Application Note

Simple IR temperature reader with MLX90614 and PIC10 MCU

Scope

This application note describes how to implement SMBus communication with MLX90614 Infra-Red thermometers. Code is in assembly language for Microchip's PIC®10. The example is read from MLX90614's RAM, measured temperatures. 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.

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

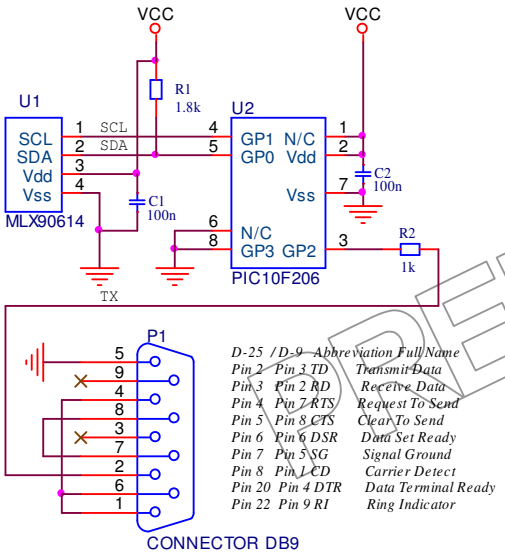
Elements used in the schematics within current application note include:
SMD ceramic capacitors C1 and C2 100nF 16V or higher.
SMD Resistors R1 1.8kOhm 5% and R2 1 kOhm 5%.
PIC10F206 or PIC10F202 microcontroller.
DB9 female conector.
Regulated (3 or 5V for 3 or 5V version of MLX90614) power source.

Accompanying files:
MPASM files to include in existing project, "SMBusFiles"
MPLAB project, "SMBusProject"
Project is built, file "main.hex" can be programmed in a PIC10F206. Also, project can be used as a "start with" base.
As provided the project will read Ta,Tobj1 and Tobj2 from MLX90614 (power supply control is not included), and transmit it via software UART (ASCII coded, CR (0x0D) after each cycle, 8 bit data, one stop bit, no parity bit, 57 600 baud with the 4.000 MHz internal oscilator 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. The read-and-transmit cycle is repeated every minute. During the idle state of the cycle both PIC and MLX90614 are in sleep mode (note that the sleep mode in 5V MLX90614 is partial, typical power drain is 100uA and the PIC10 is waken up every 2 seconds for short time).

Application Note

Simple IR temperature reader with MLX90614 and PIC10 MCU

Typical Circuit



Explanation

The connection of MLX90614 to MCU is very simple. Two general purpose pins GP0 and GP1of the PIC10F206 are used. One pull up resistor R1 is connected between Vdd and SDA line, SCL line is driven by a push-pull output GP0. 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. Schematic is given for DIL8 package pinout of the PIC10 MCU.

Series output resistor R2 protects the output against short curcuit.

On-chip 4MHz factory calibrated RC oscillators is used. SMBus clock is 28 kHz and one read frame takes about 11 ms (One frame is read of Tobj1, Tobj2 and Tamb and their transmission via UART). Refer to MLX90614 datasheets, AppNote 390119061402, “SMBus communication with MLX90614” and SMBus specification for details. MLX90614 comes in 5V and 3V versions. PIC10F206 could be used with both the 3V version (MLX90614Bxx) and 5V version (MLX90614Axx). Project can be compiled for PIC10F202, too (see below).

