Implementing SPI Communication Between MSP430™G2452 and LTC2382-16 ADC

Enwei Gu
Nov. 12, 2011

Keywords

•MCU
•ADC
•MSP430-G2452
•LTC2382-16
•16-bits
•SPI

1 Abstract

This document describes and shows how to implement SPI Communication Between MCU and ADC. A MSP430™G2452 ultralow-power MCU is used to communicate with the LTC2382-16 ADC. ADC converts an external analog signal to digital and sends it to MCU by SPI Communication mode. Details on MCU, ADC and SPI are discussed in this note. Though an MSP420G2452 is used for this application note, any MSP430 device could be used to implement this application.

Table of Contents

2 Abbreviations

MCU	Microcontroller Unit
ADC	Analog to Digital Converter
SPI	Serial Peripheral Interface
USI	Universal Serial Interface
SDO	Serial Data Out
CS	Chip Select
MISO	Master Input, Slave Output (output from slave)
MOSI	Master Output, Slave Input (output from master)
SS	Slave Selection
SCK	Serial Clock (output from master)
SDO	Serial Data Out

3 SPI in MSP430G2452

3.1 Introduction to SPI

The Serial Peripheral Interface (SPI) bus is a synchronous serial data link that was introduced by Motorola and is the simplest communication protocol in general use. Devices communicate in master and slave mode where the master device initiates the data frame and provides clock signals to slaves. In its full form SPI requires four wires and transmits data simultaneously in both directions between two devices.

3.2 Operation of SPI

The SPI can operate with a single master device and with one or more slave devices. However, the most basic example of SPI communication is Single Master and Single Slave (Figure 1). In this example, the SS pin acts as a chief switch. The slave will be activated and prepared to work, when SS pin was low (or high). And master provides clock signals (SCLK) to the slave in order to synchronous the data transmission (MOSI and MISO).

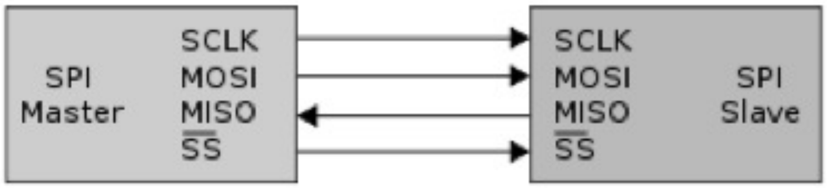


Figure 1: single master and single slave

3.3 SPI data transmission

Two shift registers can be used to represent the concept of SPI data transmission (Figure 2). Each device places a new bit on its output from the most significant bit of the shift register when the clock has negative edge and reads its input into the lsb of the shift register on a positive edge of the clock. As a result, one bit of data is transferred in each direction during one clock cycle. After eight clock cycles, the contents of the shift registers have been exchanged and transfer is complete.

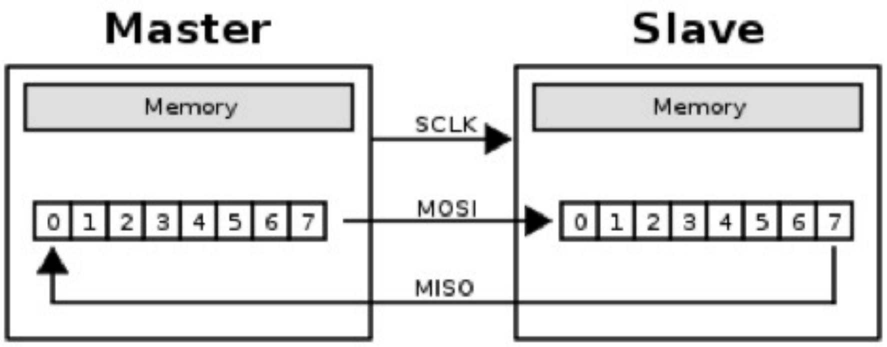


Figure 2: SPI data transmission represented by two registers

3.4 SPI clock polarity and phase

The master must configure the clock polarity (CPOL) and phase (CPHA) with respect to the data based on the property of slave. Table 1 describes the four different modes of CPOL and CPHA. Figure 3 shows a complete transfer of 4 bits using SPI in mode3 (CPHA = 1; CPOL = 1)

CPHA = 0: Read (written) on the leading (trailing) edge of each clock pulse.
CPHA = 1: Written (read) on the leading (trailing) edge of each clock pulse.
CPOL = 0: Clock idles low between transfers.
CPOL = 1: Clock idles high between transfers.

