Application Note

Read PWM from MLX90614 and calculate temperature with PIC18 MCU

Scope

This application note describes how to read the single and extended PWM output from the MLX90614 Infra-Red thermometer and calculate the temperature with a PIC MCU. The calculated Duty Cycle is sent by UART in ASCII format. A software implementation is used to read the PWM signal. The code is in assembly language for Microchip's PIC®18. 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;
- ■ Body temperature measurement

Related Melexis Products

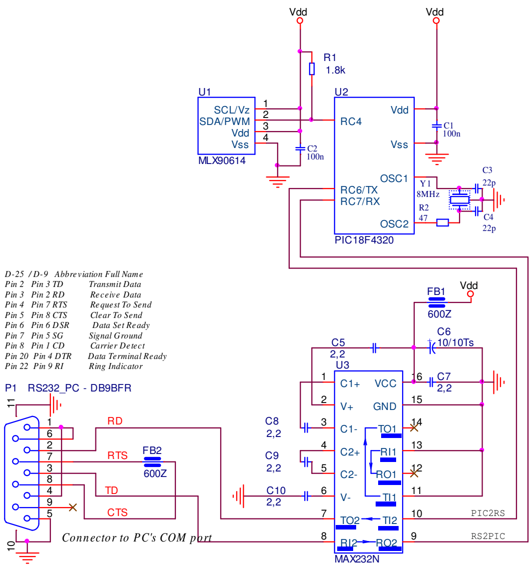
EVB90614 is the evaluation board which supports the MLX90614 devices.

Other Components Needed

Elements used in the schematics hereafter are:

- SMD ceramic capacitors C1 and C2 100nF 16V or higher.
- SMD ceramic capacitors C3 and C4 22pF 16V or higher.
- Resistors R1 1.8kOhm 5% and R2 47 Ohm 5%.
- Ceramic capacitors C5, C7, C8, C9, C10 2.2µF 16V or higher
- Electrolytic/Tantalum capacitor C6 10µF 16V or higher
- Ferrite Bead FB1, FB2 600Z
- Quarz resonator Y1 8.00MHz
- PIC18F4320 microcontroller.
- Infra Red Thermometer MLX90614Axx in TO-39, configured in EEPROM for PWM output POR default
- Connector to PC's COM port DB9 female
- RS232 driver/receiver MAX232
- Regulated 5V power source.

Typical Circuit



Explanation

The connection of a MLX90614 to a MCU is very simple. One general purpose pin RC4 of the PIC18F4320 is used. A pull-up resistor R1 is connected between the Vdd and the SDA/PWM line, SCL line. This pull-up allows the Open Drain output of the MLX90614 to be used. In case the push-pull output configuration is programmed in the MLX90614 this resistor is not needed. C1 is the local power supply bypass decoupling capacitor. The MLX90614 needs that to bypass the on-chip digital circuitry switching noise. C2 has the same function for the microcontroller. The common value of 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

MLX90614 and PWM mode

The MLX90614 can be read via PWM or SMBus compatible interface. Selection of the PWM output and settings are done in PWMCTRL and CongifRegister1 in EEPROM. The PWM output has two programmable formats: single or dual data transmission, the latter providing single wire reading of two temperatures. Fig.1 shows single and extended PWM mode timings.

PWMCTRL

Bit 0	Select the type of PWM mode:	1 - Single PWM, factory default for MLX90614xAx	0 – Extended PWM, factory default for MLX90614xBx
Bit 1	Enable/disable the PWM:	1 - Enable PWM, disable SMBus	0 - Disable PWM (Enable SMBus), Factory default
Bit 2	Configuration of the pin PWM:	1 - Push-Pull,	0 – OpenDrain NMOS, factory default
Bit 3	Mode selection	1 - ThermoRelay,	0 - PWM, Factory default
Bits[8:4]	Extemded PWM definition	Number of repetitions divided by 2 of sensor 1 and 2 in Extended PWM mode. The number of repetitions can vary from 0 to 64 times.	
Bits[15:9]	PWM clock configuration	2MHz divided by the number written in this place. (128 in case the number is 0.) A single PWM period consists of 2048 clocks and extended PWM of 4096 clocks for each period (2T in figure 6). The 2 MHz clock is valid for the nominal HFO frequency.	

