Application Note

MLX90614 SMBus implementation in PIC MCU

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 MLX90614's RAM reading. Software implementation of SMBus communication is used so 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 and MCC18 (MPLAB® C18 COMPILER) for which an evaluation version is available 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.

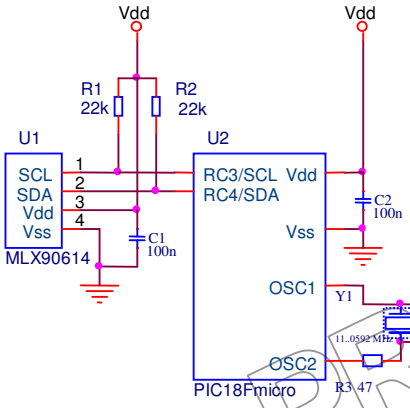
Other Components Needed

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 11.0592MHz
PIC18F4320 microcontroller or other from the Microchip's PIC18 family.

Accompanying files:

1. MPASM files to include in existing project, "SMBusFiles"
2. MPLAB assembler project
3. MPLAB C project

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. Refer to MLX90614 datasheets, AppNote, “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).

Used Conventions

	Key word
	Comment
	Directive
	Macro

PART I: ASSEMBLER EXAMPLE

Build and use

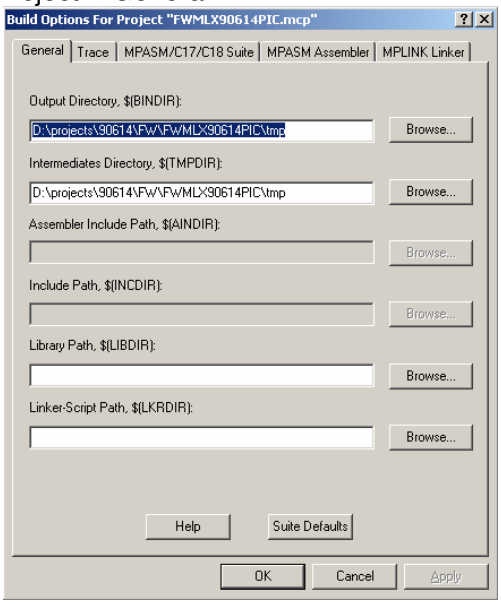
Below is the assembly language code. It consists of:

- 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

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

If the accompanying MPLAB project is used the next paths must be defined by the user in Project -> Build Options -> Project ->General



- “main.asm” file in both files and 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 consists of:

```
MOVLW   SA<<1                   ; Put Slave Address in the upper 7 bits
MOVWF   SlaveAddress            ; SA -> SlaveAddress
MOVLW   RAM_Address|RAM_Access  ; Form RAM access command + RAM
MOVWF   command                 ; address
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) 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<<1                 ; Put Slave Address in the upper 7 bits
MOVWF   SlaveAddress            ; SA -> SlaveAddress
MOVLW   0x06                    ; Form RAM access command + RAM
MOVWF   command                 ; address
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<<1                 ; Put Slave Address in the upper 7 bits
MOVWF   SlaveAddress            ; SA -> SlaveAddress
MOVLW   0x07                    ; Form RAM access command + RAM
MOVWF   command                 ; address
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
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 message again. The unsigned value in register Nack_Counter-1 defines how many time the message to be retransmitted in case that a module returns “not acknowledge” or a wrong PEC byte.
Register PEC contains CRC calculated for the entire communication SMBus frame. This value is comparing 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.

Assembler files description

main.asm

```
;*******************************************************************************************************
;                           BASIC INFORMATION ABOUT THIS PROGRAM
;*******************************************************************************************************
;File name:     main.asm
;Data:          21/09/2007
;Version:       02.00
;Company:       Melexis-Bulgaria(www.melexis.com)
;Description:   Software SMBus implementation for MLX90614 using PIC18F4320
;               Language: Microchip Assembler
;               Fosc=11.0592MHz, Tcy=362ns
;  Author:      Dimo petkov, dpv@melexis.com
;*******************************************************************************************************
;                           DEFINE MCU TYPE AND INCLUDE HEADER FILES
;*******************************************************************************************************
        LIST            p=18F4320
        #include        <p18F4320.inc>
        #include        "boot.inc"
        #include        "config.inc"
        #include        "GPRs.inc"
        #include        "macros.inc"
;*******************************************************************************************************
;                                   BEGINNING OF THE PROGRAM
;*******************************************************************************************************
TYPE_PICSTART   =1 ;Bootloader not used
TYPE_BOOTLOAD =2 ;Bootloader (http://members.rogers.com/martin.dubuc/Bootloader/)
                 ; is used
                 ;(http://mdubuc.freeshell.org/Colt/)

type_prog=2      ;EQU THE PROGRAMMING TOOL HERE! - results in vectors' offEQU
                 ; (e.g. reEQU 0x0000 <-> 0x0200)

IF type_prog == 2
        ORG 0x0000
        #include "BLOAD_PM.asm"  ;NB: EQU the last EEPROM address to 0xFF ! At the end
                                 ; of code...
        ORG 0x0200               ;revised in order to add bootloader,
                                 ;http://members.rogers.com/martin.dubuc/Bootloader/
ELSE
        ORG 0x0000
ENDIF

        GOTO        MAIN         ; Jump to main program
```

```
;*******************************************************************************************************
;                                        INTERRUPT SERVISE ROUTINES
;*******************************************************************************************************

        IF type_prog == 2
                ORG     0x00208
        ELSE
                ORG     0x00008
        ENDIF
        GOTO            isr_high


        IF type_prog == 2
                ORG     0x00218
        ELSE
                ORG     0x00018
        ENDIF
        GOTO            isr_low

;----------------------HIGH PRIORITY LEVEL INTERRUPT SERVISE ROUTINE--------------------------
isr_high
;Check for low interrupt source
;..............
        RETFIE          FAST                ;If interrupt source is different go out


;-----------------------LOW PRIORITY LEVEL INTERRUPT SERVISE ROUTINE-------------------------
isr_low

  MOVWF       WREG_temp               ;|
  MOVFF       STATUS,STATUS_temp      ; > Save STATUS, WREG and BSR registers in RAM
  MOVFF       BSR, BSR_temp           ;|

;Check for low interrupt source
;...........
  RETFIE                              ;If interrupt source is different go out

isr_low_end

  MOVFF       BSR_temp,BSR            ;|
  MOVF        WREG_temp,W             ; > Restore BSR,WREG,STATUS
  MOVFF       STATUS_temp,STATUS      ;|

  RETFIE
```

```
;*******************************************************************************************************
;                                          SUBROUTINES
;*******************************************************************************************************
#include "MemoryAccess.asm"
#include "SMBusSubr.asm"
#include "CRC8.asm"
#include "MCUinit.asm"
#include "delay.asm"

;*******************************************************************************************************
;                                            MAIN
;*******************************************************************************************************
MAIN
        CALL    MCUinit                 ; MCU initialization

        MOVLW   SA<<1                   ; Put SA in the upper 7 bits
        MOVWF   SlaveAddress            ; Set SMBus address
        MOVLW   RAM_Tobj1|RAM_Access    ; Form RAM access command + RAM address
        MOVWF   command                 ; Load the command register

        CALL    delay_200ms             ; Wait after POR,Tvalid=0.15s

ReadLoop

        CALL    MemRead                 ; Read RAM address
        CALL    delay_200ms             ;|
        CALL    delay_200ms             ; > Wait before next measurement
        CALL    delay_200ms             ;|
        BRA     ReadLoop                ; Read again


        END

;*********************************************END OF PROGRAM *****************************************
```

MCUinit.asm

```
;*******************************************************************************************************
;
                        SUBTITLE "PIC18F4320 initialization"
;*******************************************************************************************************
MCUinit
; I/O config
        MOVLW       B'00001111'
        MOVWF       ADCON1          ; All channels are digital I/O
;-------------------------------------------------------------------------------------------------------
;SMBus_init

        _SCL_HIGH                   ; The bus is in idle state
        _SDA_HIGH                   ; SDA and SCL are in high level from pull up resistors
;-------------------------------------------------------------------------------------------------------

        RETURN
```

SMBusSubr.asm

```
;*******************************************************************************************************
;                                        START CONDITION ON SMBus
;*******************************************************************************************************
;Name:          START_bit
;Function:      Generates START condition on SMBus
;Input:         No
;Output:        No
;Comments:      Refer to "System Management BUS(SMBus) specification Version 2.0" and
;               AN "SMBus communication with MLX90614"
;*******************************************************************************************************
START_bit

        _SDA_HIGH                       ; Set SDA line
        MOVLW       TBUF                ; Wait a few us
        CALL        delay               ;
        _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                        ; Clears SCL line
        MOVLW       TBUF                ;
        CALL        delay               ; Wait

        RETURN
;*******************************************************************************************************
;                                        STOP CONDITION ON SMBus
;*******************************************************************************************************
;Name:          STOPbit
;Function:      Generate STOP condition on SMBus
;Input:         No
;Output:        No
;Comments:      Refer to "System Management BUS(SMBus) specification Version 2.0" and
;               AN "SMBus communication with MLX90614"
;*******************************************************************************************************
STOP_bit

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

;*******************************************************************************************************
;                                   TRANSMIT DATA ON SMBus
;*******************************************************************************************************
;Name:          TX_byte
;Function:      Send a byte on SMBus
;Input:         TX_buffer
;Output:        WREG - contains acknowledge value, 0 for ACK and 1 for NACK
;Comments:
;*******************************************************************************************************
TX_byte
        MOVWF       TX_buffer
        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 bit
        CALL        Receive_bit     ; Check for acknowledgement from the receiver
        BTFSS       bit_in          ; Receiver send ACK?
        RETLW       0               ; Yes, return 0
        RETLW       1               ; No, return 1

;-------------------------------------------------------------------------------------------------------------------
Send_bit
        BTFSC       bit_out         ;|
        GOTO        bit_high        ;|
        _SDA_LOW                    ;|
        GOTO        clock           ; > Send bit on SDA line
bit_high                            ;|
        _SDA_HIGH                   ;|
        NOP                         ;|
clock
        _SCL_HIGH                   ;|
        NOP                         ;|
        NOP                         ;|
        NOP                         ;|
        NOP                         ;|
        NOP                         ;|
        NOP                         ;|
        NOP                         ;|
        NOP                         ; > Send clock pulse
        NOP                         ;|
```

```
        NOP                             ;|
        NOP                             ;|
        NOP                             ;|
        NOP                             ;|
        _SCL_LOW                        ;|
        NOP                             ;|
        NOP                             ;|
        RETURN

;*******************************************************************************************************
;                                   RECEIVE DATA ON SMBus
;*******************************************************************************************************
;Name:          RX_byte
;Function:      Receives 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                   ;|
        NOP                         ;|
        NOP                         ;|
        NOP                         ;|
        NOP                         ;|
        NOP                         ;|
        NOP                         ;|
        NOP                         ;|
        NOP                         ; > Send 9th clock and check for acknowledge from Slave
        NOP                         ;|
        NOP                         ;|
        NOP                         ;|
        NOP                         ;|
        NOP                         ;|
        BTFSS       _SDA            ;|
        BCF         bit_in          ;|
```

```
        _SCL_LOW                ;|
        NOP                     ;|
        NOP                     ;|
        RETURN                  ; Bit is received
;-------------------------------------------------------------------------------
```

CRC8.asm

