Application Report
SNAA213–October 2013

Interfacing LDC1000 with the MSP430 LaunchPad

Charles Cheung

ABSTRACT

The LDC1000 is a programmable inductance-to-digital converter for use in inductance-sensing applications. It offers a complete signal path solution between an LC tank and a microcontroller by providing proximity and inductance data. The MSP430 is an ideal microcontroller solution for low-cost, low-power precision sensor applications because it consumes very little power. This library provides functions to facilitate the interfacing of any MSP430 device to a LDC1000. Any SPI-capable device within the MSP430 family utilizing USART, USI, or USCI can be used with this library, made possible by hardware abstraction. This document provides descriptive information and instructions for using the library either for demonstration purposes or for project implementation. This is the recommended starting point for developing software for the LDC1000 and MSP430 combination. The software examples have been developed for the MSP430G2xx3/2/1 series and MSP430F20x2/3 series compatible with the MSP-EXP430G2 LaunchPad Value Line [4], the MSP430F55xx series compatible with the MSP-EXP430F5529LP LaunchPad [3], and the MSP430F261x/241x and MSP430AFE2xx series, but can easily be ported to another hardware platform.

Contents

1 Introduction

This application report describes different ways to interface and use the TI LDC1000 device with an MSP430. The accompanying software contains a function library allowing quick prototyping of LDC1000 setup and control. The software provided in this library is a starting point for developers wanting to get the most out of MSP430 and LDC1000 devices.

2 Purpose and Scope

To aid in interfacing these devices, TI has produced a code library that significantly reduces the need to write low-level interface functions. It provides a boost in the development of an MSP430/LDC1000-based product, saving time and allowing quick progression to the application-specific aspects of the project. This library is designed to be used with any SPI-capable MSP430 device. Since a SPI master can be implemented using one of many peripherals within the MSP430 family, and since the peripherals available may differ by device and application, library calls are provided for each of these interfaces. The chosen interface is selected by assigning a value to a system variable, which causes the compiler to conditionally include the appropriate function calls. As such, application code utilizing the library remains portable between various MSP430 devices, with minimal modification required.

Several complete example application projects are provided with the library. The purpose of these projects is to demonstrate use of the library. It is not intended as a comprehensive guide to using the LDC1000, and it does not make use of all the features of these devices. However, it does make use of all of the register access functions the library provides.

3 Quick Start

All of the examples are written for Code Composer Studio IDE [5] and use less than 16kB of flash memory. To quickly get started:

1. Download and install Code Composer Studio version 5.5.0+.
2. Launch Code Composer Studio.
3. In the top menu, select “Project > Import Existing CCS Eclipse Project”
4. Select the LDC1000 code library archive file: http://www.ti.com/litv/zip/snac059
5. Select all projects to import from the newly generated list. It is essential to import all projects since there are shared files.
6. Press “Finish.”
7. Connect a LaunchPad to the Host computer. This LaunchPad is interfaced to the LDC1000+Sensor (see Section 9).
8. In the Project Explorer window, select an imported project.
9. In the top menu, select “Run > Debug.” The program will compile and the output will be flashed to the LaunchPad target.

4 File Organization

The library files can be found in individual projects under folder library. The projects are named LDC1000_[MSP430 Variant]_[Project Type]. The workspace directory tree is shown in Figure 1.

```
/LDC1000_G2xx3_Threshold
      /library
/LDC1000_G2xx3_DRDY
      /library
/LDC1000_G2xx2_Threshold
...
```

Figure 1. Workspace Directory Tree

The library has been implemented with modular hardware abstraction. There are header files specific to each of the hardware components: LDC1000, LDC1000 EVM (board), and SPI. Table 1 shows the hardware definition header files. Table 2 shows the library code files and headers. Table 3 shows the demonstration applications that accompany the library.

Table 1. Hardware Definition Files

Filename	Library	Description
LDC1000_cmd.h	LDC1000EVM	Definitions specific to the LDC1000 device, including register addresses and bit masks.
LDC1000_evm.h	LDC1000EVM	Definitions specific to the board being used. Defines the pin interface between the board and LDC1000 device(s).
spi_1p1_port.h	SPI	Definitions specific to the selected SPI peripheral.
spi_1p1.h	SPI	Definitions specific to the SPI protocol. Defines the SPI baud rate, pin configuration, peripheral to select (USART, USI, USCI), and timeout.