ConfigRegister1

		Bit 5	Bit 4	Data 1	Data 2
Bits[5:4]	– Configure the type of data transmitted through PWM:	**0**	**0**	**Ta**	**IR 1**
		0	**1**	**Ta**	**IR 2**
		1	**1**	**IR 1**	**IR 2**
		1	**0**	**IR 2**	**Undefined***

Note: In Single PWM mode the settings for Data1 only are used

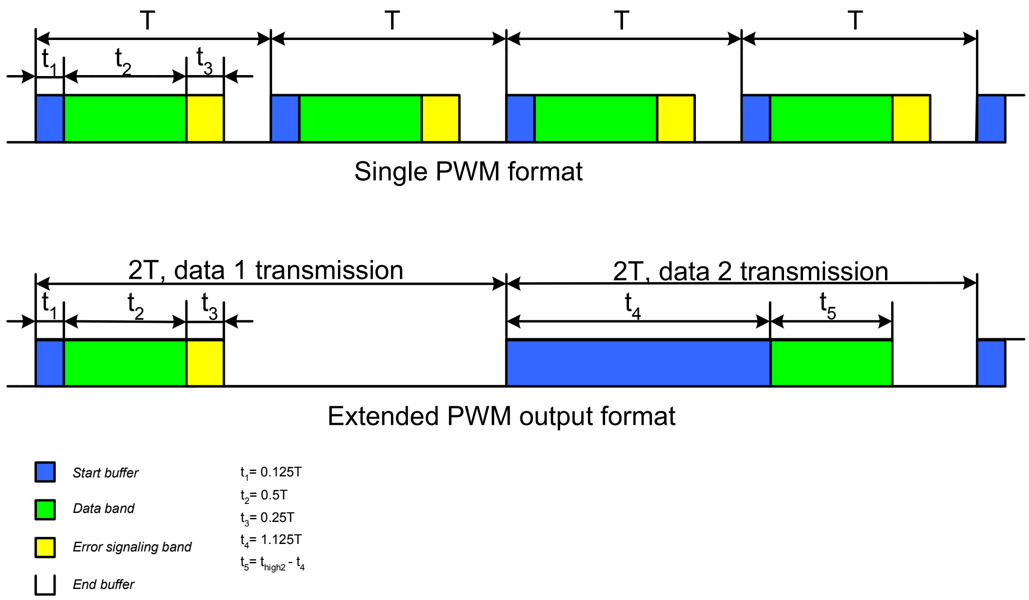


Fig.1 PWM timing

The temperature can be calculated from the signal timing as:

Single PWM:

$$Tout = \left[\frac{2t_2}{T} * (Tmax - Tmin)\right] + Tmin$$

Extended PWM:

$$Tout1 = \left[\frac{2t_2}{T} * (T1max - T1min)\right] + T1min$$

$$Tout2 = \left[\frac{2t_5}{T} * (T2max - T2min)\right] + T2min$$

Tmin and Tmax are the EEPROM PWM temperature ranges (refer to the MLX90614 datasheet for details) and T is the PWM period. There is one range for both object temperatures (To), and another one for ambient temperature (Ta).
Factory defaults are -20…+120℃ for To and -20.28…+119.88℃ for Ta.

Theory of the PWM measurement

Fig.2 represents the concept of the single PWM measurement. When the rising edge of the PWM pulse is detected a TIMER is started. When the falling edge is detected the value of the TIMER is stored in variable t_{pwm}. On the next rising edge the value of the TIMER is stored in variable T_{pwm}, and the TIMER is cleared. When extended mode is measured two contiguous periods must be captured and data should be saved in t1pwm, T1pwm and t2pwm, T2pwm variables (Fig.3). T1pwm is typically equal to T2pwm.