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:          delay_30ms
;Function:      Produces fixed delay 30ms
;Name:          delay_2ms
;Function:      Produces fixed delay 2ms
;Name:          SendRequest
;Function:      Switch module in SMBus mode
;Name:          hex2asc
;Function:      Convert a byte in ASCII code
```

;Name: pic10_uart
;Function: Send a byte by UART
Macros definitions
"Asembly of everything together" – main program

Build and use
Accompanying project can be directly used (main code file is "main.asm"). Files from the project can also be integrated in a new or existing project.
For details about MLX90614 refer to the MLX90614 Data Sheet available at www.melexis.com

Code that reads MLX90614 then consists of:

```
MOVLW SA<<1                     ; Slave address occupy MSB<7:1>
MOVWF SlaveAddress              ; SA -> SlaveAddress
MOVLW RAM_Address|RAM_Access    ; Form RAM access command + RAM
MOVWF command                   ; address

MemRead                         ; Read RAM address macro
```

RAM_Address can be Ta,To1 or To2. Result will be in DataH:DataL.

Factory default SMBus Slave Address (SA) for all MLX90614 is 0x5A. All MLX90614 devices also accept SA 0x00. Note that SA 0x00 will be useless in a network with more than one MLX90614 device.

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                   ; Slave address occupy MSB<7:1>
MOVWF SlaveAddress              ; SA -> SlaveAddress
MOVLW 0x06                      ; Form RAM access command + RAM
MOVWF command                   ; address

MemRead                         ; Read RAM address macro
```

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                ; Slave address occupy MSB<7:1>
MOVWF SlaveAddress           ; SA -> SlaveAddress
MOVLW 0x07                   ; Form RAM access command + RAM
MOVWF command                ; address

MemRead                      ; Read RAM address macro
```

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

MemRead                      ; Read EEPROM address macro
```

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 provided with this Application Note handles these in the following manner:
When a module returns “not acknowledge” then the communication frame is restarted. The value in register Nack_Counter defines how many times the communication frame is restarted in case that a module returns “not acknowledge”. If this counter overflows the program will continue with the next communication frame.
Packet Error Code check is not supported in this application.

Both the PIC MCU and the MLX90614 are in SLEEP mode most of the time. 3V MLX90614 has a typical sleep power drain of 2.5 µA and the PIC10 Watch-dog timer drains virtually the same current. During the on period the power drain is typically less than 3 mA. Thus, the average power drain is about 6 µA.

Sleep mode entry can be skipped and the cycle be made continuous read-and-transmit.

- conditional assembly for PIC10F202
  If PIC10F202 is used uncomment the row #define PIC10F202 (see full project)
- conditional assembly for continous cycle (without sleep mode)
  If sleep function is not need comment the row #define SLEEPON (see full project)