```
;*******************************************************************************************************
;                                        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 "System Management BUS (SMBus) specification Version 2.0" or
;               refer to AN “SMBus communication with MLX90614” for more information about
;               SMBus communication 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

; check 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
```

delay.asm

```
;*******************************************************************************************************
;                                        TUNABLE DELAY
;*******************************************************************************************************
;Name:          delay
;Function:      Produces time delay depending on the value in counterL
;Input:         WREG
;Output:        No
;Comments:      Used in START _bit and STOP_bit subroutines to meet SMBus timing
;                requirements.
;                Refer to "System Management BUS (SMBus) specification Version 2.0" and
;                AN "SMBus communication with MLX90614"
;*******************************************************************************************************
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"

;*******************************************************************************************************
;                        FIXED DELAY 50ms@Fosc=11.0592MHz
;*******************************************************************************************************
;Name:          delay_50ms
;Function:      Produces time delay 50ms
;Input:         No
;Output:        No
;Comments:
;*******************************************************************************************************
delay_50ms
        MOVLW       d'7'
        MOVWF       counterU

        MOVLW       d'26'
        MOVWF       counterH

        MOVLW       d'252'
        MOVWF       counterL

        DECFSZ      counterL,F
        BRA         $-2
        DECFSZ      counterH,F
        BRA         $-d'10'
        DECFSZ      counterU,F
        BRA         $-d'18'

        RETURN
```

```
;*******************************************************************************************************
;                                FIXED DELAY 200ms@Fosc=11.0592MHz
;*******************************************************************************************************
;
;Name:          delay_200ms
;Function:      Produces time delay 200ms
;Input:         No
;Output:        No
;Comments:
;*******************************************************************************************************
;
delay_200ms
        MOVLW       d'28'
        MOVWF       counterU

        MOVLW       d'26'
        MOVWF       counterH

        MOVLW       d'252'
        MOVWF       counterL

        DECFSZ      counterL,F
        BRA         $-2
        DECFSZ      counterH,F
        BRA         $-d'10'
        DECFSZ      counterU,F
        BRA         $-d'18'

        RETURN
;-------------------------------------------------------------------------------------------------------------------
```

MemoryAccess.asm

```
;*******************************************************************************************************
;                       Read MLX90614 RAM or EEPROM address subroutine
;*******************************************************************************************************
;Name:          MemRead
;Function:      Reads 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 AN “SMBus communication with MLX90614” for more information about
;               SMBus communication with a MLX90614 module
;               If receiver doesn’t answer with ACK or wrong PEC is received, the number of the
;               attempts a message to be send will be equal of the unsigned value in
;               Nack_Counter-1
;*******************************************************************************************************
MemRead
        LoadNACKcounter                 ; Set Nack_Counter
restart
        CALL    STOP_bit                ; STOP SMBus communication
        DECF    Nack_Counter,F          ; If((Nack_Counter-1) == 0) stop transmission
        BNZ     start                   ; Else start transmission
        RETURN                          ; Go out

start
        CALL    START_bit               ; Start condition

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

        ANDLW   0x01                    ; W & 0x01 -> W
        BTFSS   STATUS,Z                ; If Slave acknowledge, continue
        GOTO    restart                 ; Else restart communication

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

        ANDLW   0x01                    ; W & 0x01 -> W
        BTFSS   STATUS,Z                ; If Slave acknowledge, continue
        GOTO    restart                 ; Else restart communication

        CALL    START_bit               ; Repeat start condition

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

        ANDLW   0x01                    ; W & 0x01 -> W
        BTFSS   STATUS,Z                ; If Slave acknowledge, continue
        GOTO    restart                 ; Else restart communication
```

```
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          ; Receive high data byte
MOVFF       RX_buffer,DataH  ; Save it in DataH

BSF         bit_out          ; bit_out=1 (master will send NACK)
CALL        RX_byte          ; Receive high 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  ; Calculates CRC8, result is in PEC
MOVF        PecReg,W         ;
XORWF       PEC,W            ; PEC xor PecReg ->WREG
BTFSS       STATUS,Z         ; If PEC=PecReg go out
GOTO        restart          ; Else repeat all transmission

RETURN                       ; End of RamMemRead
```

GPRs.inc

```
;*******************************************************************************************************
;                                   GPRs AND CONSTANTs DEFINITIONS
;*******************************************************************************************************



CBLOCK      H'00'

  WREG_temp         ; Storage memory of WREG in ISRs with low priority
  STATUS_temp       ; Storage memory of STATUS in ISRs with low priority
  BSR_temp          ; Storage memory of BSR in ISRs with low priority

  counterU          ;|
  counterH          ; > Registers used in delay subroutines
  counterL          ;|

  TX_buffer         ; Saves the byte which will be send on the SMBus
  Bit_counter       ; Register used in TX_byte and RX_byte subroutines
  RX_buffer         ; Save the byte received from the SMbus

  flagreg0          ; Defines user flags
  DataL             ; Contains the low data byte read from MLX90614
  DataH             ; Contains the high data byte read from MLX90614
  PecReg            ; Contains the PEC byte read from MLX90614

  SlaveAddress      ; Contains the device address
  command           ; Contains the SMBus command
  Nack_Counter      ; Defines the numbers of the attempts one message to be
                    ; sent in case of a failure

  PEC4              ;|
  PEC3              ;|
  PEC2              ;|
  PEC1              ;|
  PEC0              ;|
  PEC               ;|
  CRC4              ;|
  CRC3              ; > Registers used in CRC8.asm file
  CRC2              ;|
  CRC1              ;|
  CRC0              ;|
  CRC               ;|
  BitPosition       ;|
  shift             ;|

ENDC
```

```
; Delay constants
#define TBUF d'23'                  ; Defines SMBus timings (Tbuf- bus free time between Stop
                                    ; and Start condition)