Mode	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

Table 1: The four standard modes for the clock in SPI

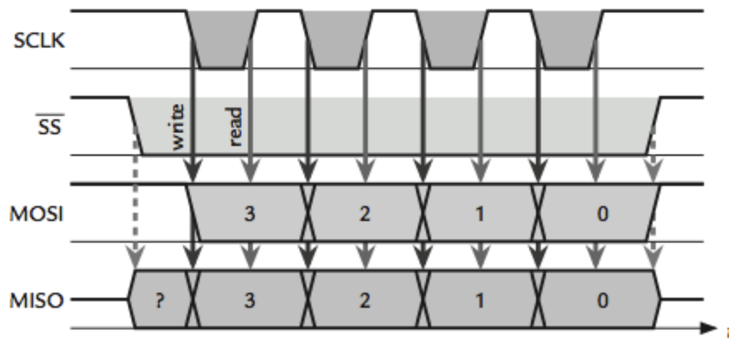


Figure 3: a complete transfer of 4 bits using SPI in mode 3

4 Description of LTC2382-16

4.1 Introduction to LTC2382-16

The LTC2382-16 is a low noise, low power, high speed 16-bit successive approximation register (SAR) ADC (Figure 4). It has a speed SPI-compatible serial interface that support 1.8V, 2.5V, 3.3V and 5V logic while also featuring a daisy chain mode. The fast 500Ksps throughout with no cycle latency makes the LTC2382-16 ideally suited for a wide variety of high-speed applications.

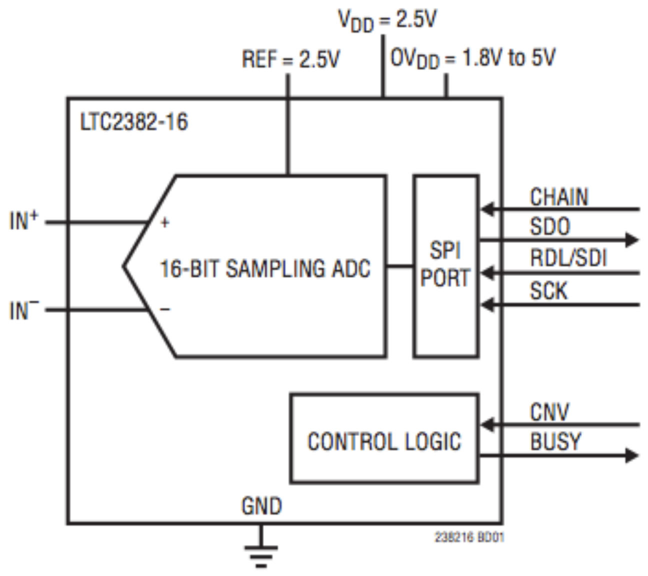


Figure 4: functional block diagram of LTC2382-16

4.2 Conversion timing using the SPI

The simplest way to operate the LTC2382-16 is to set it in Normal Mode with single device. In this mode, CHAIN and RDL/SDI connected to ground, SDO is enabled and the MSB(D15) of the new conversion data available at the falling edge of BUSY. Then, MCU can use SCK to take data from SDO pin of ADC (Figure 5).