Table 2. Library Code

Filename	Library	Description
LDC1000_evm.c	LDC1000EVM	Functions for configuring and calibrating LDC1000.
spi_1p1.c	SPI	Functions for reading and writing registers to the LDC1000 using the SPI protocol.

Table 3. Demo Projects Included with the Library

Project Type	Description
Threshold	Demonstrates comparator threshold interrupt capability.
DRDY	Demonstrates reading of proximity and inductance on data ready interrupt.
XTAL	Demonstrates crystal-driven timed data polling.
LowSpeed	Demonstrates low speed, low power data ready sampling.

NOTE: The register settings values can be obtained from the “Register Settings” file saved from the LDC1000 EVM GUI.

5 Library Description

The included SPI library should be used as a reference for test and evaluation. It includes support for USART, USI, and USCI peripherals in 3-wire mode, which is technically a 4-wire SPI without the Chip Select (CS) pin. For the CS pin: setup, select, and deselect functions are user-defined. The separation of CS from the SPI library allows support for pin-limited and multi-coil applications.

An example of EVM functionality is included in the LDC1000EVM library, which interfaces the SPI library to setup and configure the default EVM sensor coil for a user-defined range of Rp Min values and Rp Max values. A LDC1000 command pre-processor definition file is included for register mapping.

Both the LDC1000EVM and SPI libraries are written to be minimal as to facilitate code comprehension and usability.

6 Configuring the SPI Library

6.1 Configuring SPI Port

To configure the library for a specific SPI port connected to the LDC1000:

1. Defines in the header file spi_1p1.h should be edited for SDO (MISO/SOMI), SDI (MOSI/SIMO), and SCLK pin definitions.
2. Un-comment the port to select.
   - For USCI: SPI_USCIA0/1/2/3 or SPI_USCIB0/1/2/3
   - For USART: SPI_USART0/1
   - For USI: SPI_USI
3. Set the SCLK source.
   - For USCI: SPI_UCSSEL
   - For USART: SPI_USARTSEL
   - For USI: SPI_USISEL
4. Set the SCLK divider.
   - For USCI and USART: SPI_BRLOVAL and SPI_BRHIVAL
   - For USI: SPI_USIDIV
   - The LDC1000 max SPI clock speed is 4 MHz.
5. Optionally enable interrupts (only for x5xx and x6xx devices).
   - #define SPI_ENABLE_INTERRUPTS
6. Optionally set the timeout in clock cycles, SPI_TIMEOUT_IN_CYCLES

Whether the interrupt is enabled or not, the SPI Library is blocking, since there is a wait-for-completion routine in the static function **spi_exec()**. To have true interrupt, non-blocking functionality, please refer to the example project LDC1000_F5529LP_DRDY. To prepare this SPI Library for commercial deployment, it is suggested to disable wait-for-completion routines in spi_1p1.c, and to configure a separate timer peripheral as a SPI watchdog.

In cases of limited flash memory, if non-interrupt SPI is used, SPI macros SPI_RXRDY(), SPI_TXRDY(), SPI_IDLE() can be written as functions. Moreover, if programming for a single coil application, the SPI_1P1_CS_Pin structure can be abandoned in favor of hard-coding Chip Select routines into **spi_exec()**, SPI macros, and, if used, the interrupt state machine.

6.2 Configuring Chip Select

Chip Select (CS) is configured using a SPI_1P1_CS_Pin structure, which contains pointers to three functions defined by the user: chip setup, select, and deselect.

Chip setup is an initialization function that is called by **spi_cs_setup()**. To configure the library:

1. Chip setup, select, and deselect functions must be defined per CS pin
2. A **SPI_1P1_CS_Pin** structure must be defined per CS pin
3. An array of all **SPI_1P1_CS_Pin** structures must be passed to **spi_cs_setup().**
4. Optionally set the default CS structure with **spi_cs_setDefault().** This is recommended for single coil applications. If the default CS structure is not set, calls to read or write data will require passing in a valid CS structure instead of NULL.

The example LDC1000 EVM interface library LDC1000_evm.c shows how to configure CS as default for a single coil application.

