This function contains all the necessary SPI FIFO functions in order for the TMS320F28377D to interact with the DC2100A board. The user may configure the settings found in this function, but should be diligent in what is changed so that SPI communications between devices remains functional

- spi_init(void)

  Description: This function instantiates all the necessary parameters for the SPI protocol to work in conjunction with the DC2100A board. The function contains a slew of options that the user may change and configure. Also, the user should be careful changing any of the configurations in case the device is no longer able to communicate with the DC2100A board.

7 Additional Information

It is worth noting that the TMS320F28377D has been configured to compute ADC voltage readings and balance commands consistently so that the Watch Dog Timer (WDT) does not expire on the LTC6804-2. The LTC6804-2 has a WDT which expires every 2 seconds while the LTC3300 has a WDT that expires every 1.5 seconds. As a result, if the user wishes to keep the two IC's in wakeup mode, the user must consistently issue commands to the LTC6804-2/LTC3300 every second.

For the purpose of the LTC6804-2 and LTC3300 functionality, PEC calculations have already been implemented as separate functions into the code. If the user wishes to take advantage of the PEC calculator, it may be found in the source code as LTC_pec_calc() and ltc3300_crc_calc(). All the user has to do in order to utilize the PEC calculations is to create an array of values to be entered followed by the size of the array.

8 References

 TMS320F28377D Technical Reference Document

 TMS320F28377D Control Card Pin Out

 TMS320F28377D Datasheet

 TMS320C28x CPU and Instruction Set Reference Guide

 DC2100A Software User's Guide

 LTC6804-2 Datasheet

 LTC3300-1 Datasheet

9 Appendix

Oscilloscope images of working SPI protocol with TMS320F28377D and DC2100A boards: