## Scope

This application note describes how to implement SMBus communication with MLX90614 Infra-Red thermometers. Code is for Microchip's PIC®18. The example is a read from MLX90614's RAM. The software implementation of the SMBus communication is used so that the source code can be migrated for other families 8 bits PIC MCU with small changes. The development tools used are MPLAB IDE and MPASM (microchip assembler) which are free to use from www.microchip.com.

## Applications

- ■ High precision non-contact temperature measurements;
- ■ Thermal Comfort sensor for Mobile Air Conditioning control system;
- ■ Temperature sensing element for residential, commercial and industrial building air conditioning;
- ■ Windshield defogging;
- ■ Automotive blind angle detection;
- ■ Industrial temperature control of moving parts;
- ■ Temperature control in printers and copiers;
- ■ Home appliances with temperature control;
- ■ Healthcare;
- ■ Livestock monitoring;
- ■ Movement detection;
- ■ Multiple zone temperature control – up to 100 sensors can be read via common 2 wires
- ■ Thermal relay/alert
- ■ Body temperature measurement

## Related Melexis Products

EVB90614 is the evaluation board which supports the MLX90614 devices.

## Other Components Needed

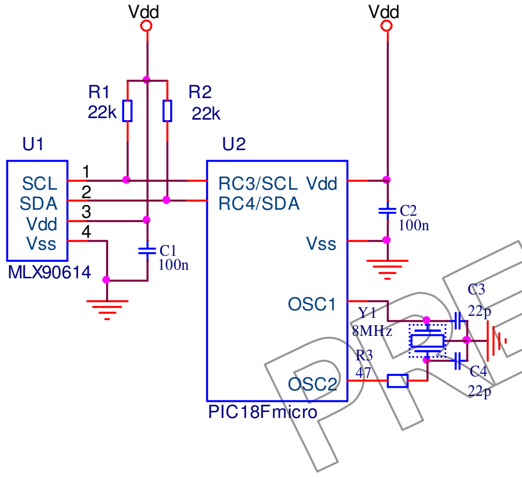
Elements used in the schematics within current application note include:
SMD ceramic capacitors C1 and C2 100nF 16V or higher.
SMD ceramic capacitors C3 and C4 22pF 16V or higher.
SMD Resistors R1 and R2 22 kOhm 5%.
SMD Resistor R3 47 Ohm 5%.
Quarz resonator Y1 8.00MHz
PIC18F4320 microcontroller or other from the Microchip's PIC18 family.

Accompanying files:

1. MPASM files to include in existing project, "SMBusFiles"
2. MPLAB project, "SMBusProject"

Project is build, file "main.hex" can be programmed in a PIC18F4320. Also, project can be used as a "start with" base.
As provided the project will read Tobj,1 from MLX90614 (power supply control is not included), and transmit it via UART (ASCII coded, CR (0x0D) after each value, 8 bit data, one stop bit, no parity bit, 19 200 baud if a 11.0592 MHz crystal is used). Format is 15 bit unsigned integer, right-justified. Resolution is 0.02 degrees Kelvin / LSB. Refer to explanation of the routines below for examples on the temperature format.

## Typical Circuit



## Explanation

The connection of MLX90614 to MCU is very simple. Two general purpose pins RC3 and RC4 of the PIC18 are used. Two pull up resistors R1 and R2 are connected to Vdd and to SCL and SDA lines respectively. C1 is the local power supply bypass decoupling capacitor. The MLX90614 needs that for bypassing of the on-chip digital circuitry switching noise.
C2 has the same function for the microcontroller. The well known value 100nF (SMD ceramic type) is typically adequate for these components.
Note that the power supply typically needs more capacitors (like 100µF on voltage regulator input and output), not shown on the schematic.

The components R1, C3, C4 and Y1 are used for the MCU oscillator. On-chip RC oscillators can also be used. For example, with a PIC18F4320 internal RC oscillator set to 8 MHz can be used without problem. SMBus is synchronous communication and therefore is not critical to timings. With 8 MHz crystal (2 MIPs) the SMBus clock is 54 kHz and one read frame takes 1 ms. On a test setup the SMBus was working with PIC clock from 2 MHz to 11.0592 MHz.
Refer to MLX90614 datasheets, Application Note 390119061402, “SMBus communication with MLX90614” and SMBus standard for details. MLX90614 comes in 5V and 3V versions. PIC18LF4320 could be used with the 3V version (MLX90614Bxx) and both PIC18F4320 and PIC18LF4320 – with the 5V version (MLX90614Axx).

Below is the assembly language code. It consists of:

- definition of the RAM usage (as well as PIC I/Os)
- subroutines

  ;Name: START_bit
  ;Function: Generate START condition on SMBus
  ;Name: STOP_bit
  ;Function: Generate STOP condition on SMBus
  ;Name: TX_byte
  ;Function: Send a byte on SMBus
  ;Name: RX_byte
  ;Function: Receive a byte on SMBus
  ;Name: delay
  ;Function: Produces time delay depending on the value in counterL
  ;Name: PEC_calculation
  ;Function: Calculates the PEC of received bytes
- Macros definitions
- “Asembly of everything together” – main program

Code ends with waveforms that the firmware generates.

## Build and use

What is needed to read the object temperature (refer to MLX90614 Data Sheet available at www.melexis.com for details):

- use the accompanying MPLAB project, make a new one, or use existing one
- “main.asm” file in both ZIPs: “SMBus files” and “SMBus project” that come with the current Application note is enough for full configuration of the PIC MCU, and it also contains the “include” directives for the other files used.

Code that reads MLX90614 then consist of:

```
        MOVLW  SA                          ;
        MOVWF  SlaveAddress                ; SA -> SlaveAddress
        MOVLW  RAM_Address|RAM_Access      ; Form RAM access command + RAM
        MOVWF  command                     ; address

Readloop
        CALL   MemRead                     ; Read RAM address
```

Result will be in DataH:DataL.

Factory default SMBus Slave Address (SA) for all MLX90614 is 0x5A.

The most important RAM addresses of MLX90614 are:

| RAM_Address | Temperature read |
|---|---|
| 0x06 | Ta – die temperature |
| 0x07 | Tobj,1 – object temperature (MLX90614xAx)<br>zone 1 object temperature (MLX90614xBx) |
| 0x08 | zone 2 object temperature (MLX90614xBx only). |

To read the die temperature (RAM address 0x06) of MLX90614 with slave address 0x5A (factory default) the code would be:

```
        MOVLW  0x5A                        ;
        MOVWF  SlaveAddress                ; SA -> SlaveAddress
        MOVLW  0x06                        ; Form RAM access command + RAM
        MOVWF  command                     ; address

Readloop
        CALL   MemRead             ; Read RAM address
```

DataH:DataL will consist of 15 bit temperature in unsigned integer, right-justified format. Resolution is 0.02 degrees Kelvin / LSB. For example,

0 °K would be represented as 0x0000
0.02 °K – 0x0001
0.04 °K – 0x0002
Ta minimum for MLX90614 -40 °C = 233.15 °K – 0x2D8A
Ta of +25 °C = 298.15 °K – 0x3A3C
Ta maximum for MLX90614 +125 °C = 398.15 °K – 0x4DC4

To read Tobj,1 temperature:

```
        MOVLW  0x5A                        ;
        MOVWF  SlaveAddress                ; SA -> SlaveAddress
        MOVLW  0x07                        ; Form RAM access command + RAM
        MOVWF  command                     ; address

Readloop
        CALL   MemRead                     ; Read RAM address
```

Output temperature format will be the same, for example,
DataH:DataL would be 0x3C94 for Tobj,1 = +37°C = 310.15°K.

Note that the calibration ranges for MLX90614 are
Ta -40…+125°C
To -70…+382°C

All MLX90614 accept SA=0x00. There are two important consequences of that:

- any MLX90614 can be both read and written without knowing what SA is programmed in the EEPROM (if a single MLX90614 is present on the SMBus)
- communication with more than one MLX90614 on an SMBus at SA 0x00 would not work

For example, read of SA from a single MLX90614 on a SMBus would be:

```
        MOVLW  0x00                      ;
        MOVWF  SlaveAddress              ; SA -> SlaveAddress
        MOVLW  0x2E                      ; Form EEPROM access command + EEPROM
        MOVWF  command                   ; address

Readloop
        CALL   MemRead                   ; Read RAM address
```

The Slave Address (read from EEPROM) would be on DataH:DataL. In this case the SA for the SMBus will be the right 7 bits.

ERROR HANDLING:
SMBus provides two general error indication mechanisms:

- PEC, Packet Error Code, a CRC-based check of the entire communication frame
- Acknowledge of each byte

Code given with this Application Note handles these in the following manner:

When a module returns “not acknowledge” then the firmware is trying to retransmit the byte again. The value in register Nack_Counter defines how many time the byte to be retransmitted in case that a module returns “not acknowledge”.

