

Scope

This application note gives some guidelines to implement a serial communication interface between a microcontroller and the MLX90129. Basic knowledge about the characteristics of the MLX90129 Serial Peripheral Interface (SPI) and source code examples written in C are provided allowing an easy implementation.

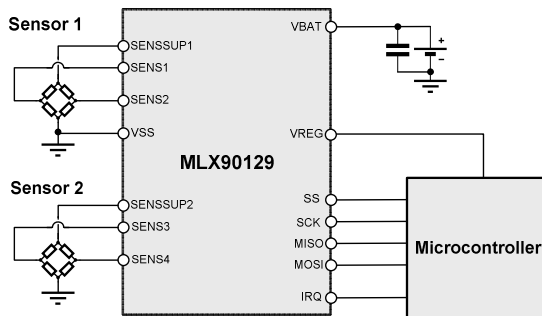
Applications

- ☐ Cold chain monitoring
- ☐ Asset management and monitoring (security and integrity)
- ☐ RFID front end system
- ☐ Building automation
- ☐ Industrial, medical and residential control and monitoring

Related Melexis Products

Part No.	Temperature suffix	Package Code	Option code
MLX90129RGO	R (-40°C to 105°C)	GO [TSSOP 20]	-

Typical Circuit



Introduction

Thanks to its SPI interface, the MLX90129 can be easily interfaced with most of the microcontrollers. In this application the MLX90129 is the slave and the microcontroller is the master of the communication. As SPI interface allows addressing all the memory areas of the MLX90129, it can be used to configure the device and to read data from sensors. It also allows setting some security features which are not available by RFID. The application when MLX90129 is used as an SPI master will not be discussed in this document. For more information, please refer to the application note "data logging". This application note puts together all the necessary information, such as electrical specification and source code, in order to allow a quick and easy implementation of the SPI interface.

Contents

MLX90129 SPI description.....	2
Electrical specification.....	2
SPI basic timing specification.....	3
SPI frame specification.....	4
Header file.....	6
SPI initialization routine	7
Basic SPI communication routine	7
Write routine.....	7
Read routine	9
Update routine	10
Main routine	11
Trouble shooting notes.....	12
Conclusion.....	12

MLX90129 SPI description

The SPI bus of the MLX90129 is composed of four lines:

Name	Description	Pin
SS	Slave Select	12
SCK	Serial Clock	11
MOSI	Master data Output, Slave data Input	10
MISO	Master data Input, Slave data Output	9

The serial clock frequency is up to 1 MHz. The data are sent MSB first on MISO and MOSI line. The SPI mode is CPOL=0 and CPHA=0. Please refer to the MLX90129 datasheet for more information.

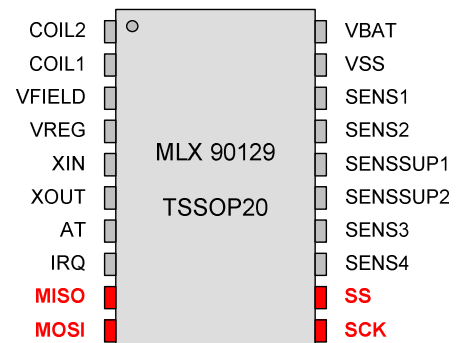


Fig 1: Pin out

Electrical specification

The MLX90129 can work with two different internal reference voltages called High Voltage (HV) and Low Voltage (LV). The mode depends of the bit 3 of the EELATCH #04.

EELATCH#04 [3] PreReg_LV = '0' => High Voltage mode

EELATCH#04 [3] PreReg_LV = '1' => Low Voltage mode

The electrical specifications of the SPI bus differ in function of the MLX90129 internal reference voltage.