7 Configuring the LDC1000EVM Library

Example projects assume that the coil interfaced is the EVM coil. This fixed constraint lowers the complexity of the code so users can focus on the LDC1000 EVM library's demonstration of LDC1000 initialization, calibration, and test. To configure the LDC1000EVM Library:

1. Define Chip Select Pin, Output LDCLK Pin, LED Pin mappings
2. Define calibration ranges for Rp Min and Rp Max
3. Define final test range values for proximity and frequency counter data

If proximity data is sufficient, then LDCLK can be disabled and the LDCLK pin configuration step in **evm_init()** commented out. The calibration ranges should be centered on the calculated Rp Min and Rp Max values, which can be derived from instructions in the quick start guide or from formulas in the datasheet [2]. For a specific coil, calling **evm_test()** is not necessary if optimal Rp Min and Rp Max values are known. In this case, defining TEST_RPMAX_INIT and TEST_RPMIN_INIT are sufficient since all registers are programmed in **evm_init()**.

8 Example Projects

8.1 Threshold: Comparator Interrupt

The threshold examples demonstrate comparator interrupt functionality on the LDC1000 INT pin. The application initializes the LDC1000, programming the standard EVM high and low thresholds. Next, the microcontroller sleeps. Edge transitions of the interrupt pin toggle on and off a LED.

8.2 DRDY: Data Ready Interrupt

The data ready examples demonstrate data ready (DRDY) interrupt functionality on the LDC1000 INT pin. Falling edges of the interrupt pin wakeup the microcontroller from sleep. When the microcontroller is awake, proximity and frequency counter data is read and sent out on a serial interface. The microcontroller then returns to sleep.

NOTE: For the MSP430F5529LP, the serial interface is implemented as USB-CDC. The user must manually install the included CDC driver located in the project directory. The MSP430F5529LP example supports time-stamping.

8.3 XTAL: Crystal-Driven Polling

The crystal example demonstrates timed polling of LDC1000 data. A timer clocked by an external crystal wakes up the microcontroller. When the microcontroller is awake, proximity and frequency counter data is read and sent out on a serial interface. The microcontroller then returns to sleep.

8.4 LowSpeed: Low Speed Polling

The low speed example demonstrates low-speed, low-power polling. A timer is configured to wake up the microcontroller at a low frequency. When the microcontroller is awake, LDCLK is started, LDC1000 is set to an active state, and a sample of proximity and frequency counter data is taken after the first data ready interrupt pulse. This sample is time-stamped and sent out on a serial interface. Before returning to sleep, the microcontroller sets the LDC1000 to standby and stops the LDCLK.

The example polls at every 250ms, but the interval can be adjusted to be faster. The speed of sampling depends heavily on the output data rate (ODR). If the speed of the serial interface is infinite, then the polling interval is dominated by 4/ODR, which is roughly the time it takes for the first data sample to be ready.

NOTE: For the MSP430F5529LP, the serial interface is implemented as USB-CDC. The user must manually install the included CDC driver located in the project directory.

9 EVM Interface

All projects are written for the LDC1000+Sensor EVM interface, which can be obtained by snapping off the LDC1000+Sensor from the LDC1000 EVM.

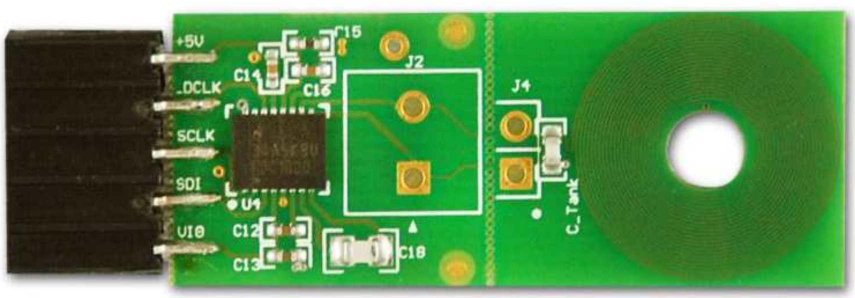

Figure 2. LDC1000+Sensor: Example with Soldered Headers

Table 4. Pinout for LDC1000+Sensor

LDC1000+Sensor Pin Name	Description
+5 V	Input Power
LDCLK	TBCLK (external clock)
SCLK	SPI Clock
SDI	SPI Data In (MOSI / SIMO)
VIO	IO Voltage
GND	Ground
INT	Interrupt
CSB	Chip Select Bar (~SS)
SDO	SPI Data Out (MISO / SOMI)

10 Interfacing the MSP-EXP430F5529 LaunchPad

The MSP-EXP430F5529 Launchpad is recommended for applications where high samplingrate, high memory, and/or high processing speed are required. This is suitable for high-speed counting, multi-coil, and high precision applications.