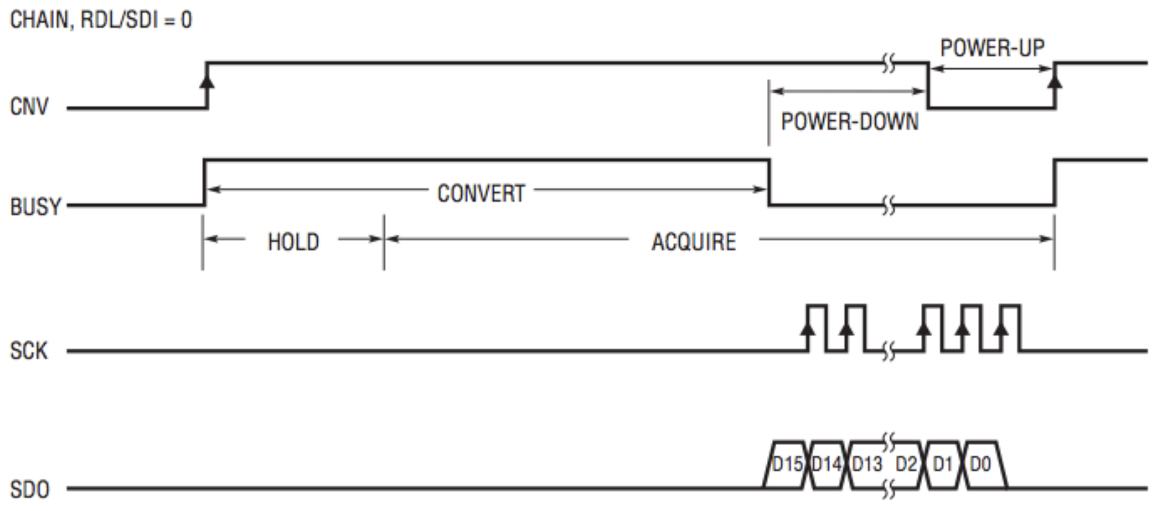


Figure 5: conversion timing using the SPI

5 Hardware implementation

Figure 6 shows the pin connections between MCU and ADC. The ADC works in Normal Mode, and as a slave. MCU is configured as a master and provides clock signal as well as control signal(SS) to the ADC. As a result, ADC sends back converted data from its SDO pin to SDI pin of MCU.

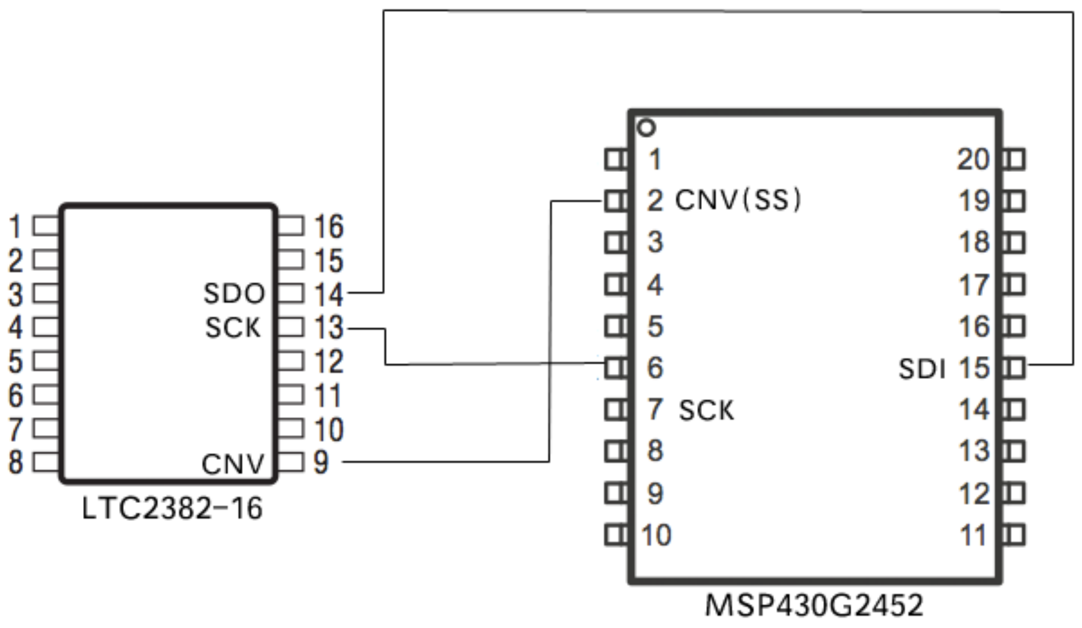


Figure 6: pin connections between ADC and MUC for SPI communication

6 Software implementation

6.1 Introduction to software configuration

P2 is configured as an output and drove it high while the USI is active to act as CNV(SS). P7 provides clock signal in order to synchronous the whole system. P15 is the data entrance of MCU. The followings are the details on software configuration:

