Reading data on SPI

The new range of encoders 25P RCB1 E 05SPI S14 CW OCR LT is built with a chip from MELEXIS : MLX90363. The purpose of this application note is to guide the customer for their first steps with SPI bus.
The example given is voluntarily not optimized in order to give a total access to the bus. Of course most microcontrollers contain a SPI UART, it is not necessary to emulate it like on this app note.

The 90363 obeys to commands sent by the master. The main commands are Get1 and Nop. The other commands can be ignored for a simple application. For further information, please get the latest datasheet on the MELEXIS web site, page MLX90363.

The message, sent by the master, and therefore transmitted by 90363 is made of 8 bytes.

The most useful bytes are the first and second. They contain the angle value, coded on 14 bits. (0 is 0°, 16383 is for 360,0°)

Data1								Data0							
E1	E0	13	12	11	10	9	8	7	6	5	4	3	2	1	0

E1 and E0 form the following status :

E1	E0	Description
0	0	First diagnostic sequence not yet finished
0	1	Diagnostic fail
1	0	Diagnostic pass (previous cycle)
1	1	Diagnostic pass (new cycle completed)

The data are read on the falling edge of SCLK (CPHA=1, CPOL=0, LSBFE=0 in the langage of SPI)

The answer of a command on a SPI bus is dephased. That means that you send a command, but simultaneously you get the answer of the PREVIOUS command. This is not an issue if the readings are continuous.

But if you want to get an angle value at an exact instant of time, you must proceed in 2 steps :

-Set SS (Slave select) at level 0
-Wait at least 1µs
-Send a Get1 command to 90363; It will act as a trigger, ignoring the answer.
-Set SS at level 1

-Wait at least 1ms

-Set SS at level 0
-Send a Nop command. The answer is the angle measured at the previous Get1 command, even if the Nop command is sent after a consequent amount of time.
-Set SS at level 1

-Wait at least 1ms

Example of code (PIC18F2431, XC8 compiler)

```
/*
 * File:   main.c
 * Author: Denis Stremplewski
 *
 * Created on 9 march 2018, 14:26
 */
#define USE_OR_MASKS

#include <xc.h>
#include <plib\timers.h>
#include "config.h"

#define _XTAL_FREQ 40000000UL

#define mosi LATCbits.LATC6
#define miso PORTCbits.RC7
#define sclk LATBbits.LATB0
#define ss LATBbits.LATB1

typedef union {
    struct {
        unsigned char low, high;
    };
    unsigned int val_int;
} value_int;

struct _mlx90363 {  // The MLX90363 returns always
8 bytes
    unsigned char data0; // LSB angle
    unsigned char data1; // MSB angle (2 MSB bits
must be cleared)
    unsigned char data2;
    unsigned char data3;
    unsigned char data4;
    unsigned char data5;
    unsigned char data6;
    unsigned char data7; // CRC

};

void MLX90363Init(void) {
    TRISCbits.RC6 = 0;
    TRISCbits.RC7 = 1;
    TRISBbits.RB0 = 0;
    TRISBbits.RB1 = 0;

    ss = 1;
    sclk = 0;
    mosi = 0;
}

char writeAndReadByteOnSPI(char v) {
    char c = 0;
    for (char i = 0; i < 8; i++) {
        if ((v & 0x80) == 0)
            mosi = 0;
        else
            mosi = 1;
        sclk = 1;
        __delay_us(10);
        v = v << 1;
        sclk = 0;
        __delay_us(10);
        c = c << 1;
        c = c + miso;
        __delay_us(10);
    }
    return c;
}


struct _mlx90363 readMlx90363(void) {
    struct _mlx90363 u;

    // Sending a Get1 Command (u.data will contain
the previous data)
    ss = 0;
    u.data0 = writeAndReadByteOnSPI(0x00);
    u.data1 = writeAndReadByteOnSPI(0x00);
    u.data2 = writeAndReadByteOnSPI(0xFF);
    u.data3 = writeAndReadByteOnSPI(0xFF);
    u.data4 = writeAndReadByteOnSPI(0x00);
    u.data5 = writeAndReadByteOnSPI(0x00);
    u.data6 = writeAndReadByteOnSPI(0x13);
    u.data7 = writeAndReadByteOnSPI(0xEA);
    ss = 1;

    __delay_ms(1);

    // Sending a NOP command & get data. U.data
contains the data associated with the previous Get1
    // If the reading is continuous, only Get1 is
necessary. But if you want get the data at the
    // moment of a trigger, it necessary to proceed
in 2 steps : first a Get1 as a trigger, then a Nop
to read data.

    ss = 0;
    u.data0 = writeAndReadByteOnSPI(0x00);
    u.data1 = writeAndReadByteOnSPI(0x00);
    u.data2 = writeAndReadByteOnSPI(0xAA);
    u.data3 = writeAndReadByteOnSPI(0xAA);
    u.data4 = writeAndReadByteOnSPI(0x00);
    u.data5 = writeAndReadByteOnSPI(0x00);
    u.data6 = writeAndReadByteOnSPI(0xD0);
    u.data7 = writeAndReadByteOnSPI(0xAB);
    ss = 1;

    __delay_ms(1);

    return u; // returns the last data
}

void main(void) {
    MLX90363Init();
    struct _mlx90363 u;
    value_int alpha;
    unsigned int angle;

    while (1) {
        u = readMlx90363();

        alpha.low=u.data0;
        alpha.high=u.data1 & 0x3F;
 // ****************************************
 // *** here is the angle value on 14 bits ***
 angle=alpha.val_int;
 // ****************************************

        Nop();
    }

    while (1);
    return;
}
```