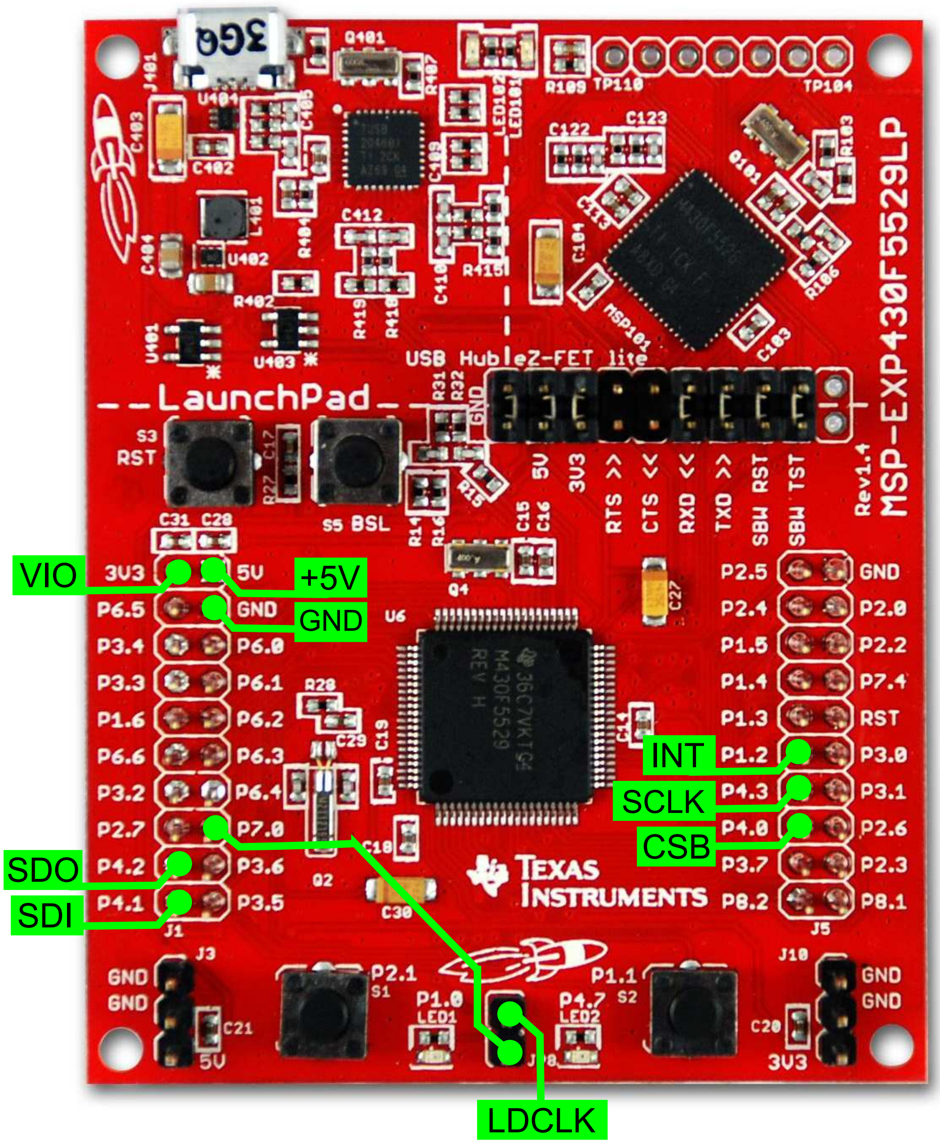

Figure 3. LDC1000+Sensor and MSP-EXP430F5529 LaunchPad Interface Connections for F5529LP Examples

11 Interfacing the MSP-EXP430G2 Value Series LaunchPad

The MSP-EXP430G2 Value Series Launchpad is recommended for applications where low sampling-rate and very minimal processing is required. This is ideal for prototyping low-speed counting, inductive switch, or inductive slider applications.