For High Voltage mode:

Parameter	Description	Min	Typ	Max	unit
VIH	Input High Voltage (SPI slave)	2.1	3.0	3.5	V
VIL	Input Low Voltage (SPI slave)	-0.3	0	0.9	V
VOH	Output High Voltage (I sunk = -2 mA)	2.2		-	V
VOL	Output Low Voltage (I forced = 2 mA)	-		0.4	V

For Low Voltage mode:

Parameter	Description	Min	Typ	Max	unit
VIH	Input High Voltage (SPI slave)	1.4	2.0	2.5	V
VIL	Input Low Voltage (SPI slave)	-0.3	0	0.6	V
VOH	Output High Voltage (I sunk = -2 mA)	1.2		-	V
VOL	Output Low Voltage (I forced = 2 mA)	-		0.4	V

SPI basic timing specification

To ensure a safe SPI communication between the master and the slave, the master needs to respect some basic timing as following described.

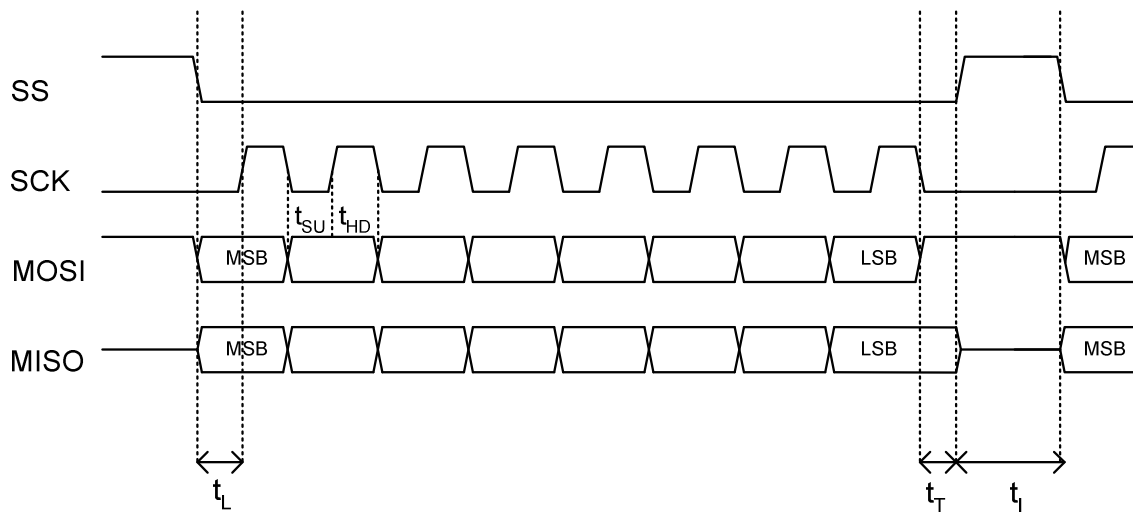


Fig 2: Timing diagram

Timing specifications

Parameter	Description	Slave side		Units
		Min	Max	
tch	SCK high time	500	-	ns
tcl	SCK low time	500	-	ns
tSU	Setup time of data, after a falling edge of SCK	100	-	ns
tHD	Hold time of data, after a rising edge of SCK	500	-	ns
tL	Leading time before the first SCK edge			
	_ when the MLX90129 is not in sleep mode	600	-	ns
	_ when the MLX90129 is in sleep mode (***)	1.5	-	ms
tT	Trailing time after the last SCK edge	500	-	ns
tI	Idling time between transfers (SS=1 time)	500	-	ns

(***) –A specific leading time before the first SCK edge is necessary to wake-up the MLX90129 from sleep mode

SPI frame specification

The SPI frames exchanged between the master and the slave are specified in the following diagram.