; SMBus control signals
#define _SCL_IO      TRISC,3        ; Defines SCL pin direction
#define _SDA_IO      TRISC,4        ; Defines SDA pin direction
#define _SCL         PORTC,3        ; Defines SCL data pin
#define _SDA         PORTC,4        ; Defines SDA data pin

#define bit_out      flagreg0,0     ; Contains the bit that will be send on the SDA line
#define bit_in       flagreg0,1     ; Contains the bit that is received on the SDA lina

; MLX90614 definitions
#define  RAM_Access  0x00           ; Defines the MLX90614 command RAM_Accsess
#define  RAM_Tobj1   0x07           ; Defines address from MLX90614 RAM memory
#define  SA          0x00           ; Defines SMBus device address
```

boot.inc

```
;-------------------------------------------------------------------------------
;                          Reserves EEPROM for bootloader
;-------------------------------------------------------------------------------

        ORG 0xF000FE            ; PIC18F4320 has 256 bytes EEPROM

        DE  0x66, 0x44

;-------------------------------------------------------------------------------
```

config.inc

```
;*******************************************************************************************************
;                                          CONFIGURATION BITS
;*******************************************************************************************************
        CONFIG      OSC=HS,FSCM=OFF,IESO=ON
        CONFIG      PWRT=ON,BOR=OFF,BORV=42
        CONFIG      WDT=OFF
        CONFIG      MCLRE=ON,PBAD=DIG,CCP2MX=C1
        CONFIG      STVR=ON,LVP=OFF,DEBUG=OFF
        CONFIG      CP0=OFF,CP1=OFF,CP2=OFF,CP3=OFF
        CONFIG      CPB=ON,CPD=OFF
        CONFIG      WRT0=OFF,WRT1=OFF,WRT2=OFF,WRT3=OFF
        CONFIG      WRTC=OFF,WRTB=ON,WRTD=OFF
        CONFIG      EBTR0=OFF,EBTR1=OFF,EBTR2=OFF,EBTR3=OFF
        CONFIG      EBTRB=ON
```

macros.inc

```
;*******************************************************************************************************
;                                        MACROS
;*******************************************************************************************************
LoadNACKcounter    macro

                   MOVLW       D'0'
                   MOVWF       Nack_Counter

                   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     ; Put ‘0’ on _SDA line

                   endm
;-------------------------------------------------------------------------------------------------------
_SCL_LOW           macro

                   BCF         _SCL        ;
                   BCF         _SCL_IO     ; Put ‘0’ on _SLA line

                   endm
```

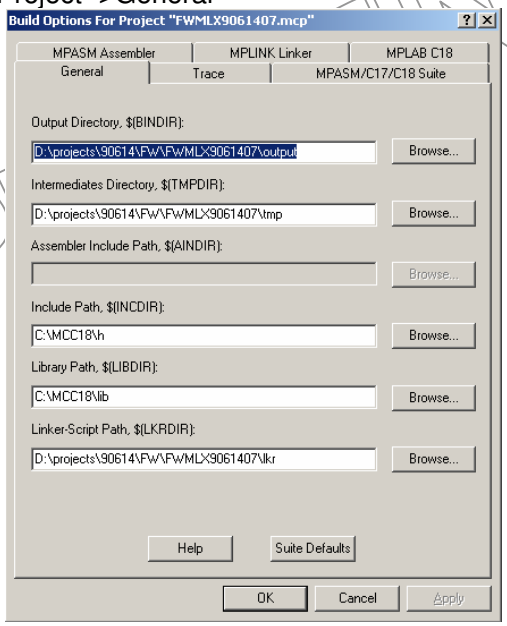
PART I I: C EXAMPLE

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

If the accompanying MPLAB project is used the next paths must be defined by the user in Project -> Build Options -> Project ->General



- The files main.c, SMBus.c, CRC8.c, delay.c

The code which reads MLX90614 consists of:

```
unsigned char   SlaveAddress;          //Contains device address
unsigned char   command;               //Contains the access command
unsigned int    data;                  //Contains data value

SlaveAddress=SA<<1;                    //Set device address
command=RAM_Access|RAM_Tobj1;          //Form RAM access command + RAM address
data=MemRead(SlaveAddress,command);    //Read memory
```

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) 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:

```
SlaveAddress=0x5A<<1;                 //Set device address
command=0x06;                         //Form RAM access command + RAM address
data=MemRead(SlaveAddress,command);   //Read memory
```

The data 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:

```
SlaveAddress=0x5A<<1;                 //Set device address
command=0x07;                         //Form RAM access command + RAM address
data=MemRead(SlaveAddress,command);   //Read memory
```

Output temperature format will be the same, for example,
data 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:

```
SlaveAddress=0x00;                    //Set device address
command=0x2E;                         // Form EEPROM access command + EEPROM address
data=MemRead(SlaveAddress,command);   //Read EEPROM memory
```

The Slave Address (read from EEPROM) would be on data. 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 message again. The unsigned value in register ErrorCounter – 1 defines how many time the message to be retransmitted in case that a module returns “not acknowledge” or a wrong PEC byte.

The variable PecReg contains CRC calculated for the entire communication SMBus frame. This value is comparing 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.

C files description

main.c

```
//*****************************************************************************************************************
//                              BASIC INFORMATION ABOUT THIS PROGRAM
//*****************************************************************************************************************
//File name:    main.c
//Data:         19/09/2007
//Version:      01.00
//Company:      Melexis-Bulgaria(www.melexis.com)
//Details:      SMBus comunication with MLX90614 using PIC18F4320
//              Language C: MCC18 compiler
//              Fosc=11.0592MHz, Tcy=362ns
//Author:       Dimo Petkov( dpv@melexis.com)
//*****************************************************************************************************************
//                                          HEADER FILES
//*****************************************************************************************************************
#include <p18F4320.h>
#include "main.h"