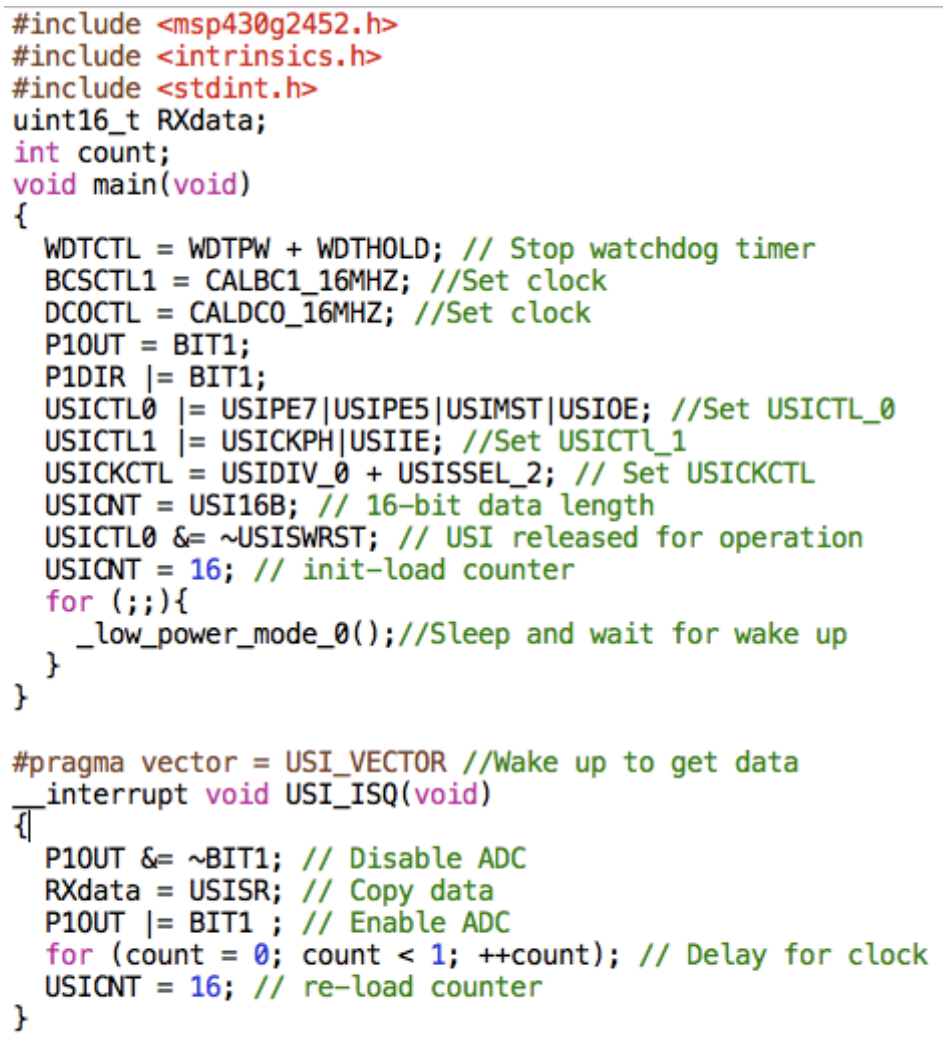
-Input clock-

- The SMCLK is set to operate in 16MHz.

-USI(SPI mode)-

- The USIMST bit is set because this is an SPI master.
- The USIWRST bit is set during the write to USICTL0 to prevent any unwanted activity.
- The USICKPH bit is set, so it is equivalent to CPHA = 0, which means read on the leading edge and written on the trailing edge.
- The clock is taken from SMCLK(USISSEL_2) and divided by 1(USIDIV_0). It idles low, so clear USICKPL bit.
- After releasing USI by clearing USISWRST, _low_power_mode_0( ) put the MCU into LPM0 sleep mode and wait for interrupt.
- The USI16B is set in order to take 16-bit data from ADC.

6.2 Code in C



```c
#include <msp430g2452.h>
#include <intrinsics.h>
#include <stdint.h>
uint16_t RXdata;
int count;
void main(void)
{
  WDTCTL = WDTPW + WDTHOLD; // Stop watchdog timer
  BCSCTL1 = CALBC1_16MHZ; //Set clock
  DCOCTL = CALDCO_16MHZ; //Set clock
  P1OUT = BIT1;
  P1DIR |= BIT1;
  USICTL0 |= USIPE7|USIPE5|USIMST|USIOE; //Set USICTL_0
  USICTL1 |= USICKPH|USIIE; //Set USICTl_1
  USICKCTL = USIDIV_0 + USISSEL_2; // Set USICKCTL
  USICNT = USI16B; // 16-bit data length
  USICTL0 &= ~USISWRST; // USI released for operation
  USICNT = 16; // init-load counter
  for (;;){
    _low_power_mode_0();//Sleep and wait for wake up
  }
}

#pragma vector = USI_VECTOR //Wake up to get data
__interrupt void USI_ISQ(void)
{
  P1OUT &= ~BIT1; // Disable ADC
  RXdata = USISR; // Copy data
  P1OUT |= BIT1 ; // Enable ADC
  for (count = 0; count < 1; ++count); // Delay for clock
  USICNT = 16; // re-load counter
}
```