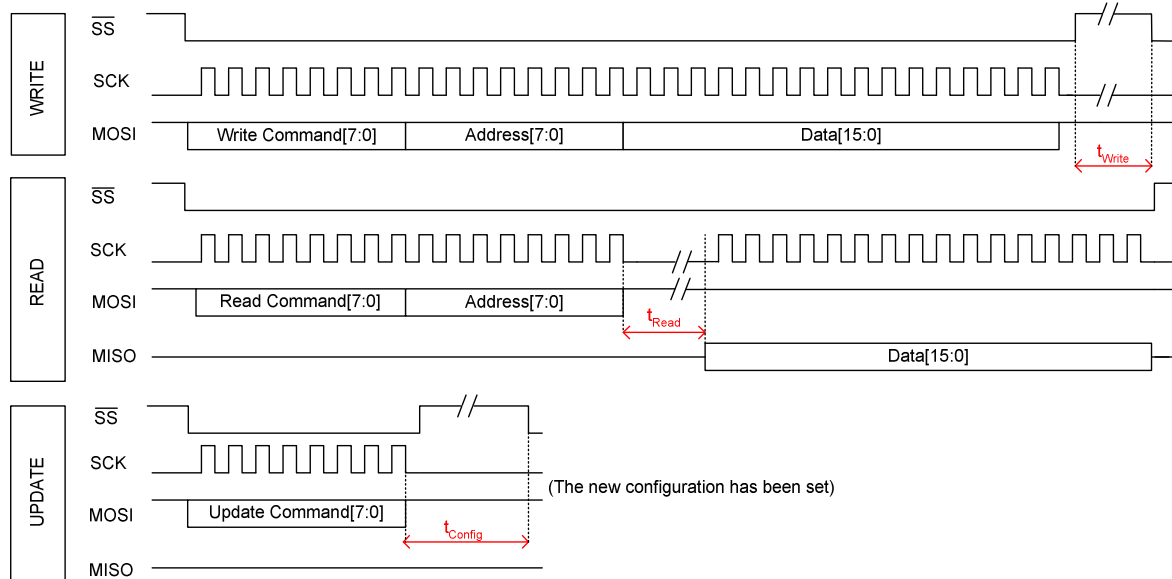


Fig 3: Frame diagram

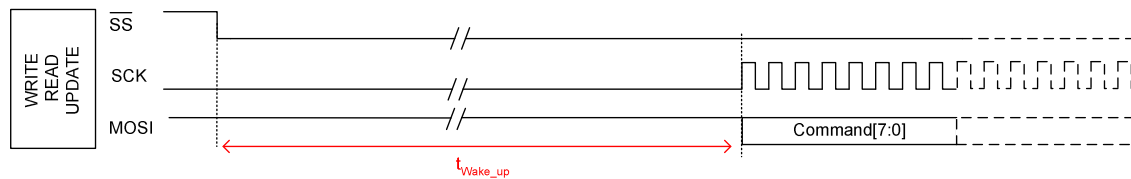


Fig 4: Frame diagram in sleep mode

Timing specifications

Parameter	Description	Slave side		Units
		Min	Max	
tRead (**)	Delay to read a register word	2	-	µs
	Delay to read an EEPROM word	50	-	µs
	Delay to read an EE-Latch word	2	-	µs
	Delay to get the ADC output code	430	(*)	µs
tWrite (**)	Delay to write a register word	500	-	ns
	Delay to write an EEPROM word	17	-	ms
	Delay to write an EE-Latch word	11	-	ms
tConfig	Execution delay for commands Update	1.5		ms
tWake_up	Delay before the first SCK edge when the MLX90129 is in sleep mode	1.5		ms

(*) – The conversion time depends on the programmed initialization time and on the ADC options.

(**) _ For the Read/Write Internal Devices commands, the delay depends on the nature of the so-called Internal Device: (Register, EE-Latch bank, ADC...)