//*****************************************************************************************************************
//                                          CONFIGURATION BITS
//*****************************************************************************************************************
#pragma       config  OSC=HS,FSCM=OFF,IESO=ON
#pragma       config  PWRT=ON,BOR=OFF,BORV=42
#pragma       config  WDT=OFF
#pragma       config  MCLRE=ON,PBAD=DIG,CCP2MX=C1
#pragma       config  STVR=ON,LVP=OFF,DEBUG=OFF
#pragma       config  CP0=OFF,CP1=OFF,CP2=OFF,CP3=OFF
#pragma       config  CPB=ON,CPD=OFF
#pragma       config  WRT0=OFF,WRT1=OFF,WRT2=OFF,WRT3=OFF
#pragma       config  WRTC=OFF,WRTB=ON,WRTD=OFF
#pragma       config  EBTR0=OFF,EBTR1=OFF,EBTR2=OFF,EBTR3=OFF
#pragma       config  EBTRB=ON
//*****************************************************************************************************************
//                                          VECTORS REMAPING
//*****************************************************************************************************************
#ifdef BOOTLOADER
#pragma code _HIGH_INTERRUPT_VECTOR = 0x000208
void _high_ISR (void)
{
   _asm GOTO high_isr _endasm ;
}
#pragma code _LOW_INTERRUPT_VECTOR = 0x000218
void _low_ISR (void)
{
   _asm GOTO low_isr _endasm ;
}
#pragma code
#endif
```

```
//***************************************************************************************************************
//                                          INTERRUPTS
//***************************************************************************************************************
#pragma interrupt high_isr
void high_isr(void)
{
// User code...
}


#pragma interruptlow low_isr
void low_isr(void)
{
// User code...
}

#pragma code
//***************************************************************************************************************
//                                          MAIN PROGRAM
//***************************************************************************************************************
//Name:         main()
//Function:     Demonstrates the steps for implementation of a full SMBus frame
//Parameters:
//Returnt:
//Comments:
;***************************************************************************************************************
void main(void)
{
        unsigned char       SlaveAddress;   //Contains device address
        unsigned char       command;        //Contains the access command
        unsigned int        data;           //Contains data value
        float               t               //Contains the calculated temperature

        MCUinit();                          //MCU initialization
        SlaveAddress=SA<<1;                 //Set device address
        command=RAM_Access|RAM_Tobj1;       //Form RAM access command + RAM address

//      SendRequest();                      //Switch to SMBus mode - this is need if module is
                                            // in PWM mode only
//      DummyCommand(SlaveAddress);         //This is need if Request Command is sent even
                                            //when the module is in SMBus mode
        delay(DEL200ms);                    //Wait after POR,Tvalid=0.15s

        while(1)
        {
                data=MemRead(SlaveAddress,command); //Read memory
                t=CalcTemp(data);                   //Calculate temperature
                delay(DEL1SEC);                     //Wait 1 second
        }

}/* End of main() */
```

```
//*****************************************************************************************************************
//                                        FUNCTION'S DEFINITIONS
//*****************************************************************************************************************
void MCUinit(void)
{
//IO setting-up
        ADCON1=0b00001111;          //All chanels are digital I/O

//SMBus setting-up
        mSDA_HIGH();                //The bus is in idle state
        mSCL_HIGH();                //SDA and SCL are in high level from pull up resitors
}/* End of init() */
//-----------------------------------------------------------------------------------------------------------------
void SendRequest(void)
{
        mSCL_LOW();                 //SCL 1 _____________|<-----80ms------>|_____________
        delay(DEL80ms);             //       0           |_____________|
        mSCL_HIGH();
}
//-----------------------------------------------------------------------------------------------------------------
void DummyCommand(unsigned char byte)
{
        START_bit();                //Start condition
        TX_byte(byte);              //Send Slave Address or whatever,no need ACK checking
        STOP_bit();                 //Stop condition
}
//-----------------------------------------------------------------------------------------------------------------
float CalcTemp(unsigned int value)
{
        float temp;

        temp=(value*0.02)-273.15;

        return temp;
}
```

```
//*************************************************************************************************************
//                      Read MLX90614 RAM or EEPROM address subroutine
//*************************************************************************************************************
//Name:         MemRead
//Function:     Reads specified RAM or EEPROM address of a MLX90614 module
//Parameters    unsigned char SlaveAddress,
//              unsigned char  command=RAM_Address(EE_Address) |
//                                     RAM_Access(EE_Accsess)
//Return:       unsigned int data
//Comments:     Refer to AN "SMBus communication with MLX90614" for more information about
//              SMBus comunication with a MLX90614 module
//              If receiver doesn't answer with ACK or wrong PEC is received, the number of the
//              attempts a message to be send will be equal of the unsigned value in
//              ErrorCounter-1
//*************************************************************************************************************
unsigned int MemRead(unsigned char SlaveAddress,unsigned char command)
{
        unsigned int    data;           // Data storage (DataH:DataL)
        unsigned char Pec;              // PEC byte storage
        unsigned char DataL;            // Low data byte storage
        unsigned char DataH;            // High data byte storage
        unsigned char arr[6];           // Buffer for the sent bytes
        unsigned char PecReg;           // Calculated PEC byte storage
        unsigned char ErrorCounter;     // Defines the number of the attempts for communication
                                        //  with MLX90614

        ErrorCounter=0x00;                      // Initialising of ErrorCounter

        do{
        repeat:
                STOP_bit();                     //If slave send NACK stop comunication
                --ErrorCounter;                 //Pre-decrement ErrorCounter
                if(!ErrorCounter){              //ErrorCounter=0?
                        break;                  //Yes,go out from do-while{}
                }
                START_bit();                    //Start condition

                if(TX_byte(SlaveAddress)){      //Send SlaveAddress
                        goto    repeat;         //Repeat comunication again
                }

                if(TX_byte(command)){           //Send command
                        goto    repeat;         //Repeat comunication again
                }
                START_bit();                    //Repeated Start condition

                if(TX_byte(SlaveAddress)){      //Send SlaveAddress
                        goto    repeat;         //Repeat comunication again
                }