7 Results

7.1 Sampling rate

Wareforms	Represented
Orange	CNV(SS)
Green	BUSY
Blue	SCK
Violet	16-bit Data

Table 2: wareforms on figure 7

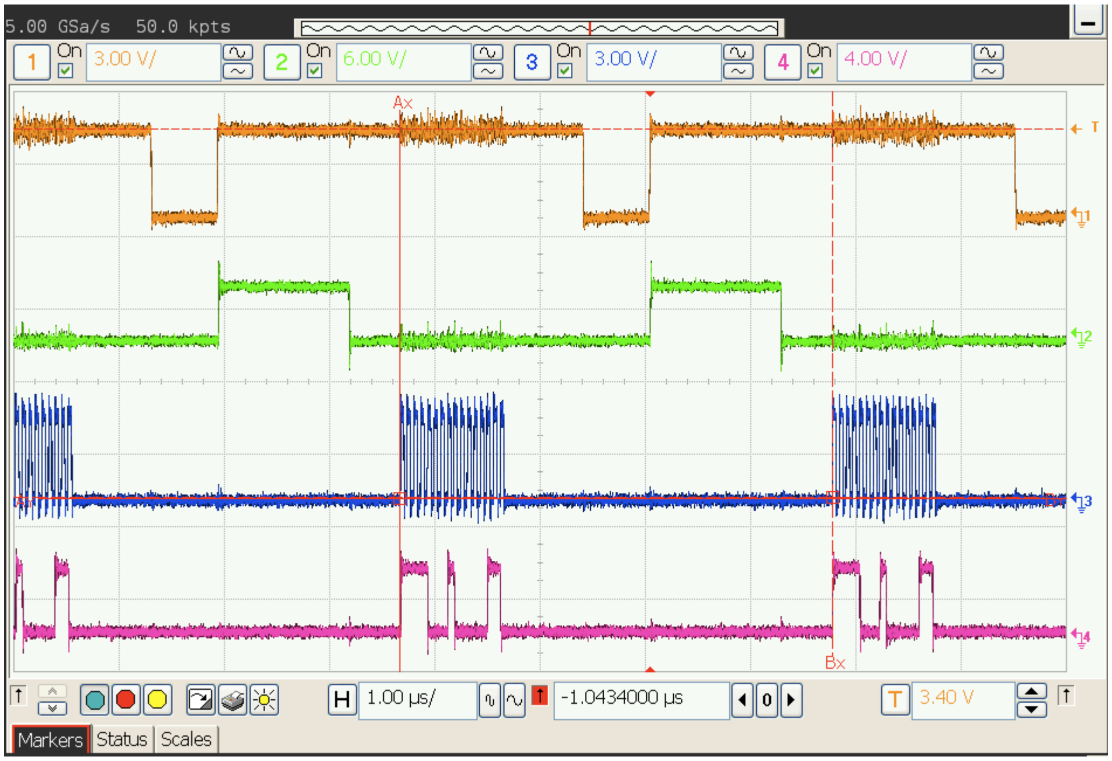
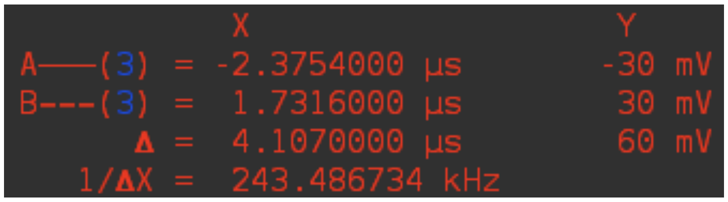