SMBus subroutines used for communication with MLX90614

```
;*******************************************************************************************************
;                                   GPRs AND CONSTANTs DEFINITIONs
;*******************************************************************************************************

                CBLOCK      H'00'
                        Nack_Counter
                         WDTcounter
                        TX_buffer
                        TX_temp
                        Bit_counter
                        flagreg0
                        RX_buffer
                        counterL
                        counterH
                        counterU
                        SlaveAddress
                        command
                        DataL
                        DataH
                        PecReg
                        tx_0
                        tx_1
                        digit1
                        digit2
                ENDC

;constants
#define TBUF            d'2'     ; Define delays(see SMBusSubr.asm)
#define BAUDRATE   d'2'     ; 57600@Fosc=4MHz
#define SLEEP_          0xFF    ; Define SLEEP command
#define PECconst        0xF3    ; Define PEC constant
#define WDTCOUNT   d'22'   ; Approximately 1min time out

;SMBus control signals
#define _SCL    GPIO,1
#define _SDA    GPIO,0

;Flag register definitions
#define bit_out         flagreg0,0
#define bit_in          flagreg0,1

;
#define  RAM_Access   0x00 ; Define the MLX90614 command RAM_Accsess
#define  Ta                0x06 ; Define Ta  address in RAM
#define  To1               0x07 ; Define To1 address in RAM
#define  To2               0x08 ; Define To2 address in RAM
#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   TBUF
        CALL    delay           ;Wait a few microseconds

        RETLW   0               ; 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

        RETLW       0           ; End of  “STOP_bit”
```

```
;*******************************************************************************************************
;                                        Transmit data on SMBus
;*******************************************************************************************************
;Name:          TX_byte
;Function:      Send a byte on SMBus
;Input:         W (The sent byte must be load in W)
;Output:        W (Acknowledge bit is received in W)
;Comments:      If slave send acknowledge W=0, else W=1
;*******************************************************************************************************
TX_byte
        MOVWF       TX_buffer
        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
        RLF         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                             ; > Send clock pulse
        NOP                             ;|
```

```
NOP                             ;|
NOP                             ;|
NOP                             ;|
NOP                             ;|
_SCL_LOW                        ;|
NOP                             ;|
NOP                             ;|
RETLW       0
```

```
;*******************************************************************************************************
;                                        Receive data on  SMBus
;*******************************************************************************************************
;Name:          RX_byte
;Function:      Receive a byte on SMBus
;Input:         No
;Output:        RX_buffer(Received byte)
;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
        RLF         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
        RETLW 0                     ; End of "RX_byte"

Receive_bit
        BSF         bit_in          ; Set bit_in
        MOVLW       B'00001001'
        TRIS        6               ; _SDA(GP0)-input,GP1,GP2 -outputs
        _SCL_HIGH                   ;|
        NOP                         ;|
        NOP                         ;|
        NOP                         ;|
        NOP                         ;|
        NOP                         ;|
        NOP                         ;|
        NOP                         ;|
        NOP                         ; > Send 9th clock and check for acknowledge from Slave
        NOP                         ;|
        BTFSS       _SDA            ;|
        BCF         bit_in          ;|
        NOP                         ;|
        NOP                         ;|
        NOP                         ;|
        NOP                         ;|
        _SCL_LOW                    ;|
        NOP                         ;|
        NOP                         ;|
        RETLW 0                     ; Bit is received
```

```
;*******************************************************************************************************
;                                          Tunable delay
;*******************************************************************************************************
;Name:          delay
;Function:      Produces time delay depending on the value in counterL
;Input:         W
;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
        GOTO        $-1             ; Else decrement counterL again
        RETLW       0               ; End of “delay”

;*******************************************************************************************************
;                                   Fixed delay 30ms@Fosc=4MHz
;*******************************************************************************************************
;Name:          delay_30ms