Register PEC contains CRC calculated for the entire communication SMBus frame. This value is compared with the received value in the last byte of the message, which represents the CRC returned by the module. If they are not equal the entire message is retransmitted again.

SMBus subroutines used for communication with MLX90614

```
;*****************************************************************************************************
;                                    DEFINE GPR AND CONSTANT
;*****************************************************************************************************

                  CBLOCK       H'00'
                    TX_buffer
                    TX_temp
                    Bit_counter
                    RX_buffer
                    flagreg0
                    counterL
                    DataL
                    DataH
                    PecReg
                    SlaveAddress
                    command
                    Nack_Counter
                    PEC4
                    PEC3
                    PEC2
                    PEC1
                    PEC0
                    PEC
                    CRC4
                    CRC3
                    CRC2
                    CRC1
                    CRC0
                    CRC
                    BitPosition
                    shift
                  ENDC
;delay constants
#define   TBUF            d'2'

;SMBus control signals
#define   _SCL_IO         TRISC,3      ;
#define   _SDA_IO         TRISC,4      ;
#define   _SCL            PORTC,3      ; RC3 is defined as SCL line
#define   _SDA            PORTC,4      ; RC4 is defined as SDA line

#define   bit_out         flagreg0,0   ; Define the bit sent on SDA line in transmit mode
#define   bit_in          flagreg0,1   ; Define the bit received from SDA in received mode

#define  RAM_Access    0x00            ; Define the MLX90614 command RAM_Accsess
#define  RAM_Address  0x07             ; Define address from MLX90614 RAM memory
#define  SA                0x00         ; Define SMBus device address
```

```
;************************************************************************************************************
;                                   START CONDITION ON SMBus
;************************************************************************************************************
;Name:          START_bit
;Function:      Generate START condition on SMBus
;Input:         No
;Output:        No
;Comments:   Refer to "System Management BUS(SMBus) specification Version 2.0" or
;                390119061402 application note for more information about SMBus
;               comunication with a MLX90614 module
************************************************************************************************************
START_bit

        _SDA_HIGH                       ;Set SDA line
        MOVLW       TBUF                ;
        CALL        delay               ; Wait a few microseconds
        _SCL_HIGH                       ;Set SCL line

        MOVLW       TBUF                ;Generate bus free time between Stop
        CALL        delay               ;and Start condition (Tbuf=4.7us min)

        _SDA_LOW                        ;Clear SDA line
        MOVLW       TBUF                ;Hold time after (Repeated) Start
        CALL        delay               ;Condition. After this period, the first clock is generated.
                                        ;(Thd:sta=4.0us min)
         _SCL_LOW                       ;Clear SCL line
        MOVLW       d'5'
        CALL        delay               ;Wait

         RETURN                         ; End of “START_bit
```

```
;*********************************************************************************************************
;
;                                        STOP CONDITION ON SMBus
;*********************************************************************************************************
;
;Name:          STOP_bit
;Function:      Generate STOP condition on SMBus
;Input:         No
;Output:        No
;Comments:   Refer to "System Management BUS(SMBus) specification Version 2.0" or
;                390119061402 application note for more information about SMBus
;                comunication with a MLX90614 module
;*********************************************************************************************************
;
STOP_bit

        _SCL_LOW                        ;Clear SCL line
        MOVLW           TBUF            ;
        CALL            delay           ; Wait a few microseconds
        _SDA_LOW                        ;Clear SDA line


        MOVLW           TBUF
        CALL            delay           ;Wait

        _SCL_HIGH                       ;Set SCL line
        MOVLW           TBUF            ;Stop condition setup time
        CALL            delay           ;(Tsu:sto=4.0us min)
        _SDA_HIGH                       ;Set SDA line

        RETURN                          ; End of  “STOP_bit”
```

```
;*************************************************************************************************
;                                  TRANSMIT DATA ON SMBus
;*************************************************************************************************
;Name:          TX_byte
;Function:      Send a byte on SMBus
;Input:         TX_buffer
;Output:        No
;Comments:      If receiver don't answer with ACK, the number of the attempts to be send will be
;               equal  of the value in Nack_Counter
;*************************************************************************************************
TX_byte
        LoadNACKcounter                 ; Set time out value
        MOVF        TX_buffer,W         ;
        MOVWF       TX_temp             ; Tx_buffer -> Tx_temp

TX_again