Figure 7: wareforms for CNV(SS), BUSY, SCK and data

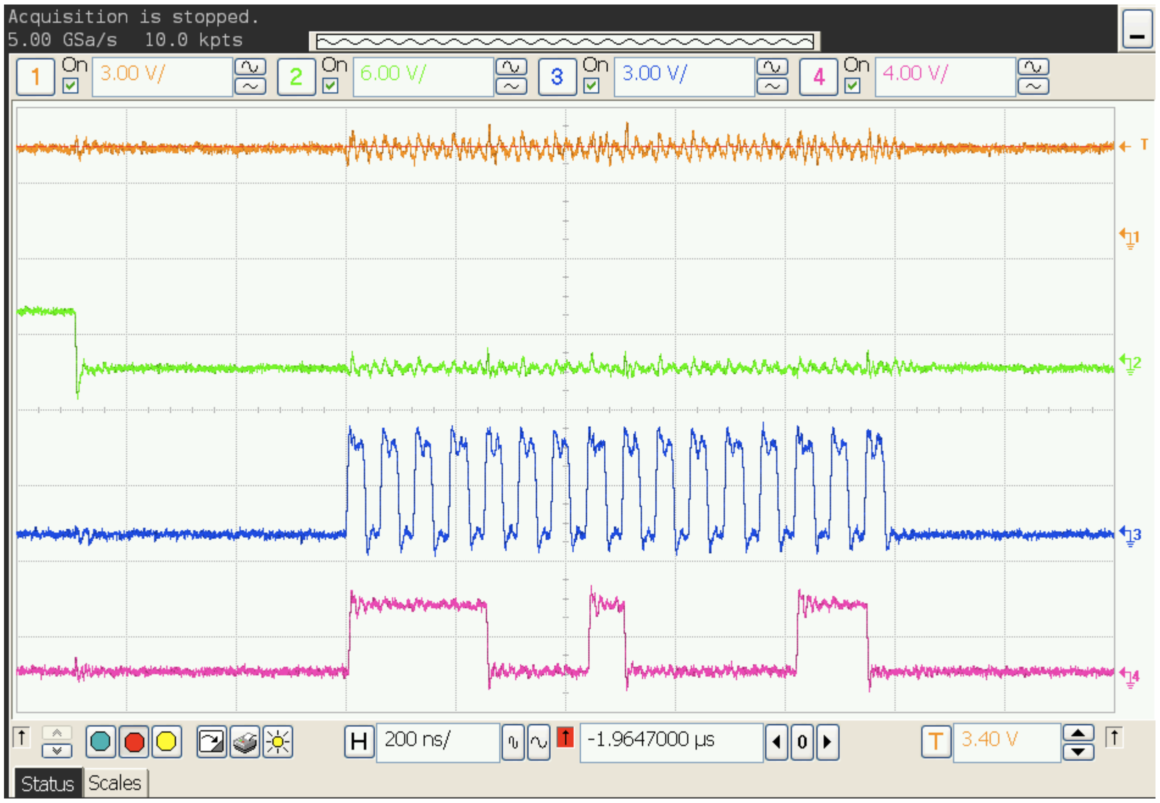


```
          X                       Y
A——(3) = -2.3754000 µs          -30 mV
B---(3) =  1.7316000 µs          30 mV
      Δ =  4.1070000 µs          60 mV
   1/ΔX =  243.486734 kHz
```

Figure 8: Sampling rate

Sampling rate = 1/period = 1/4.107us = 243.487 K samples/sec.

7.2 An example



Acquisition is stopped.
5.00 GSa/s 10.0 kpts

Figure 9: a sampled analog signal(Violet)

Ref. (V)	Input+(V)	Input-(V)	Result(DEC)	Result(V)	Error%
2.4964	2.3570	0	30851	2.3504	0.281

Table 3: an example

Result(V) = (30851/(65536/2))*2.4964 = 2.3504 V
Error% = ((2.3570-2.3504)/2.3570)% = 0.281%

8 Conclusion

While some other communication methods can be used to implement the connection between MCU and ADC, SPI is still an easy and straightforward choice for the most of the tasks.

9 References

[1] Wikipedia: Serial Peripheral Interface Bus
http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus
[2] MSP430G2452 Data Sheet
http://www.ti.com/lit/ds/symlink/msp430g2452.pdf
[3] LTC2382-16 Data Sheet
http://cds.linear.com/docs/Datasheet/238216f.pdf
[4] MSP430x2xx Family User's Guide
http://www.ti.com/lit/ug/slau144h/slau144h.pdf
[5] John H. D. (2008). MSP430 Microcontroller Basics. UK: Newnes.