;Function:      Produces time delay 30ms
;Input:         No
;Output:        No
;Comments:  Used in EXIT SLEEP MODE subroutine
;*******************************************************************************************************
delay_30ms

        MOVLW       d'2'            ;
        MOVWF       counterU        ; Load counterU

        MOVLW       d'20'           ;
        MOVWF       counterH        ; Load counterH < ----------
                                    ;                 |
        MOVLW       d'255'          ;                 |
        MOVWF       counterL        ; Load counterL < ---       |
                                    ;           |     |
        DECFSZ      counterL,F      ;           |     |
        GOTO        $-1             ;           |     |
        DECFSZ      counterH,F      ;           |     |
        GOTO        $-d'5'          ;-----------      |
        DECFSZ      counterU,F      ;                 |
        GOTO        $-d'9'          ;-------------------

        RETLW       0
```

```
;*******************************************************************************************************
;                                Fixed   delay 2ms@Fosc=4MHz
;*******************************************************************************************************
;
;Name:          delay_2ms
;Function:      Produces time delay 2ms
;Input:         No
;Output:        No
;Comments:
;*******************************************************************************************************
;
delay_2ms

        MOVLW       d'1'                ;
        MOVWF       counterU            ; Load counterU

        MOVLW       d'3'                ;
        MOVWF       counterH            ; Load counterH < ----------
                                        ;                           |
        MOVLW       d'220'              ;                           |
        MOVWF       counterL            ; Load counterL < ---       |
                                        ;                   |       |
        DECFSZ      counterL,F          ;                   |       |
        GOTO        $-1                 ;                   |       |
        DECFSZ      counterH,F          ;                   |       |
        GOTO        $-d'5'              ;-------------------        |
        DECFSZ      counterU,F          ;                           |
        GOTO        $-d'9'              ;---------------------------


        RETLW       0
```

```
;*******************************************************************************************************
;                                          Send Request
;*******************************************************************************************************
;Name:          SendRequest
;Function:      Switch module in SMBus mode
;Input:         No
;Output:        No
;Comments:  _SCL ___________<-----30ms---->___________
;                           \_____________/
;*******************************************************************************************************

SendRequest

        _SCL_LOW                ; Clear _SCL line
        CALL        delay_30ms  ; Wait 30ms
        _SCL_HIGH               ; Set _SCL line

        RETLW       0
```

```
;*******************************************************************************************************
;                                   Hexidecimal to ASCII conversion
;*******************************************************************************************************
Name:           hex2asc
;Function:      Convert a byte in ASCII code
;Input:         W
;Output:        digit1(high nibble),digit2(low nibble)
;Comments:
;*******************************************************************************************************
hex2asc
        MOVWF       tx_1                    ; W -> tx_1
        SWAPF       tx_1,W                  ; Swap tx_1 -> W
        ANDLW       0x0F                    ; Get low nibble
        MOVWF       tx_0                    ; W -> tx_0
        MOVLW       0x09                    ; 9 -> W
        SUBWF       tx_0,W                  ; tx_0 - 9 -> W
        BTFSC       STATUS,Z                ; nibble=9?
        GOTO        Equal_or_less_9_first   ; Yes, jump to Equal_or_less_9_first
        BTFSS       STATUS,C                ; nibble>9?
        GOTO        Equal_or_less_9_first   ; No, jump to to Equal_or_less_9_first
        MOVLW       0x37                    ;|
        ADDWF       tx_0,W                  ; > tx_0 + 0x37 -> digit1
        MOVWF       digit1                  ;|
        GOTO        second_nibble
Equal_or_less_9_first
        MOVLW       0x30                    ;|
        ADDWF       tx_0,W                  ; > tx_0 + 0x30 -> digit1
        MOVWF       digit1                  ;|
        GOTO        second_nibble

second_nibble
        MOVF        tx_1,W                  ; tx_1 -> W
        ANDLW       0x0F                    ; Get low nibble
        MOVWF       tx_0                    ; W -> tx_0
        MOVLW       0x09
        SUBWF       tx_0,W                  ; tx_0 - 9->W
        BTFSC       STATUS,Z                ; nibble=9?
        GOTO        Equal_or_less_9_second  ; Yes, jump to Equal_or_less_9_second
        BTFSS       STATUS,C                ; nibble>9?
        GOTO        Equal_or_less_9_second  ; No, jump to  Equal_or_less_9_second
        MOVLW       0x37                    ;|
        ADDWF       tx_0,W                  ; > tx_0 + 0x37 -> digit2
        MOVWF       digit2                  ;|
        RETLW       0

Equal_or_less_9_second