        MOVLW       D'8'
        MOVWF       Bit_counter         ; Load Bit_counter
tx_loop
        BCF         bit_out             ; 0 -> bit_out
        RLCF        TX_buffer,F         ; Tx_buffer<MSb> -> C
        BTFSC       STATUS,C            ; C is 0 or 1?        If C=0 don't set bit_out
        BSF         bit_out             ; 1 -> bit_out
        CALL        Send_bit            ;Send bit_out         on SDA line
        DECFSZ      Bit_counter,F       ; All 8th bits are sent? If not, send next bit ,else check for
                                        ; acknowledgement from the receiver
        GOTO        tx_loop             ; Send next bir
        CALL        Receive_bit         ; Check for acknowledgement from the receiver
        BTFSS       bit_in              ; If receiver has sent NACK stops the transmission
        RETURN                          ; End of "Tx_byte"

        CALL        STOP_bit            ; Stops transmission
        DECFSZ      Nack_Counter,F      ; Repeat transmission till Nack_Counter become 0
        GOTO        Repeat
        RETURN                          ; The receiver don't answer, stop the repeating
Repeat
        CALL        START_bit           ; Start transmission again
        MOVF        TX_temp,W           ;
        MOVWF       TX_buffer           ; Reload the sending byte in Tx_buffer again
        GOTO        TX_again            ; Send byte again

Send_bit
        BTFSC       bit_out             ; If bit_out=0 send 0 on SDA line
        GOTO        bit_high            ; else send 1 on SDA line
        _SDA_LOW                        ; Clear SDA line
        GOTO        clock
bit_high
         _SDA_HIGH                      ;Set SDA line
         NOP

clock
        _SCL_HIGH                       ; Set SLC line
```

```
NOP                         ;|
NOP                         ;|
NOP                         ;|
NOP                         ;|
NOP                         ;|
NOP                         ;|
NOP                         ; > This defines the high level of clock pulse!!!!!!!
NOP                         ;|
NOP                         ;|
NOP                         ;|
NOP                         ;|
NOP                         ;|
NOP                         ;|
_SCL_LOW                    ; Clear SCL line
NOP                         ;|
NOP                         ; > This defines the low level of clock pulse!!!!!!!!
NOP                         ;|
RETURN                      ; End of "Tx_byte"
```

```
;**************************************************************************************************
;                                   RECEIVE DATA ON SMBus
;**************************************************************************************************
;Name:          RX_byte
;Function:      Receive a byte on SMBus
;Input:         No
;Output:        RX_buffer(Received byte),bit_in(acknowledge bit)
;Comments:
;**************************************************************************************************
RX_byte
        CLRF            RX_buffer        ; Clear the receiving buffer
        MOVLW           D'8'             ;
        MOVWF           Bit_counter      ; Load Bit_counter
        BCF             STATUS,C         ;C=0
RX_again
        RLCF            RX_buffer,F      ; RX_buffer< MSb> -> C
        CALL            Receive_bit      ; Check bit on SDA line
        BTFSC           bit_in           ; If received bit is ‘1’ set RX_buffer<LSb>
        BSF             RX_buffer,0      ; Set RX_buffer<LSb>
        DECFSZ          Bit_counter,F    ; ALL 8th bis are received? If no receive next bit
        GOTO            RX_again         ; Receive next bit
        CALL            Send_bit         ;Send NACK or ACK
        RETURN                           ; End of “RX_byte”

Receive_bit
        BSF             bit_in           ; Set bit_in
        BSF             _SDA_IO          ; Make SDA-input
        _SCL_HIGH                        ; Set SCL line
        NOP                              ;|
        NOP                              ;|
        NOP                              ;|
        NOP                              ;|
        NOP                              ;|
        NOP                              ;|
        NOP                              ; > This defines the high level of clock pulse!!!!!!!
        NOP                              ;|
        NOP                              ;|
        NOP                              ;|
        NOP                              ;|
        NOP                              ;|
        NOP                              ;|
        BTFSS           _SDA             ; Read SDA line, if SDA=0 clear bit_in
        BCF             bit_in           ; Clear bit_in
        _SCL_LOW                         ; Clear SCL line
        NOP                              ;|
        NOP                              ; > This defines the low level of clock pulse!!!!!!!
        NOP                              ;|
        RETURN                           ; Bit is received
```

```
;*******************************************************************************************************
;                                          DELAY SUBROUTINE
;*******************************************************************************************************
;Name:          delay
;Function:      Produces time delay depending on the value in counterL
;Input:         WREG
;Output:        No
;Comments:
;*******************************************************************************************************
delay
        MOVWF           counterL        ; WREG -> counterL
        DECFSZ          counterL,f      ; If (counerL=counterL-1)  =0 go out
        BRA             $-2             ; else decrement counterL again