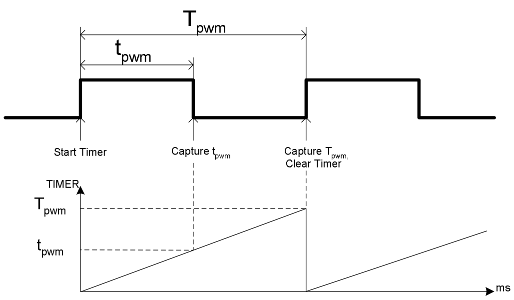


Fig.2 Single PWM measurement

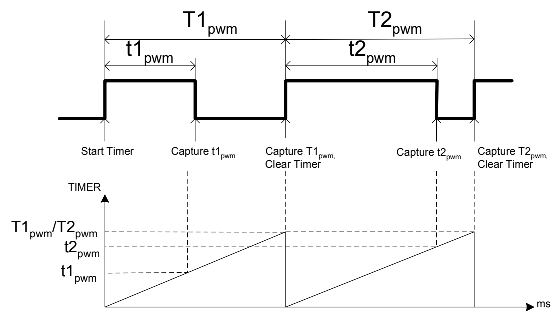


Fig. 3 Extended PWM measurement

The formulas for the temperatures can be rewritten as:

Single mode:

$$Tout = \left[\frac{2t_2}{T} * (Tmax - Tmin)\right] + Tmin = \left[\frac{2(tpwm - t1)}{Tpwm} * (Tmax - Tmin)\right] + Tmin =$$

$$= \left[2\left(\frac{tpwm}{Tpwm} - 0.125\right) * (Tmax - Tmin)\right] + Tmin = [2(DC - 0.125) * (Tmax - Tmin)] + Tmin \text{, where}$$

$$DC = \frac{tpwm}{Tpwm} \text{, Duty Cycle}$$

Extended Mode:

$$Tout1 = [2(DC1 - 0.125) * (T1max - T1min)] + T1min \text{, where}$$

$$DC1 = \frac{t1pwm}{\frac{T1pwm}{2}} = 2\frac{t1pwm}{T1pwm}$$

$$Tout2 = [2(DC2 - 1.125) * (T2max - T2min)] + T2min \text{, where}$$

$$DC2 = \frac{t2pwm}{\frac{T2pwm}{2}} = 2\frac{t2pwm}{T2pwm}$$

Note: In extended PWM mode the period is twice the period of the single mode so in the formulas for temperature calculation only the half of the period is used (see Fig.1).
Note: The MLX90614 supports repetition of PWM periods as setting in EEPROM. Current Application Note always assumes no repetitions are enabled.

To calculate the temperature measured from PWM we need to find Duty Cycle (DC, high time to period ratio). Temperature ranges (as set in MLX90614 EEPROM) must also be known.

Firmware Description

The minimum period which will be measured is 1ms (in single mode) and the maximum period will be 262ms (in extended mode). TIMER used has to be able to cover the maximum range and to maintain good resolution for the minimum period.
This example uses Timer1 of a PIC18.This is a 16bit timer. It is extended to 24 bit in software. When Timer1 registers TMR1H:TMR1L overflow an interrupt is generated. In the interrupt service routine a register called TMR1U is incremented. TMR1U is actually the upper byte of Timer1.
Timer1 also has a prescaler which is not used here. Timer1 increments on every instruction cycle Tcy. The data field resolution is 10 bit and the data band is 0.5T. Thus the capture resolution needs to be better than T/2048 in order not to degrade the resolution of the MLX90614 PWM data. In the accompanying project PIC MCU uses 8 MHz quartz resonator with PLL enabled. This way the capture resolution is Tcy=125 ns.
The firmware is written as absolute code. The main source file is called main.asm and uses the following files:

```
#include    "config.inc"        ; Contans description of configuration bits of PIC18
#include    "GPRs.inc"          ; Contains RAM definitions
#include    "PwmSettings.inc"   ; Contains the PWM settings
#include    "macros.inc"        ; Contains general purpose macros used in the firmware
#include    "math.inc"          ; Contains mathematical macros

#include    "BLOAD_PM.asm"      ; Contains bootloader firmware
#include    "MCUinit.asm"       ; Contains MCU initialization firmware
#include    "pwm.asm"           ; Contains PWM reading firmware
#include    "Uart.asm"          ; Contains UART subroutines
#include    "ArrangeData.asm"   ; Arranges the measured data in the order t1,T1,t2,T2
#include    "Calculation.asm"   ; Calculates the temperatures
#include    "SendDC.asm"        ; Sends Duty Cycle by UART in ASCII view
#include    "delay.asm"         ; Contains subroutines for delays
#include    "Convert.asm"       ; Converts the 16-bits sign number in BCD code
#include    "hex2asc.asm"       ; Converts a hexadecimal digit in its ASCII representation
#include    "header.asm"        ; Sends a header by UART for log file utilization
```