Hardware implementation

The main concern about hardware implementation is the voltage of the SPI bus. In case of the I/O pins of the microcontroller are compliant with the voltage mode of the MLX90129 (3V for HV mode and 2V for LV mode), the Melexis device can be directly connected to the microcontroller as shown in the following example:

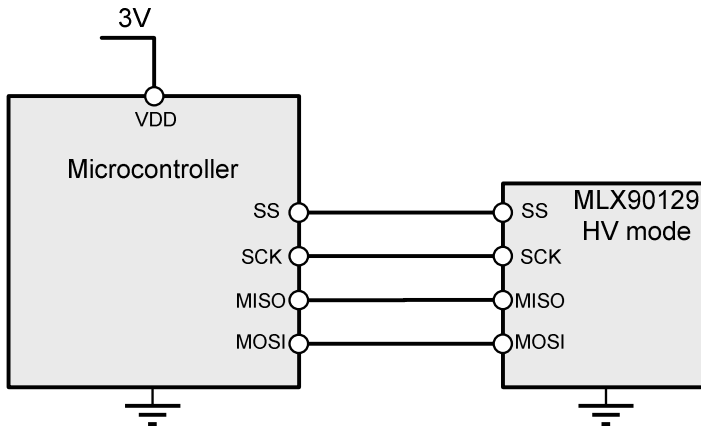


Fig 5: Hardware implementation

In case of the I/O pins of the microcontroller are not compliant with the voltage mode of the MLX90129, a level shifter has to be inserted between the Melexis device and the microcontroller as shown in the following diagram:

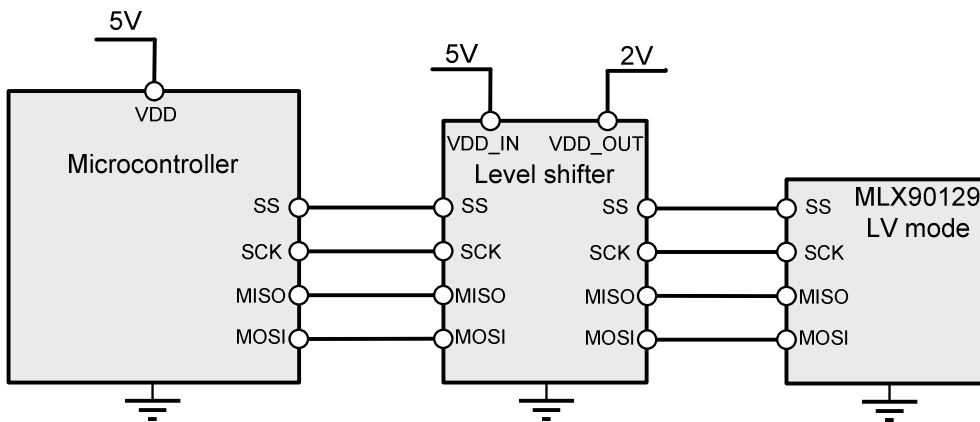


Fig 6: Hardware implementation with level shifter

Routines implementation

The routines have been developed with the Atmel Integrated Development Environment AVRStudio 4.15 and the C compiler WinAVR 2008. As code is in C, it is portable to other microcontrollers such as Microchip, STMicroelectronics, NXP, Freescale, Texas Instrument, etc. The routines use the SPI hardware block of the microcontroller ATmega168 because this greatly simplifies the code and allows fast SPI communication. The SPI hardware block is available on most of the current microcontrollers. However, a “software” SPI bus can be developed and implemented on any general purpose I/O port.

The following routines have been implemented for an Atmega168 using its internal oscillator running at 8 MHz.

Header file

The header file SPI_90129.h allows an abstraction between hardware and higher level routines. In the ATmega168 pin configuration:

- PB2 pin is assigned to control the SS line (output)
- PB3 pin is assigned to control the MOSI line (output)
- PB4 pin is assigned to control the MISO line (input)
- PB5 pin is assigned to control the SCK line (output)

```

/*-----SPI_90129.h-----*/

#ifndef SPI_90129_H
#define SPI_90129_H

#ifdef _AVR_IOM168_H_ // Definition for ATmega168
/*----- SPI -----*/
SS    => PB2 =>  OUTPUT
MOSI  => PB3 =>  OUTPUT
MISO  => PB4 =>  INPUT
SCK   => PB5 =>  OUTPUT
----- */
#define PORTSPI  PORTB
#define SS      PB2
#define MOSI    PB3
#define MISO    PB4
#define SCK     PB5

#define SS_H    PORTB=(1<<SS) // SS line with high level output
#define SS_L    PORTB=(0<<SS) // SS line with low level output

#define DDR_SPI  DDRB // set the pins of PORTB as Input or Output
#define DD_SS    DDB2 // set the pin PB2 as Input or Output
#define DD_MOSI  DDB3 // set the pin PB3 as Input or Output
#define DD_MISO  DDB4 // set the pin PB4 as Input or Output
#define DD_SCK   DDB5 // set the pin PB5 as Input or Output

#endif
#endif
/*-----*/

```

SPI initialization routine

The routine SPI_MasterInit is provided by Atmel to initialize the SPI hardware block of the ATmega168 with the required configuration. The SPI clock rate is set at 1MHz, the MSB is sent first and the data mode CPOL=0 and CPHA=0 is selected.

```
/*-----SPI_MasterInit-----*/
void SPI_MasterInit(void)
{
    /* Set SS, MOSI and SCK output, MISO input */
    DDR_SPI = (1<<DD_MOSI) | (1<<DD_SCK) | (1<<DD_SS);
    /* Enable SPI, Master, set clock rate fck/8 = 1MHz */
    SPCR = (1<<SPE) | (1<<MSTR) | (1<<SPR0);
    SPSR = (1<<SPI2X);
}
/*-----*/
```

Basic SPI communication routine

The routine SPI_MasterTransmit is provided by Atmel to send the data "cData" to the slave and get the slave response in SPDR register. This routine automatically manages the SCK, MOSI, MISO signals. Only the SS signal has to be externally controlled.

```
/*-----SPI_MasterTransmit-----*/
char SPI_MasterTransmit(char cData)
{
    /* Start transmission */
    SPDR = cData;
    /* Wait for transmission complete */
    while(!(SPSR & (1<<SPIF)));
    /* Return Data Register */
    return SPDR;
}
/*-----*/
```

Write routine

The routine SPI_MLX90129_write sends a write command, the address targeted and the 2-byte data to write and then wait the required time for data writing. According to the MLX90129 datasheet, the write commands for the different memory domains are:

Command	Code	Operation
EEP_WR	0x0E	Write the addressed EEPROM word
REG_WR	0x09	Write the addressed register in the Register File
DEV_WR	0x18	Write a word into the selected internal device (Control, EELatches)

```

/*-----SPI_MLX90129_write-----*/
void SPI_MLX90129_write(void)
{
    char write_command;
    char address;
    char data_MSB;
    char data_LSB;

    // Write EEPROM at address 0x11 with data 0xABCD
    write_command=0x0E;
    address=0x11;
    data_MSB=0xAB;
    data_LSB=0xCD;

    SS_L; // SS low selects the slave for the communication
    //_delay_us(1500); // delay only when MLX90129 is in sleep mode

    SPI_MasterTransmit(write_command);
    SPI_MasterTransmit(address);
    SPI_MasterTransmit(data_MSB);
    SPI_MasterTransmit(data_LSB);

    SS_H; //SS high deselects the slave and end the communication

    _delay_ms(17); // delay to write EEPROM
    //_delay_ms(11); // delay to write EELatch

}
/*-----*/

```

The following screen shot shows the SPI bus lines during an EEPROM write operation. Data 0x0E11ABCD is sent to the device.