        MOVLW       0x30                    ;|
        ADDWF       tx_0,W                  ; > tx_0 + 0x37 -> digit2
        MOVWF       digit2                  ;|
        RETLW       0
```

```
;*******************************************************************************************************
;                                        Send byte by UART
;*******************************************************************************************************
;Name:          pic10_uart
;Function:      Send a byte by uart
;Input:         W
;Output:        No
;Comments:      Baudrate 57 600@Fosc=4MHz
;*******************************************************************************************************
#define TX_     GPIO,2

pic10_uart
        MOVWF   TX_buffer               ; Load the sent byte in a data register
        CALL    start_bit               ; Generate START bit
        CALL    send_byte               ; Send byte
        CALL    stop_bit                ; Generate STOP bit
        RETLW   0
;-------------------------------------------------------------------------------------------------------
stop_bit
        BCF     TX_                     ; Set TX_ line

        MOVLW   BAUDRATE                ;|
        MOVWF   counterL                ;|
        DECFSZ  counterL,f              ; > Define a single bit duration
        GOTO    $-1                     ;|
        NOP                             ;|

        RETLW   0
;-------------------------------------------------------------------------------------------------------
start_bit
        BSF     TX_                     ; Clear TX_ line

        MOVLW   BAUDRATE                ;|
        MOVWF   counterL                ;|
        DECFSZ  counterL,f              ; > Define a single bit duration
        GOTO    $-1                     ;|
        NOP                             ;|

         RETLW  0
;-------------------------------------------------------------------------------------------------------
send_byte
        MOVLW   D'8'                            ; 8 -> W
        MOVWF   Bit_counter                     ; W -> Bit_counter
send_next_bit
        RRF     TX_buffer,F                     ; Shift TX_buffer to right
        BTFSS   STATUS,C                        ; Check STATUS<C>
        GOTO    send_1                          ; If C=0 send 1 by the uart
        GOTO    send_0                          ; Else send 0 by the uart

send_1
        BSF     TX_                             ; Set TX_ line
```

```
        MOVLW   BAUDRATE                ;|
        MOVWF   counterL                ;|
        DECFSZ  counterL,f              ; > Define a single bit duration
        GOTO    $-1                     ;|
        NOP                             ;|

        GOTO    end_send_byte

send_0
        BCF     TX_                     ; Clear TX_ line

        MOVLW   BAUDRATE                ;|
        MOVWF   counterL                ;|
        DECFSZ  counterL,f              ; > Define a single bit duration
        GOTO    $-1                     ;|
        NOP                             ;|

end_send_byte
        DECFSZ  Bit_counter,F           ; All bits are sent?
        GOTO    send_next_bit           ; No,send next bit
        RETLW   0
```

```
;*******************************************************************************************************
;                                              MACROS
;*******************************************************************************************************
LoadNACKcounter  macro
                 MOVLW     D'0'
                 MOVWF     Nack_Counter
                 endm
;-------------------------------------------------------------------------------------------------------
_SDA_HIGH        macro
                 MOVLW     B'00001001'
                 TRIS      6              ; _SDA(GP0)-input,GP1,GP2 -outputs
                 endm
;-------------------------------------------------------------------------------------------------------
_SCL_HIGH        macro
                 BSF       _SCL           ; Set _SCL line
                 endm
;-------------------------------------------------------------------------------------------------------
_SDA_LOW         macro
                 BCF       _SDA           ; Clear _SDA bit
                 MOVLW     B'00001000'
                 TRIS      6              ; _SDA(GP0)-output,GP1,GP2 -outputs
                 endm
;-------------------------------------------------------------------------------------------------------
_SCL_LOW         macro
                 BCF       _SCL           ; Clear _SCL line
                 endm
;-------------------------------------------------------------------------------------------------------
EnterSleepMode    macro

                 local   restart
                 local   end_transmition
                 local   start

restart
                 DECFSZ    Nack_Counter,F ; If((Nack_Counter-1) == 0) stop
                                          ; transmition
                 GOTO      start          ; Else start transmition
                 GOTO      end_transmition
start
                 CALL      STOP_bit       ; Stop SMBus comunication
                 CALL      START_bit      ; Start SMBus comunication

                 MOVF      SlaveAddress,W ;
                 CALL      TX_byte        ; Send Slave address(Bit R/-W no
                                          ; meaning)

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

                 MOVLW     SLEEP_         ;
                 CALL      TX_byte        ; Send Command
```