                DataL=RX_byte(ACK);             //Read low data,master must send ACK
                DataH=RX_byte(ACK);             //Read high data,master must send ACK
```

```
            Pec=RX_byte(NACK);          //Read PEC byte, master should send NACK
            STOP_bit();                 //Stop condition


            arr[5]=SlaveAddress;         //
            arr[4]=command;              //
            arr[3]=SlaveAddress;         //Load array arr
            arr[2]=DataL;                //
            arr[1]=DataH;                //
            arr[0]=0;                    //
            PecReg=PEC_calculation(arr);//Calculate CRC

        }while(PecReg != Pec);          //If received and calculated CRC are equal go out
                                        //from do-while{}


        *((unsigned char *)(&data))=DataL;      //
        *((unsigned char *)(&data)+1)=DataH ;//data=DataH:DataL

        return data;
}
```

SMBus.c

```
//***************************************************************************************************************
//                                          HEADER FILES
//***************************************************************************************************************
#include "SMBus.h"
#include <delays.h>
//***************************************************************************************************************
//                                    START CONDITION ON SMBus
//***************************************************************************************************************
//Name:         START_bit
//Function:     Generates START condition on SMBus
//Parameters:   No
//Return:       No
//Comments:     Refer to "System Managment BUS(SMBus) specification Version 2.0"
//              or AN"SMBus communication with MLX90614" on the website www.melexis.com
//***************************************************************************************************************
void START_bit(void)
{
        mSDA_HIGH();                    // Set SDA line
        Delay10TCYx( TBUF );            // Wait a few microseconds
        mSCL_HIGH();                    // Set SCL line
        Delay10TCYx( TBUF );            // Generate bus free time between Stop
                                        // and Start condition (Tbuf=4.7us min)
        mSDA_LOW();                     // Clear SDA line
        Delay10TCYx( TBUF );            // Hold time after (Repeated) Start
                                        // Condition. After this period, the first clock is generated.
                                        //(Thd:sta=4.0us min)
        mSCL_LOW();                     // Clear SCL line
        Delay10TCYx( TBUF );            // Wait a few microseconds
}
//***************************************************************************************************************
//                                    STOP CONDITION ON SMBus
//***************************************************************************************************************
//Name:         STOPbit
//Function:     Generates STOP condition on SMBus
//Parameters:   No
//Return:       No
//Comments:     Refer to "System Managment BUS(SMBus) specification Version 2.0"
//               or AN"SMBus communication with MLX90614" on the website www.melexis.com
//***************************************************************************************************************
void STOP_bit(void)
{
        mSCL_LOW();                     // Clear SCL line
        Delay10TCYx( TBUF );            // Wait a few microseconds
        mSDA_LOW();                     // Clear SDA line
        Delay10TCYx( TBUF );            // Wait a few microseconds
        mSCL_HIGH();                    // Set SCL line
        Delay10TCYx( TBUF );            // Stop condition setup time(Tsu:sto=4.0us min)
        mSDA_HIGH();                    // Set SDA line
}
```

```
//*****************************************************************************************************************
//                                          TRANSMIT DATA ON SMBus
//*****************************************************************************************************************
//Name:         TX_byte
//Function:     Sends a byte on SMBus
//Parameters:   TX_buffer ( the byte which will be send on the SMBus )
//Return:       Ack_bit         ( acknowledgment bit )
//Comments:     Sends MSbit first
//*****************************************************************************************************************
unsigned char TX_byte(unsigned char Tx_buffer)
{
        unsigned  char          Bit_counter;
        unsigned  char          Ack_bit;
        unsigned  char          bit_out;

        for(Bit_counter=8; Bit_counter; Bit_counter--)
        {
                if(Tx_buffer&0x80) bit_out=1; // If the current bit of Tx_buffer is 1 set bit_out
                else               bit_out=0; // else clear bit_out

                send_bit(bit_out);              // Send the current bit on SDA
                Tx_buffer<<=1;                  // Get next bit for checking
        }

        Ack_bit=Receive_bit();                  // Get acknowledgment bit

        return  Ack_bit;
}// End of TX_bite()

//-----------------------------------------------------------------------------------------------------------------
void send_bit(unsigned char bit_out)
{
        if(bit_out==0) {mSDA_LOW();}
        else           {mSDA_HIGH();}
        Nop();                                  //
        Nop();                                  // Tsu:dat = 250ns minimum
        Nop();                                  //
        mSCL_HIGH();                            // Set SCL line
        Delay10TCYx( HIGHLEV );                 // High Level of Clock Pulse
        mSCL_LOW();                             // Clear SCL line
        Delay10TCYx( LOWLEV );                  // Low Level of Clock Pulse
//      mSDA_HIGH();                            // Master release SDA line ,
        return;
}//End of send_bit()
//-----------------------------------------------------------------------------------------------------------------
unsigned char Receive_bit(void)
{
        unsigned char           Ack_bit;

        _SDA_IO=1;                              // SDA-input
        mSCL_HIGH();                            // Set SCL line
        Delay10TCYx( HIGHLEV );                 // High Level of Clock Pulse
```