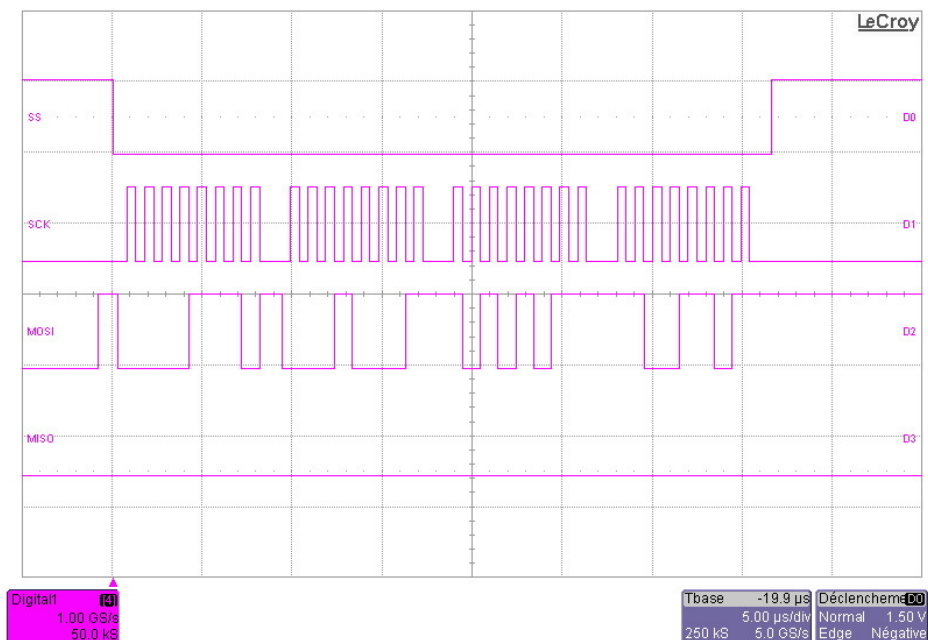


Fig 7: Write frame

Read routine

The routine SPI_MLX90129_read sends a read command, the address targeted and gets the 2-byte response. According to the MLX90129 datasheet, the read commands for the different memory domains are:

Command	Code	Operation
EEP_RD	0x0F	Read the addressed EEPROM word
REG_RD	0x0D	Read the addressed Register File word
DEV_RD	0x10	Read the addressed Internal Device word (Control, status, ADC...)

The time between the read request and data to be read differs depending on the memory domain. Routine has to be adapted to match with the timing requirements.

```

/*-----SPI_MLX90129_read-----*/
void SPI_MLX90129_read(void)
{
    char read_command;
    char address;
    char data_MSB;
    char data_LSB;

    // Read Register at address 0x11
    read_command=0x0F;
    address=0x11;
    data_MSB=0xFF;
    data_LSB=0xFF;

    SS_L; // SS low selects the slave for the communication
    //_delay_us(1500); // delay only when MLX90129 is in sleep mode

    // send read command and targeted address
    SPI_MasterTransmit(read_command);
    SPI_MasterTransmit(address);

    //_delay_us(50); // delay to read EEPROM
    _delay_us(2); // delay to read Register file
    //_delay_us(2); // delay to read EELatch
    //_delay_us(430); // minimum delay to read ADC output

    // send 0x00 (or other random data) to generate SPI clock
    // and get the slave answer
    data_MSB=SPI_MasterTransmit(0x00);
    data_LSB=SPI_MasterTransmit(0x00);

    SS_H; // SS high deselects the slave and end the communication
}
/*-----*/

```

The following screen shot shows the SPI bus lines during a Register read operation. Data 0x0F11 is sent to the device and data 0xABCD is retrieved on the MISO line.