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

                MOVLW       PECconst        ;
                CALL        TX_byte         ; Send PEC

                ANDLW       0x01            ; W & 0x01 -> W
                BTFSS       STATUS,Z        ; If Slave acknowledge,continue
                GOTO        restart         ; Else restart communication
                                            ; end_transmition
                CALL        STOP_bit        ; Stop SMBus comunication

                _SCL_LOW                    ; Clear _SCL line

                endm
;-------------------------------------------------------------------------------------------------------------------------
ExitSleepMode   macro

                _SCL_HIGH                   ; Set _SCL line
                NOP                         ;|
                NOP                         ; > Wait 3us
                NOP                         ;|
                _SDA_LOW                    ; Clear _SDA line
                CALL        delay_30ms      ; Wait 30ms
                CALL        delay_30ms      ; Wait 30ms
                CALL        delay_30ms      ; Wait 30ms
                _SDA_HIGH                   ; Set _SDA line

                endm
;-------------------------------------------------------------------------------------------------------------------------
DummyCommand    macro
                 CALL       START_bit       ; Start SMBus comunication
                 MOVF       SlaveAddress,W  ; SlaveAddress -> W
                 CALL       TX_byte         ; Send Slave address(Bit R/-W no
                                            ; meaning)
                 CALL       STOP_bit        ; Stop SMBus comunication
                 endm
;-------------------------------------------------------------------------------------------------------------------------
MemRead         macro
                local   restart
                local    end_transmition
                local   start
restart
                DECFSZ      Nack_Counter,F  ; If((Nack_Counter-1) == 0) stop
                                            ; transmition
                GOTO        start           ; Else start transmition
                GOTO        end_transmition ;
start
                CALL        STOP_bit        ; Stop SMBus comunication
                CALL        START_bit       ; Start SMBus comunication
```

```
MOVF    SlaveAddress,W   ; Send Slave address(Bit R/-
                         ; W no meaning)
CALL    TX_byte                  ;

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

MOVF    command,W        ; 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        ; Send Repeated START bit

MOVF    SlaveAddress,W   ; Send Slave address again(Bit R/-W
                         ; no meaning)
CALL    TX_byte          ;

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

BCF     bit_out          ; Master must send acknowledge
                         ; after this received byte
CALL    RX_byte          ; Receive low data byte
MOVF    RX_buffer,W      ;
MOVWF   DataL            ; Save it in DataL

BCF     bit_out          ; Master must send acknowledge
                         ; after this received byte
CALL    RX_byte          ; Receive high data byte
MOVF    RX_buffer,W      ;
MOVWF   DataH            ; Save it in DataH

BSF     bit_out          ; Master mustn't send acknowledge
                         ; after this received byte
CALL    RX_byte          ; Receive PEC byte
MOVF    RX_buffer,W      ;
MOVWF   PecReg           ; Save it in PecReg end_transmition

CALL    STOP_bit         ; Stop SMBus comunication

endm
```

Asembly of everything together

```
;*******************************************************************************************************
;                                        MCU initialization subroutine
;*******************************************************************************************************
MCUinit
;MCU port initialization
  MOVLW     B'00001000'
  TRIS      6               ;GP0, GP1 and GP2 -outputs
  MOVLW     b'11011111'     ;<7>-Disabled Wake-up on Pin Change bit(GP0, GP1, GP3)
  OPTION                    ;<6>-Disabled Weak Pull-ups bit(GP0, GP1, GP3)
                            ;<5>-Timer 0 transition on internal instruction cycle clock, FOSC/4
                            ;<3>-Prescaler assigned to the WDT
                            ;<2:0>-Prescaler Rate Select bits 1:128

ifndef   PIC10F202

  BCF       CMCON0,CMPON    ;Comparator is off (Only for PIC10F206)

endif

;SMBus initialization
  _SDA_HIGH                 ; Set
  _SCL_HIGH                 ; SMbus in idle mode

  MOVLW     WDTCOUNT        ;
  MOVWF     WDTcounter      ;Set WDT overflow time counter
  MOVLW     SA<<1           ; Slave address occupy MSb<7:1>
  MOVWF     SlaveAddress    ;Set SMBus address