```
        if(_SDA)        Ack_bit=1;              // \ Read acknowledgment bit, save it in Ack_bit
        else            Ack_bit=0;              // /
        mSCL_LOW();                             // Clear SCL line
        Delay10TCYx( LOWLEV );                  // Low Level of Clock Pulse

        return  Ack_bit;
}//End of Receive_bit


//*************************************************************************************************************
//                                        RECEIVE DATA ON SMBus
//*************************************************************************************************************
//Name:         RX_byte
//Function:     Receives a byte on SMBus
//Parameters:   ack_nack (ackowlegment bit)
//Return:       RX_buffer(Received byte)
//Comments:     MSbit is received first
//*************************************************************************************************************
unsigned char RX_byte(unsigned char ack_nack)
{
        unsigned char           RX_buffer;
        unsigned char           Bit_Counter;

        for(Bit_Counter=8; Bit_Counter; Bit_Counter--)
        {
                if(Receive_bit())               // Get a bit from the SDA line
                {
                        RX_buffer <<= 1;        // If the bit is HIGH save 1  in RX_buffer
                        RX_buffer |=0b00000001;
                }
                else
                {
                        RX_buffer <<= 1;        // If the bit is LOW save 0 in RX_buffer
                        RX_buffer &=0b11111110;
                }
        }

        send_bit(ack_nack);                     // Sends acknowledgment bit

        return RX_buffer;
}
//-------------------------------------------------------------------------------------------------------------
```

delay.c

```
//***************************************************************************************************************
//                                    HEADER FILES
//***************************************************************************************************************
#include "delay.h"


//***************************************************************************************************************
//                                    TUNABLE      DELAY SUBROUTINE
//***************************************************************************************************************
//Name:          delay
//Function:      Generates delay time
//Parameters:  unsigned long i- defines the delay time
//Return:        No
//Comments:
//***************************************************************************************************************

void delay( unsigned long i)
{
        for(;i;i--);
}/* End of delay() */
```

CRC8.c

```
//*************************************************************************************************************
//                                  CALCULATION PEC PACKET
//*************************************************************************************************************
//Name:         PEC_calculation
//Function:     Calculates the PEC of received bytes
//Parameters:   unsigned char pec[]
//Return:       pec[0]-this byte contains calculated crc value
//Comments:     Refer to "System Managment BUS(SMBus) specification Version 2.0" and
//              AN "SMBus comunication with MLX90614"
//*************************************************************************************************************
unsigned char PEC_calculation(unsigned char pec[])
{
        unsigned char           crc[6];
        unsigned char           BitPosition=47;
        unsigned char           shift;
        unsigned char           i;
        unsigned char           j;
        unsigned char           temp;

        do{
                crc[5]=0;                               /* Load CRC value 0x000000000107 */
                crc[4]=0;
                crc[3]=0;
                crc[2]=0;
                crc[1]=0x01;
                crc[0]=0x07;
                BitPosition=47;                         /* Set maximum bit position at 47 */
                shift=0;

                //Find first 1 in the transmited message
                i=5;                                    /* Set highest index */
                j=0;
                while((pec[i]&(0x80>>j))==0 && i>0){
                        BitPosition--;
                        if(j<7){
                                j++;
                        }
                        else{
                                j=0x00;
                                i--;
                        }
                }/* End of while  */

                shift=BitPosition-8;    /*Get shift value for crc value*/


                //Shift crc value
                while(shift){
                        for(i=5; i<0xFF; i--){
                                if((crc[i-1]&0x80) && (i>0)){
                                        temp=1;
```

```
                        }
                        else{
                                temp=0;
                        }
                        crc[i]<<=1;
                        crc[i]+=temp;
                }/ *End of for */
                shift--;
        }/*End of while*/


        //Exclusive OR between pec and crc
        for(i=0; i<=5; i++){
                pec[i] ^=crc[i];
        }/* End of for */
    }while(BitPosition>8);/*End of do-while*/

    return pec[0];
}/* End of PEC_calculation */
```

Header files

main.h

```
/* Uncomment this if push pull clock need to be used */
//#define PUSH_PULL


/*  If bootloader is not desired do next:
        1. Comment #define BOOTLOADER
        2. Remove from the project BLOAD_PM.asm file
        3. In c018i.c file replace "#pragma code _entry_scn=0x000200"
                                   with   "#pragma code _entry_scn=0x000000"
        4. Replace rm18F4320i.lkr file with 18F4320i.lkr
*/
#define BOOTLOADER

// Set bit in a variable
#define bit_set(var,bitno) ((var) |= 1 << (bitno))
// Clear bit in a variable
#define bit_clr(var,bitno) ((var) &= ~(1 << (bitno)))
// Test bit in a variable
#define testbit(data,bitno) ((data>>bitno)&0x01)

//SMBus control signals
#define _SCL_IO TRISCbits.TRISC3
#define _SDA_IO TRISCbits.TRISC4
#define _SCL    PORTCbits.RC3
#define _SDA    PORTCbits.RC4

#define mSDA_HIGH()         _SDA_IO=1;
#define mSDA_LOW()  _SDA=0;_SDA_IO=0;

#ifndef PUSH_PULL
        #define mSCL_HIGH()         _SCL_IO=1;
#else
        #define mSCL_HIGH()         _SCL=1;_SCL_IO=0;
#endif

#define mSCL_LOW()  _SCL=0;_SCL_IO=0;

#define  ACK     0
#define  NACK    1

//MLX90614 constants
#define SA                  0x00    // Slave address
#define RAM_Access          0x00    // RAM access command
#define EEPROM_Access       0x20    // EEPROM access command
#define RAM_Tobj1           0x07    // To1 address in the eeprom
```