The firmware uses basic settings, such as temperature ranges. They are grouped in the file “PwmSettings.inc”. The content of this file is shown bellow.

```
;1. Uncomment this if single PWM output is measured
#define  SINGLE
;------------------------------------------------------------------------------------------
;2. User can change PWM input here
#define  PWMIO      TRISC,4
#define  PWMinput   PORTC,4
;------------------------------------------------------------------------------------------
;3. User can change PWM temperature range for T1 here
        radix dec
T1max equ 120
T1min  equ -20
K1      equ 2*(T1max-T1min)
        radix hex
```

```
;--------------------------------------------------------------------------------------------
;4.User can change PWM temperature range for T2 here
        radix dec
T2max equ 120
T2min  equ -20
K2       equ 2*(T2max-T2min)
        radix hex
;--------------------------------------------------------------------------------------------
```

From this file a user can tune the firmware depending on whether single or extended PWM output will be measured (paragraph 1), which pin of MCU will be used as PWM input (paragraph 2) and what the PWM temperature ranges are (paragraph 3 and paragraph 4). In single PWM mode paragraph 4 is “don’t care”.
In the firmware the formulas for temperature calculations are simplified in the following manner:

Single PWM Mode: $T=(DutyCycle - 0.125)*K + Tmin$
$DutyCycle=t1_{pwm}/T1_{pwm}$
$K=K1$

Extended PWM Mode: $T1=(2*DutyCycle1 - 0.125)*K1 + T1min$
$DutyCycle1=t1_{pwm}/T1_{pwm}$
$K1=2*(T1max - T1min)$

$T2=(2*DutyCycle2 - 1.125)*K2 + T2min$
$DutyCycle2=t2_{pwm}/T_{2pwm}$
$K2=2*(T2max - T2min)$

When values for T1min and T1max (in paragraph 3) and/or T2min and T2max (in paragraph 4) are given, the constants K1 and K2 are automatically calculated upon project compilation (build) so no other user intervention is needed.