        RETURN                          ; End of "delay"
```

```
;****************************************************************************************************
;                                           CALCULATION CRC8
;****************************************************************************************************
;Name:         PEC_calculation
;Function:     Calculates the PEC of received bytes
;Input:        PEC4:PEC3:PEC2:PEC1:PEC0:PEC- data registers
;              CRC4:CRC3:CRC2:CRC1:CRC0:CRC- CRC value=00000107h
;Output:       PEC
;Comments:     Refer to 390119061402 application note for more information about SMBus
;              comunication with a MLX90614 module
;****************************************************************************************************
PEC_calculation
        MOVLW       0x07    ;|
        MOVWF       CRC     ;|
        MOVLW       0x01    ;|
        MOVWF       CRC0    ; > Load CRC value 0x0107
        CLRF        CRC1    ;|
        CLRF        CRC2    ;|
        CLRF        CRC3    ;|
        CLRF        CRC4    ;|

        MOVLW       d'47'
        MOVWF       BitPosition

;chech PEC4 for '1'
        BTFSC       PEC4,7
        BRA         shift_CRC
        DECF        BitPosition
        BTFSC       PEC4,6
        BRA         shift_CRC
        DECF        BitPosition
        BTFSC       PEC4,5
        BRA         shift_CRC
        DECF        BitPosition
        BTFSC       PEC4,4
        BRA         shift_CRC
        DECF        BitPosition
        BTFSC       PEC4,3
        BRA         shift_CRC
        DECF        BitPosition
        BTFSC       PEC4,2
        BRA         shift_CRC
        DECF        BitPosition
        BTFSC       PEC4,1
        BRA         shift_CRC
        DECF        BitPosition
        BTFSC       PEC4,0
        BRA         shift_CRC

;check PEC3 for '1'
        DECF        BitPosition
        BTFSC       PEC3,7
        BRA         shift_CRC
        DECF        BitPosition
        BTFSC       PEC3,6
        BRA         shift_CRC
        DECF        BitPosition
        BTFSC       PEC3,5
        BRA         shift_CRC
        DECF        BitPosition
        BTFSC       PEC3,4
        BRA         shift_CRC
        DECF        BitPosition
        BTFSC       PEC3,3
        BRA         shift_CRC
        DECF        BitPosition
        BTFSC       PEC3,2
        BRA         shift_CRC
        DECF        BitPosition
        BTFSC       PEC3,1
        BRA         shift_CRC
        DECF        BitPosition
        BTFSC       PEC3,0
        BRA         shift_CRC

;check PEC2 for '1'
        DECF        BitPosition
        BTFSC       PEC2,7
        BRA         shift_CRC
        DECF        BitPosition
        BTFSC       PEC2,6
        BRA         shift_CRC
        DECF        BitPosition
        BTFSC       PEC2,5
        BRA         shift_CRC
        DECF        BitPosition
        BTFSC       PEC2,4
        BRA         shift_CRC
        DECF        BitPosition
        BTFSC       PEC2,3
        BRA         shift_CRC
        DECF        BitPosition
        BTFSC       PEC2,2
        BRA         shift_CRC
        DECF        BitPosition
        BTFSC       PEC2,1
        BRA         shift_CRC
        DECF        BitPosition
        BTFSC       PEC2,0
        BRA         shift_CRC

;check PEC1 for '1'
        DECF        BitPosition
        BTFSC       PEC1,7
        BRA         shift_CRC
        DECF        BitPosition
        BTFSC       PEC1,6
        BRA         shift_CRC
        DECF        BitPosition
        BTFSC       PEC1,5
        BRA         shift_CRC
        DECF        BitPosition
        BTFSC       PEC1,4
        BRA         shift_CRC
        DECF        BitPosition
        BTFSC       PEC1,3
        BRA         shift_CRC
        DECF        BitPosition
        BTFSC       PEC1,2
        BRA         shift_CRC
        DECF        BitPosition
        BTFSC       PEC1,1
        BRA         shift_CRC
        DECF        BitPosition
        BTFSC       PEC1,0
        BRA         shift_CRC

;check PEC0 for '1'
        DECF        BitPosition
        BTFSC       PEC0,7
        BRA         shift_CRC
        DECF        BitPosition
        BTFSC       PEC0,6
        BRA         shift_CRC
        DECF        BitPosition
        BTFSC       PEC0,5
        BRA         shift_CRC
        DECF        BitPosition
        BTFSC       PEC0,4
        BRA         shift_CRC
        DECF        BitPosition
        BTFSC       PEC0,3
        BRA         shift_CRC
        DECF        BitPosition
        BTFSC       PEC0,2
        BRA         shift_CRC
        DECF        BitPosition
        BTFSC       PEC0,1
        BRA         shift_CRC
        DECF        BitPosition
        BTFSC       PEC0,0
        BRA         shift_CRC

        CLRF        PEC4
        CLRF        PEC3
        CLRF        PEC2
        CLRF        PEC1
        CLRF        PEC0
        RETURN

shift_CRC
        MOVLW       d'8'
        SUBWF       BitPosition,W       ; BitPosition-8 ->W
        MOVWF       shift               ; Get shift value for CRC registers
        BCF         STATUS,C

shift_loop
        MOVF        shift,F             ; Read shift to force flag Z
        BZ          xor
        RLCF        CRC,F
        RLCF        CRC0,F
        RLCF        CRC1,F
        RLCF        CRC2,F
        RLCF        CRC3,F
        RLCF        CRC4,F
        DECFSZ      shift,F
        BRA         shift_loop

xor
        MOVF        CRC4,W
        XORWF       PEC4,F
        MOVF        CRC3,W
        XORWF       PEC3,F
        MOVF        CRC2,W
        XORWF       PEC2,F
        MOVF        CRC1,W
        XORWF       PEC1,F
        MOVF        CRC0,W
        XORWF       PEC0,F
        MOVF        CRC,W
        XORWF       PEC,F
        BRA         PEC_calculation
```