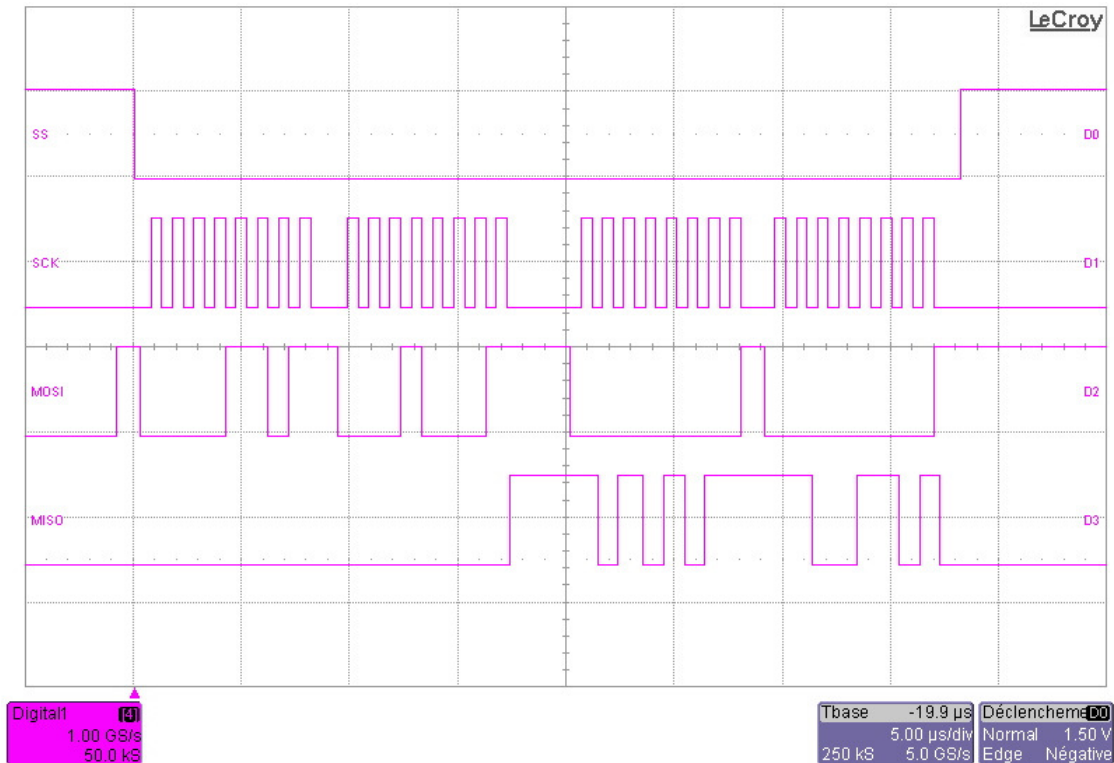


Fig 8: Read frame

Update routine

The routine SPI_MLX90129_update sends the Register File update command (0x1C) and wait the required time before the update is completed.

```
/*-----SPI_MLX90129_update-----*/
void SPI_MLX90129_update(void)
{
    char update_command;

    // Update command =0x1C
    update_command=0x1C;

    SS_L; // SS low selects the slave for the communication

    // send the update command
    SPI_MasterTransmit(update_command);

    SS_H; // SS high deselects the slave and end the communication

    _delay_us(1500); // delay for the update of the registers
}
/*-----*/
```

The following screen shot shows the register update operation. Data 0x1C is sent to the device.