  RETLW     0
```

```
;*******************************************************************************************************
;                                          MAIN PROGRAM
;*******************************************************************************************************
;Name:          main
;Function:      Read Ta ,To1 and To2 consequently by a MLX90614 module
;Input:
;Output:
;Comments:      Data is read during one minute approximately (TWDT * WDTCOUNT).
;               TWDT ≈ 2.3s - WDT time-out period (Prescaler 1:128 is used)
;               WDTCOUNT – adjust data read interval
;*******************************************************************************************************
main
        ifdef   SLEEPON

        MOVF            STATUS,W        ; Read STATUS(-TO and -PD flags) in W
        ANDLW           b'00011000'     ; Mask 3th(-TO) and 4th(-PD) bits
        CLRWDT                          ; Set -TO and -PD in STATUS again
        BTFSS           STATUS,Z        ; -TO=0 and -PD=0?
        GOTO            start           ; No,waking up is from POR,start reading-----
        DECFSZ          WDTcounter,F    ; If WDTcounter=0 wake up the module       |
        SLEEP                           ; Else put MCU in sleep mode again         |
        ExitSleepMode                   ; Take out the module from sleep mode      |
        CALL            delay_2ms       ; Wait 2ms                                 |
                                        ;                                          |
        endif                           ;                                          |
                                        ;                                          |
start                                   ;<-----------------------------------------
        CALL            MCUinit         ; MCU initialization

        CALL            delay_30ms      ;|
        CALL            delay_30ms      ;|
        CALL            delay_30ms      ; > Output valid after POR (see MLX90614
        CALL            delay_30ms      ; > datasheet electrical specification-Tvalid)
        CALL            delay_30ms      ;|
        CALL            delay_30ms      ;|

        CALL            SendRequest     ; Switch module in SMBus mode
        CALL            delay_30ms      ;
        CALL            delay_30ms      ; Wait 40ms minimum before reading data
        DummyCommand                    ; This command has meaning if a module is in
                                        ; SMBus mode and the module is switched again in
                                        ; SMBus mod through SMBus request(see
                                        ; SendRequest.asm)
ReadLoop
;-------------------------------------------------------------------------------------------------------

;Read Ta
        CLRWDT                          ; Clear WDT
        MOVLW           Ta|RAM_Access   ; Form RAM access command + RAM address
        MOVWF           command         ; Load RAM address 0x06(Ta)

        LoadNACKcounter                 ; Set Nack_counter
```

```
        MemRead                         ; Read Ta

        MOVF    Nack_Counter,W          ; Store Nack_Counter in W
        ANDLW   0xFF                    ; W & 0xFF -> W
        BTFSS   STATUS,Z                ; If Nack_Counter=0 send '-'
        GOTO    send_Ta                 ; Else send data
        MOVLW   '-'                     ;|
        CALL    pic10_uart              ; > Send '-' and ','
        MOVLW   ','                     ;|
        CALL    pic10_uart              ;|
        GOTO    Read_To1                ; Go to read To1

send_Ta
        MOVF    DataH,W                 ; Store DataH in W
        CALL    hex2asc                 ; Convert result to ASCII

        MOVF    digit1,W                ;|
        CALL    pic10_uart              ;|
        MOVF    digit2,W                ; > Send DataH the software uart
        CALL    pic10_uart              ;|

        MOVF    DataL,W                 ; Store DataL in W
        CALL    hex2asc                 ; Convert result to ASCII

        MOVF    digit1,W                ;|
        CALL    pic10_uart              ;|
        MOVF    digit2,W                ; > Send DataL by the software uart
        CALL    pic10_uart              ;|

        MOVLW   'h'
        CALL    pic10_uart              ; Send 'h'
        MOVLW   ','
        CALL    pic10_uart              ; Send comma
;-------------------------------------------------------------------------------------------------------------------
Read_To1
        CLRWDT                          ; Clear WDT
        MOVLW   To1|RAM_Access          ; Form RAM access command + RAM address
        MOVWF   command                 ; Load RAM address 0x07(To1)