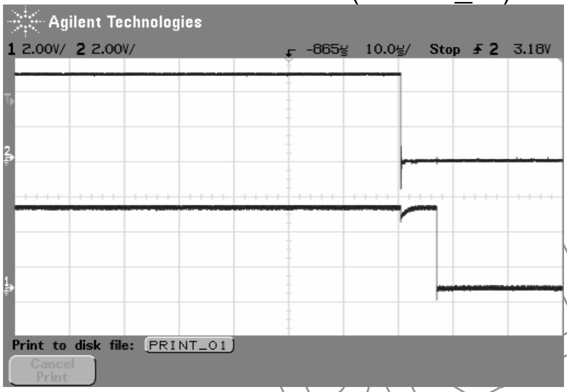
```
;*******************************************************************************************************
;                                            MACROS
;*******************************************************************************************************
LoadNACKcounter MACRO
                  MOVLW        D'255'             ; The value in Nack_Counter defines
                  MOVWF        Nack_Counter  ; time out if a device doesn't send ACK bit
                ENDM
;------------------------------------------------------------------------------------------------------
_SDA_HIGH       MACRO
                  BSF          _SDA_IO           ; _SDA-input, SDA line is high from  pull up
                ENDM
;------------------------------------------------------------------------------------------------------
_SCL_HIGH       MACRO
                  BSF          _SCL_IO           ;_SCL-input, SCL line is high from pull up
                ENDM
;------------------------------------------------------------------------------------------------------
_SDA_LOW        MACRO
                  BCF          _SDA              ;
                  BCF          _SDA_IO           ; Clear SDA line
                ENDM
;------------------------------------------------------------------------------------------------------
_SCL_LOW        MACRO
                  BCF          _SCL              ;
                  BCF          _SCL_IO           ;Clear SCL line
                 ENDM
```