```
//Delay constants @ 11.0592MHz
#define DEL1SEC      100000
#define DEL80ms        7400
#define DEL200ms      18500
#define TBUF              2
//*PROTOTYPES*****************************************************************************************
void high_isr(void);
void low_isr(void);
void MCUinit(void);
unsigned int MemRead(unsigned char SlaveAddress,unsigned char command);
void SendRequest(void);
void DummyCommand(unsigned char byte);
float CalcTemp(unsigned int value);
//*EXTERNAL FUNCTIONS*********************************************************************************
extern  void START_bit(void);
extern  void STOP_bit(void);
extern  unsigned char TX_byte(unsigned char Tx_buffer);
extern  unsigned char RX_byte(unsigned char ack_nack);
extern  void delay( unsigned long i);
extern  unsigned char PEC_calculation(unsigned char pec[]);
```

SMBus.h

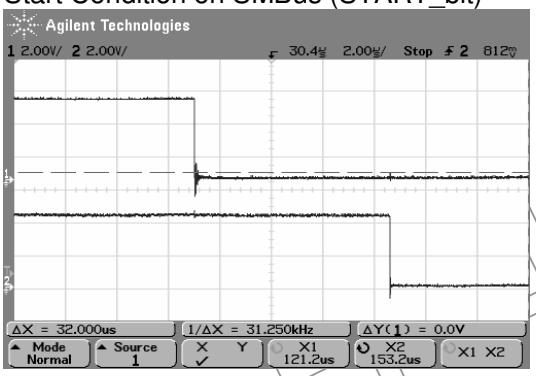
```
//*HEADER FILES*****************************************************************************************
#include <p18F4320.h>
#include "main.h"

//*High and Low level of the clock
#define HIGHLEV      3
#define LOWLEV       1

//*PROTOTYPES*******************************************************************************************
void START_bit(void);
void STOP_bit(void);
unsigned char TX_byte(unsigned char Tx_buffer);
unsigned char RX_byte(unsigned char ack_nack);
void send_bit(unsigned char bit_out);
unsigned char Receive_bit(void);
//*EXTERNAL FUNCTION************************************************************************************
extern  void delay( unsigned long i);
```

delay.h

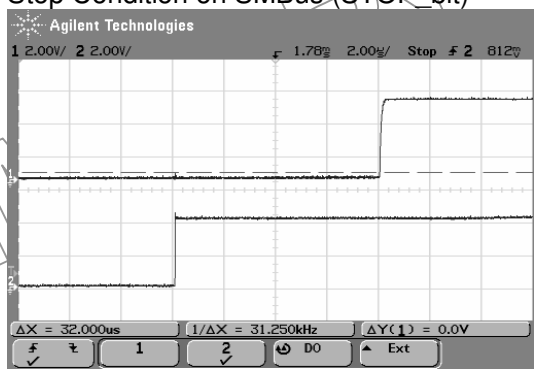
```
//*HEADER FILES*****************************************************************************************
#include <p18F4320.h>
#include "main.h"

//*PROTOTYPES*******************************************************************************************
void delay( unsigned long i);
```

Oscilograms

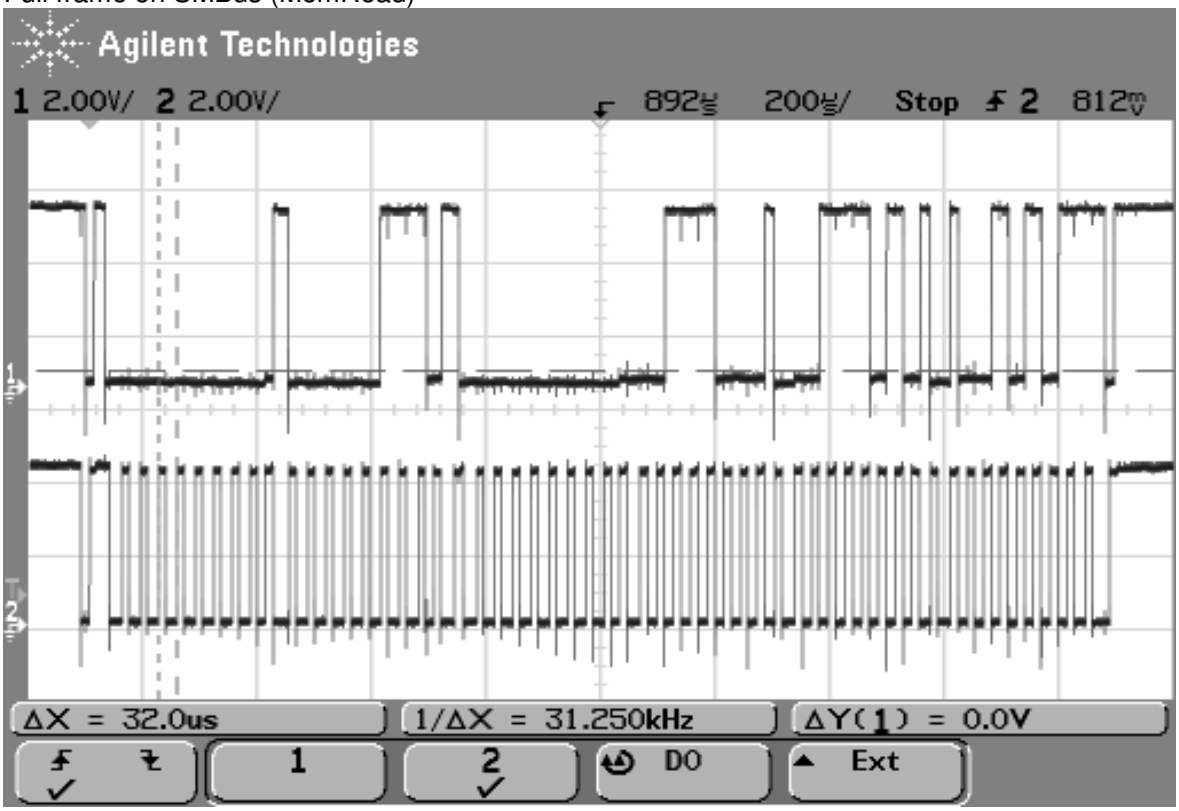
Start Condition on SMBus (START_bit)



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 AN “SMBus communication with MLX90614”).

♦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