        LoadNACKcounter                 ; Set Nack_counter
        MemRead                         ; Read To1

        MOVF    Nack_Counter,W          ; Store Nack_Counter in W
        ANDLW   0xFF                    ; W & 0xFF -> W
        BTFSS   STATUS,Z                ; If Nack_Counter=0 send '-'
        GOTO    send_To1                ; Else send data
        MOVLW   '-'                     ;|
        CALL    pic10_uart              ; > Send '-' and ','
        MOVLW   ','                     ;|
        CALL    pic10_uart              ;|
        GOTO    Read_To2                ; Go to read To2
```

```
send_To1
        MOVF    DataH,W         ; Store DataH in W
        CALL    hex2asc         ; Convert result to ASCII

        MOVF    digit1,W        ;|
        CALL    pic10_uart      ;|
        MOVF    digit2,W        ; > Send DataH by the software uart
        CALL    pic10_uart      ;|

        MOVF    DataL,W         ; Store DataL in W
        CALL    hex2asc         ; Convert result to ASCII

        MOVF    digit1,W        ;|
        CALL    pic10_uart      ;|
        MOVF    digit2,W        ; > Send DataL by the software uart
        CALL    pic10_uart      ;|

        MOVLW   'h'
        CALL    pic10_uart      ; Send 'h'
        MOVLW   ','
        CALL    pic10_uart      ; Send comma
;-------------------------------------------------------------------------------------------------------------------
Read_To2
        CLRWDT                  ; Clear WDT
        MOVLW   To2|RAM_Access  ; Form RAM access command + RAM address
        MOVWF   command         ; Load RAM address 0x08(To2)

        LoadNACKcounter         ; Set Nack_Counter
        MemRead                 ; Read To2

        MOVF    Nack_Counter,W  ; Store Nack_Counter in W
        ANDLW   0xFF            ; W & 0xFF -> W
        BTFSS   STATUS,Z        ; If Nack_Counter=0 send '-'
        GOTO    send_To2        ; Else send data
        MOVLW   '-'             ;|
        CALL    pic10_uart      ; > Send '-' and ','
        MOVLW   '\r'            ;|
        CALL    pic10_uart      ;|
        GOTO    FrameEnd        ; Go to read loop

send_To2
        MOVF    DataH,W         ; Store DataH in W
        CALL    hex2asc         ; Convert result to ASCII

        MOVF    digit1,W        ;|
        CALL    pic10_uart      ;|
        MOVF    digit2,W        ; > Send DataH by the software uart
        CALL    pic10_uart      ;|

        MOVF    DataL,W         ; Store DataL in W
        CALL    hex2asc         ; Convert result to ASCII
```

```
        MOVF    digit1,W        ;|
        CALL    pic10_uart      ;|
        MOVF    digit2,W        ; > Send DataL by the software uart
        CALL    pic10_uart      ;|

        MOVLW   'h'
        CALL    pic10_uart      ; Send 'h'
        MOVLW   '\r'
        CALL    pic10_uart      ; Send Carriage return
;-------------------------------------------------------------------------------------------------------------

FrameEnd
        ifdef SLEEPON

        LoadNACKcounter         ; Set Nack_Counter
        EnterSleepMode          ; Put module in sleep mode
        SLEEP                   ; Put PIC10 in sleep mode,…waking up from WDT
                                ; overflow
        else

        CALL    delay_30ms      ;|
        CALL    delay_30ms      ;|
        CALL    delay_30ms      ; > Wait before begin next frame
        CALL    delay_30ms      ;|
        CALL    delay_30ms      ;|
        GOTO    ReadLoop        ; Start next frame

        endif


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

Conclusion

MLX90614 can be read via SMBus even with the smallest MCUs, like the PIC10F202. This Application Note describes how to read and transmit the temepratures via UART (PC COM port). The same SMBus routines can be used to read the data and process it in a different way, too.

♦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
…	…
Melexis reserved	0x1Ch