A fragment of main.asm file is given below. It contains all basic steps for PWM measurements and calculations.

```
;*******************************************************************************************************
;                                              MAIN
;*******************************************************************************************************
;Name:          MAIN
;Function:      Read PWM output of a MLX90614 calculate temperatures and sends Duty Cycle
;               by UART in ASCII view
;Input:
;Output:
;Comments:  If Single PWM output is measured uncomment paragraph 1 in PwmSettings.inc file
;
;*******************************************************************************************************
MAIN

        CALL   MCUinit                  ; MCU initialization
        CALL   delay_100ms              ;\  MLX90614 output is valid 0.15s after POR
        CALL   delay_100ms              ;/
        CALL   header                   ; Send a header by UART

loop
```

```
;----------------------------------------------------------- Read T1-----------------------------------------------------
        CALL   CapturePWM          ; Read PWM output of a MLX90614

#ifndef SINGLE
        CALL   Arrange             ; Arrange data if extended PWM mode is measured
#endif
        CALL   Calculate_T1        ; Calculate temperature, the result is in T1H:T1L


;------------------------------------------------------- Read T2 --------------------------------------------------------
#ifndef SINGLE
        CALL   Calculate_T2        ; Calculate temperature, result is in T2H:T2L

#endif

        CALL   Send_DC             ; Send Duty Cycle by UART
        CALL   delay_100ms         ; Wait 100ms
        CALL   delay_100ms         ; Wait 100ms
        CALL   delay_100ms         ; Wait 100ms
        CALL   delay_100ms         ; Wait 100ms
        CALL   delay_100ms         ; Wait 100ms

        GOTO loop                  ; Read PWM output again


        END

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

Explanation of this code:
First step is the initialization of the MCU in MCUinit subroutine (IO initialization, UART, timers, interrupts and so on). After the initialization 200ms delay is added which allows MLX90614 output to be configured and operating after POR. A header file is send by UART for log file utilization.
Next, a subroutine CapturePWM which measures the PWM output of a MLX90614 is called. For single PWM output the high duration of the pulse is saved in the variable t1pwm which is 24bit wide (t1pwmU:t1pwmH:tpwmL) and the period in the variable T1pwm also 24bit wide (T1pwmU:T1pwmH:T1pwmL). For extended PWM mode variables are t1pwm(t1pwmU:t1pwmH:t1pwmL) and T1pwm (T1pwmU:T1pwmH:TpwmL) for one temperature channel and t2pwm(t2pwmU:t2pwmH:t2pwmL) and T2pwm (T2pwmU:T2pwmH:T2pwmL) for the another.
Measurement of extended PWM mode is asynchronous. Data1 and Data2 fields can be captured in any sequence (Data1, Data2 or Data2, Data1). Data captured first is stored in t1pwm and second one – in t2pwm. The Arrange subroutine is called afterwards to arrange the two captured data fields so that Data1 is in t1pwm and T1pwm and Data2 in t2pwm and T2pwm. The maximum duration of the high level of the first temperature channel is $t1_{highmax} = t_1+t_2+t_3 = 0.125T+0.5T+0.25T = 0.875T$. The minimum duration of high level of the second temperature channel is $t2_{highmin} = t4 = 1.125T$. From this equations follow that $t2_{high}$ is always bigger than $t1_{high}$. So if we compare the durations of the high level of the two temperature channels we can determine which temperature channel we have captured first and if is necessary to swap t1pwm and t2pwm and T1pwm and T2pwm. This is done in Arrange subroutine compares variables t1pwm(the duration of the first captured channel) and t2pwm(the duration of the second