## Asembly of everything together

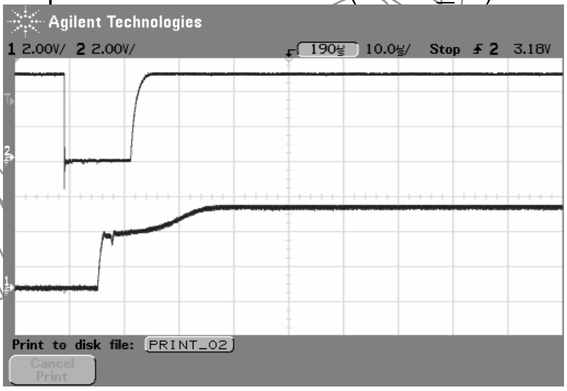
```
;*****************************************************************************************************
;                         Read MLX90614 RAM or EEPROM address subroutine
;*****************************************************************************************************
;Name:          MemRead
;Function:      Read specified RAM or EEPROM address of a MLX90614 module
;Input:         SlaveAddress, command=RAM_Address(EE_Address) |
;                RAM_Access(EE_Accsess)
;Output:        DataH:DataL
;Comments:      Refer to 390119061402 application note for more information about SMBus
;               comunication with a MLX90614 module
;*****************************************************************************************************
MemRead

        CALL            START_bit               ; Start condition

        MOVF            SlaveAddress,W          ;|
        MOVWF           TX_buffer               ; > Send SlaveAddress
        CALL            TX_byte                 ;|

        MOVF            command,W               ;|
        MOVWF           TX_buffer               ; > Send command
        CALL            TX_byte                 ;|

        CALL            START_bit               ; Repeat start condition

        MOVF            SlaveAddress,W          ;|
        MOVWF           TX_buffer               ; > Send Slave address
        CALL            TX_byte                 ;|


        BCF             bit_out                 ; bit_out=0 ( master will send ACK)
        CALL            RX_byte                 ; Receive low data byte
        MOVFF           RX_buffer,DataL         ; Save it in DataL

        BCF             bit_out                 ; bit_out=0 ( master will send ACK)
        CALL            RX_byte                 ; Receivehigh data byte
        MOVFF           RX_buffer,DataH         ; Save it in DataH

        BSF             bit_out                 ; bit_out=1 ( master will send NACK)
        CALL            RX_byte                 ; Receivehigh PEC
        MOVFF           RX_buffer,PecReg        ; Save it in PecReg

        CALL            STOP_bit                ; Stop condition

        MOVF            SlaveAddress,W          ;|
        MOVWF           PEC4                    ;|
        MOVFF           command,PEC3            ;|
        MOVF            SlaveAddress,W          ; > Load PEC3:PEC2:PEC1:PEC0:PEC
        MOVWF           PEC2                    ;|
        MOVFF           DataL,PEC1              ;|
        MOVFF           DataH,PEC0              ;|
        CLRF        PEC                   ;|

        CALL        PEC_calculation       ; Calculate CRC8, result is in PEC
        MOVF        PecReg,W              ;
        XORWF       PEC,W                 ; PEC xor PecReg ->WREG
        BTFSS       STATUS,Z              ; If PEC=PecReg go out
        GOTO        MemRead       ; Else repaet all transmission

        RETURN                            ; End of RamMemRead


;*****************************************************************************************************
;                                   MAIN PROGRAM
;*****************************************************************************************************
;Name:    MAIN
;Function: Demonstrates the steps for implementation of a full SMBus frame
:Input:
;Output:
;Comments:
;*****************************************************************************************************
MAIN

        MOVLW       SA                         ;
        MOVWF       SlaveAddress               ; SA -> SlaveAddress
        MOVLW       RAM_Address|RAM_Access     ; Form RAM access command + RAM
        MOVWF       command                    ; address

Readloop
        CALL        MemRead                    ; Read RAM address
        BRA         Readloop                   ; Read RAM address again

        END
```