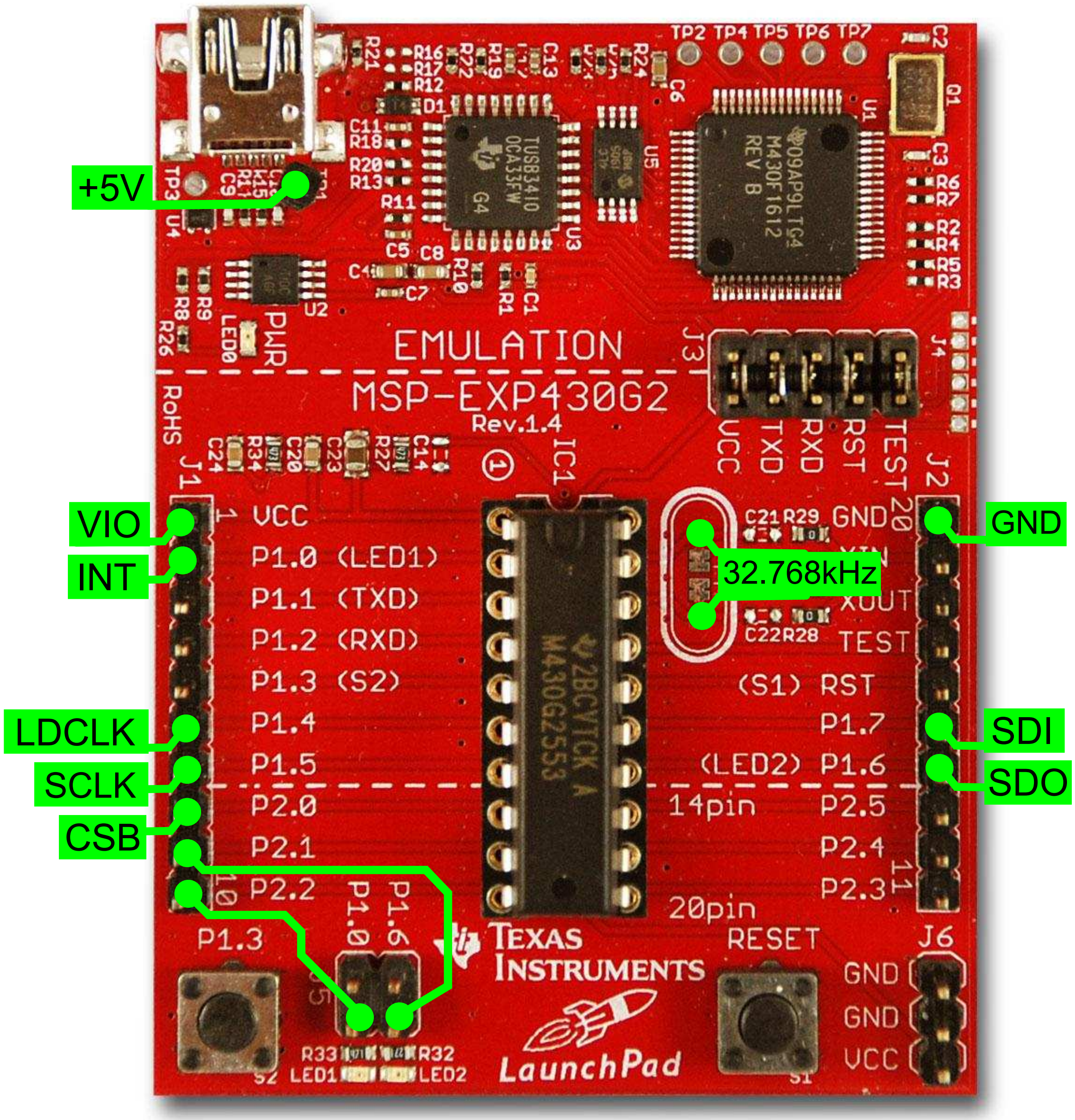

Figure 4. LDC1000+Sensor and MSP-EXP430G2 LaunchPad Interface Connections for G2xx3 Examples

12 References

1. LDC1000EVM Quick Start Guide (SLYW022)
2. LDC1000 Inductance to Digital Converter (LDC1000)
3. MSP-EXP430F5529LP LaunchPad Evaluation Kit, http://www.ti.com/tool/msp-exp430f5529lp
4. MSP-EXP430G2 LaunchPad Experimenter Board, http://www.ti.com/lit/ug/slau318d/slau318d.pdf
5. Code Composer Studio IDE, http://www.ti.com/tool/ccstudio

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265