captured channel). As T1pwm and T2pwm are virtually equal, the higher value of the high time captured defines the Data2 field.
The next step is the calculation of the temperatures from the measured values of one or two temperature channels. This is done in Caclulate_T1 and Calculate_T2 subroutines. To avoid floating point arithmetic the next arithmetic transformation is used to reduce all calculations with integer values.

Single PWM mode:

$$Tout = \left[\frac{(\ t1pwm * 100000\ )}{T1pwm} - 0.125 * 100000 \right] * K1 + T1\min * 100000$$

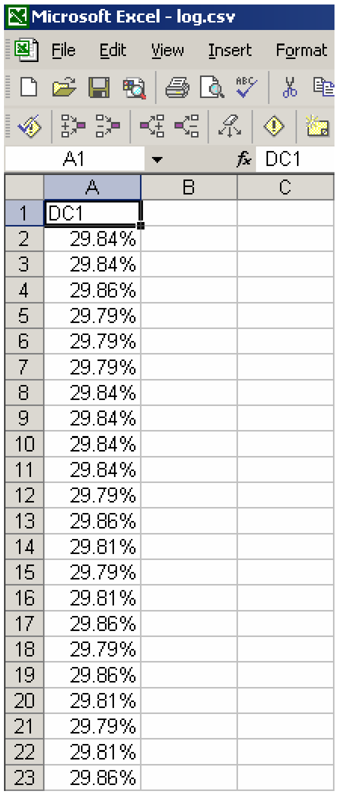
Extended PWM Mode:

$$Tout1 = \left[2\frac{(\ t1pwm * 100000\ )}{T1pwm} - 0.125 * 100000 \right] * K1 + T1\min * 100000$$

$$Tout2 = \left[2\frac{(\ t2pwm * 100000\ )}{T2pwm} - 1.125 * 100000 \right] * K2 + T2\min * 100000$$

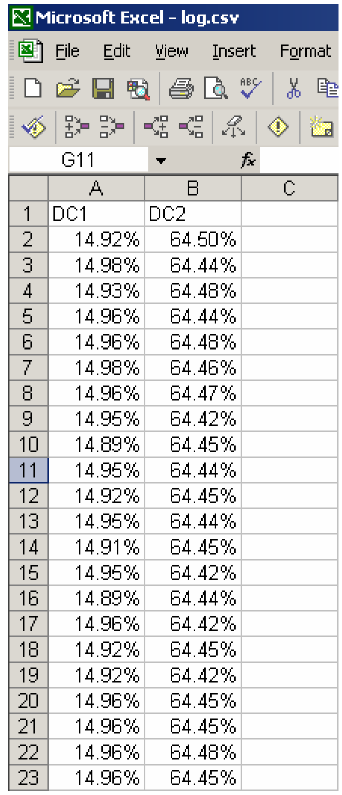
The maximum values calculations can return are 38200000 (382.00000) for the object temperature and 12500000 (125.00000) for the ambient temperature. (These are the maximum MLX90614 values for upper limits of the ranges for PWM and are in degrees Celsius.)
Multiplication by 100 000 shifts the fixed point 5 positions to the right. To truncate the resolution to 0.01℃ a division by 1,000 is done on the results. This results in 38200 (382.00) for To and 12500 (125.00) for Ta, where “38200” is the maximum number (in 0.01℃ LSBs). The final results are stored in the T1H:T1L registers for the first temperature channel and in T2H:T2L registers for the second one (in single PWM mode the result is in T1H:T1L).
Afterwards these values can be used for example to display the temperatures on a LCD.
The subroutines Calculate_T1 and Calculate_T2 also store the Duty Cycles of both temperature channels in DCU1:DCH1:DCL1 (Duty Cycle 1) registers and in DCU2:DCH2:DCL2 (Duty Cycle 2) registers. Later these values are sent by UART in ASCII code. This is done in the subroutine called Send_DC. Fig.4 and Fig.5 give fragments of log files with extension “.csv” which show how a Duty Cycle is presented in a log file.

Program loops in 500 ms cycles.



	A	B	C
1	DC1		
2	29.84%		
3	29.84%		
4	29.86%		
5	29.79%		
6	29.79%		
7	29.79%		
8	29.84%		
9	29.84%		
10	29.84%		
11	29.84%		
12	29.79%		
13	29.86%		
14	29.81%		
15	29.79%		
16	29.81%		
17	29.86%		
18	29.79%		
19	29.86%		
20	29.81%		
21	29.79%		
22	29.81%		
23	29.86%		

Fig.4 Single PWM mode



	A	B	C
1	DC1	DC2	
2	14.92%	64.50%	
3	14.98%	64.44%	
4	14.93%	64.48%	
5	14.96%	64.44%	
6	14.96%	64.48%	
7	14.98%	64.46%	
8	14.96%	64.47%	
9	14.95%	64.42%	
10	14.89%	64.45%	
11	14.95%	64.44%	
12	14.92%	64.45%	
13	14.95%	64.44%	
14	14.91%	64.45%	
15	14.95%	64.42%	
16	14.89%	64.44%	
17	14.96%	64.42%	
18	14.92%	64.45%	
19	14.92%	64.42%	
20	14.96%	64.45%	
21	14.96%	64.45%	
22	14.96%	64.48%	
23	14.96%	64.45%	

Fig.5 Extended PWM mode

Conclusion

The advantage of the PWM mode compared to SMBus mode is that only one wire is needed to read one or two temperatures. This is cost effective approach in many systems. Also , the PWM mode is preferred for long wires than. The PWM line is less susceptible to noise and thus also preferred in a hostile EMI environment.

Disclaimer

IATF 16949 and ISO 14001 Certified