## *Oscilograms*

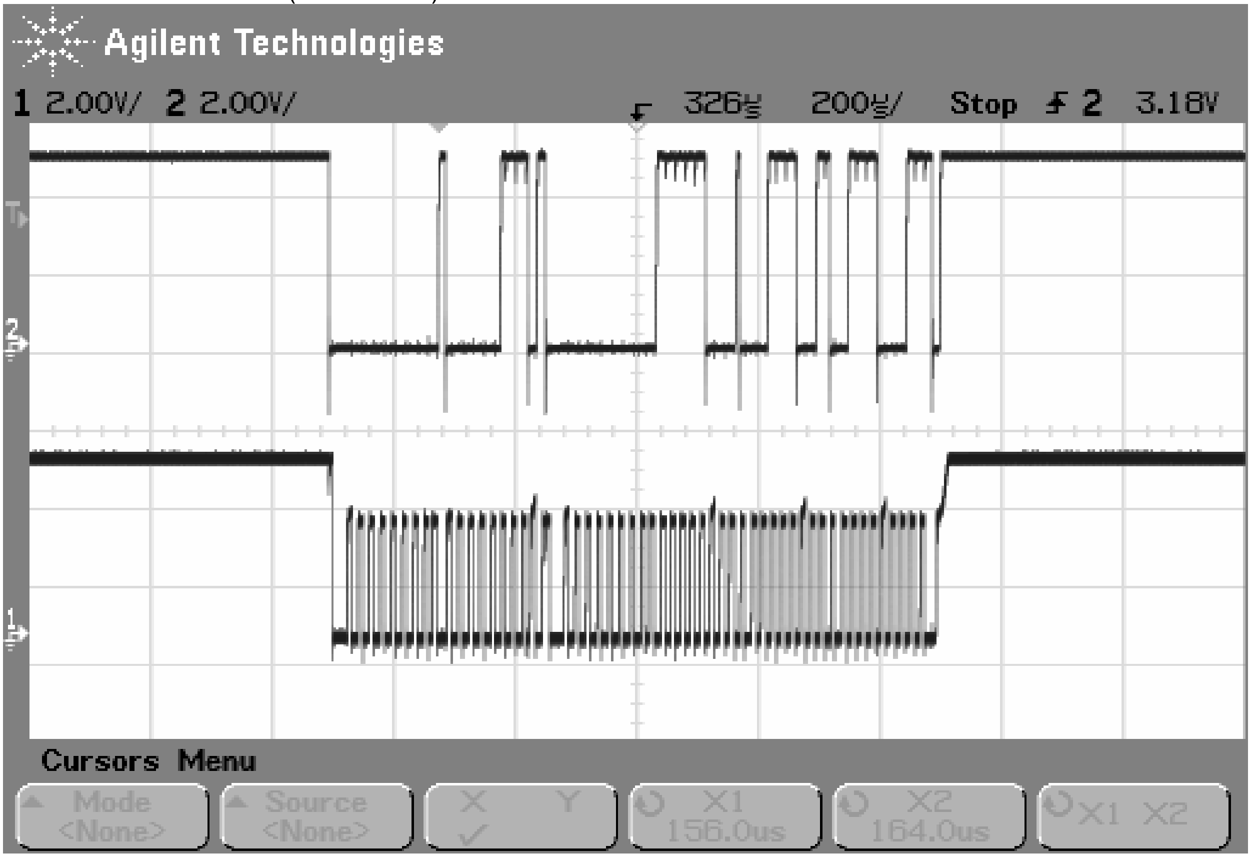
Start Condition on SMBus (START_bit)



Stop Condition on SMBus (STOP_bit)



Full frame on SMBus (MemRead)



Above each oscilogram is described what it is and in the parenthesis the subroutine which produces this oscilogram.

## Conclusion

The example of this application note demonstrates reading of RAM memory of a MLX90614 module but it can be used for EEPROM reading if instead RAM_Access command is used EEPROM_Access command (refer to 390119061402 application note).

♦APPENDIX – RAM memory map

| name | address |
|---|---|
| Melexis reserved | 0x00h |
|  | … |
| Melexis reserved | 0x02h |
| Ambient sensor data | 0x03h |
| IR sensor 1 data | 0x04h |
| IR sensor 2 data | 0x05h |
| Linearized ambient temperature Ta | 0x06h |
| Linearized object temperature (IR1) $T_{OBJ1}$ | 0x07h |
| Linearized object temperature (IR2) $T_{OBJ2}$ | 0x08h |
| Melexis reserved | 0x09h |
| $T_{A1}$ (PKI) | 0x0Ah |
| $T_{A2}$ (PKI) | 0x0Bh |
| Melexis reserved | 0x0Ch |
| Temporary register | 0x0Dh |
| Temporary register | 0x0Eh |
| Temporary register | 0x0Fh |
| Temporary register | 0x10h |
| Temporary register | 0x11h |
| Temporary register | 0x12h |
| Scale for ratio alpha ROM alpha real | 0x13h |
| Scale for alpha's slope versus object temperature | 0x14h |
| IIR filter | 0x15h |
| $T_{A1}$ (PKI) fraction | 0x16h |
| $T_{A2}$ (PKI) fraction | 0x17h |
| Temporary register | 0x18h |
| Temporary register | 0x19h |
| Temporary register | 0x1Ah |
| FIR filter | 0x1Bh |
| Temporary register | 0x1Ch |