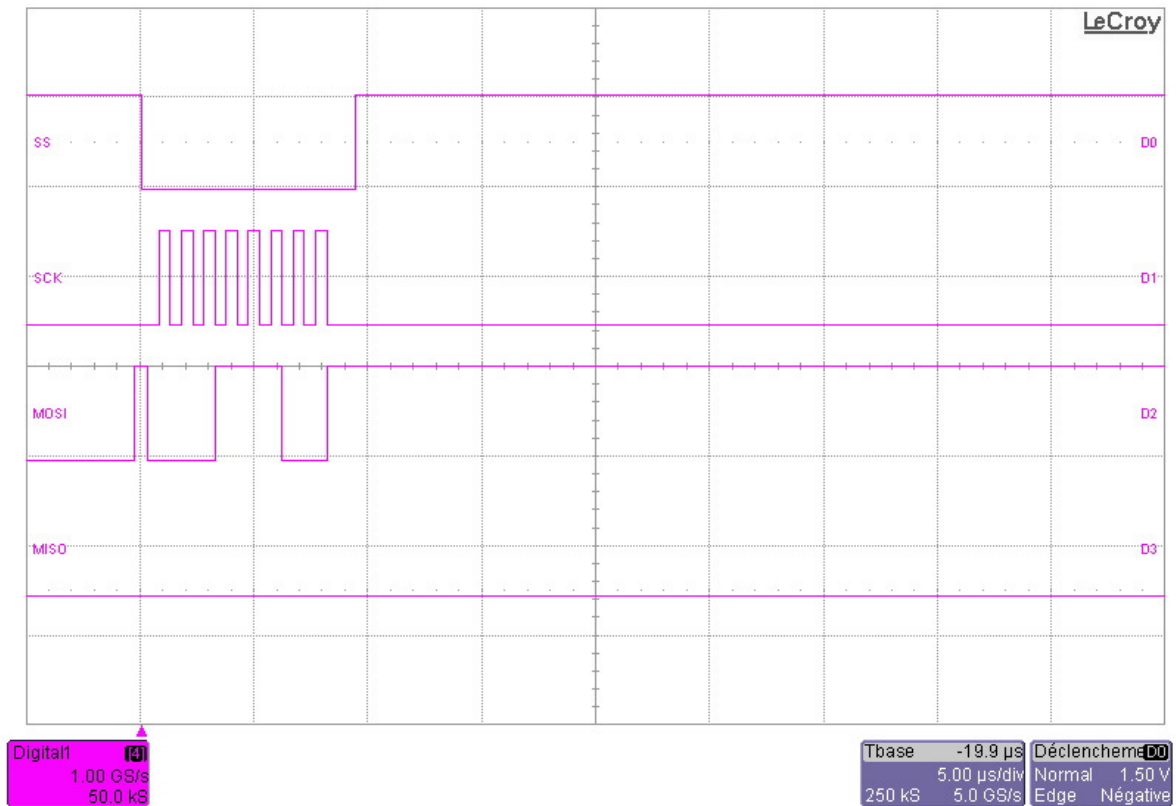


Fig 9: Update frame

Main routine

The main routine is an example of using the previous routines and also a test routine in order to check the SPI interface implementation. First SPI bus of the microcontroller is initialized as a master, then data 0xABCD is written in the EEPROM, finally a register file update is performed and data are read in the register file. For good implementation, the collected data shall be 0xABCD.

```
int main ( void )
{
    SS_H;
    SPI_MasterInit();

    SPI_MLX90129_write(); // write 0xABCD in EERPOM #11
    SPI_MLX90129_update();// update register file
    SPI_MLX90129_read(); // read in Register #11

    return(0);
}
```

Trouble shooting notes

In case of communication failure (the MLX90129 does not respond to any commands) the following list has to be checked in order to identify the issue.

- Is the SPI bus lines well conneted (MISO with MISO, SS with SS ...) ?
- Is the MLX90129 well supplied (check Vbat and Vreg voltages)?
- Is the voltage of the SPI lines compliant with the microcontroller and MLX90129 requirement (Mode Low Volt, High Volt) ?
- Are the SPI lines monitored with an oscilloscope looking similar like the screen shot of this application note?
- Is the command sent compliant with the MLX90129 ?
- Is the MLX90129 in slave mode (bit 0 EEPROM #05 = 0) (to be checked by RFID)?
- Is the timing between SS and SCK is respected if the MLX90129 is in sleep mode ?

Conclusion

This application note allows a fast implementation of the SPI interface of the MLX90129. Used as a peripheral interface of a microcontroller, the MLX90129 can add a large number of features to the application. In addition to the SPI bus, the IRQ pin can be connected to the microcontroller allowing the firmware to handle the programmable interruptions defined in the MLX90129 datasheet. More information and documentation can be found on the Melexis website, www.melexis.com. Suggestions and questions can be sent to rfid@melexis.com.