Contents

1. Legal notice

The method described in document does not fall under the Apache2 license covering the rest of the Melexis MLX90640/1 library.

Melexis has applied for a patent on this specific method of calculating the temperature by use of these lookup tables (patent application number: EP19177500.6). This method may only be used to calculate the temperature results from the raw data of a Melexis infrared sensor. With the purchase of a Melexis MLX90640/1 infrared sensor comes the right to use this software to calculate temperatures based on the output of that sensor.

2. Introduction

In some applications, the calculations needed to obtain the object temperature from the raw data generated by the MLX90640/1 sensor are too compute heavy for the available microcontroller. That can be the case if the microcontroller is a low performance version, has limited calculation power and/or If the required refresh rate of the frames is too fast. In this case, using look-up tables can be especially beneficial to handle the calculations. It does have the disadvantage that more memory will be needed and that in the concept phase one will have to be more careful in defining the temperature operating ranges.

In order to use the MLX90640 look-up table functions there are 2 files and the MLX90640 driver that should be included in the C project:

- *MLX90640 driver* – the MLX90640 driver that ensures proper communication with the device and pre-processing of the data
    - *MLX90640_I2C_Driver.h* – header file containing the definitions of the I2C related functions
    - *MLX90640_I2C_Driver.cpp or MLX90640_SWI2C_Driver.cpp* – file containing the I2C related functions
    - *MLX90640_API.h* – header file containing the definitions of the MLX90640 specific functions
    - *MLX90640_API.h* – file containing the MLX90640 specific functions
- *MLX90640_LUT.h* – header file containing the definitions of the MLX90640 look-up table functions
- *MLX90640_LUT.cpp* – file containing the MLX90640 look-up table functions

3. MLX90640 driver

This is the driver for the MLX90640 device. The user should make sure that the I2C communication is working properly and the pre-processing functions of the driver are being used accordingly. For more detail, please refer to the MLX90640 driver documentation.

4. MLX90640 look-up table functions

These are the functions that implement a proper look-up table approach to acquire the temperatures measured by MLX90640 devices. The user should not change these functions.

4.1. MLX90640 API functions

*4.1.1. int MLX90640_GenerateLUTs(float tMin, float tMax, float tStep, paramsMLX90640 *params, float *lut1, float *lut2)*

This function generates the two look-up tables required for the calculation of all the pixels object temperatures when using the look-up table approach. Using the look-up tables speeds-up the calculation of the object temperatures and allows using less powerful MCUs to handle the MLX90640. The function is flexible using as input parameters the minimum and the maximum required object temperature as well as the temperature step for the look-up tables. Having a narrower range (tMax-tMin) would require less

memory to be allocated for the look-up table. Having bigger temperature step results in less memory to be allocated, but the accuracy of the calculated temperature would be decreased. On the other hand, a smaller temperature step would require more memory to be reserved for the look-up tables, but the accuracy is also increased. The function returns the number of the generated look-up table lines. If the generated look-up tables lines is less than 1, the most probable cause is that the *float tMin, float tMax, float tStep* parameters are wrong.

Note: If the MLX90640_CalculateTo function is used to calculate the object temperatures, the MLX90640_GenerateLUTs function is not required

- *float tMin* – The minimum temperature to generate the look-up table for
- *float tMax* – The maximum temperature to generate the look-up table for
- *float tStep* – The temperature step in °C to generate the look-up table for
- *float *lut1* – pointer to the MCU memory location where the user wants to generate look-up table 1
- *float *lut2* – pointer to the MCU memory location where the user wants to generate look-up table 2

Example:

1. *Generate the look-up tables for a range of [-40*°C*:200*°C] with a 2°C step

```
#define   LUT_MIN_TEMPERATURE     -40     //°C
#define   LUT_STEP                2       //°C
#define   LUT_MAX_TEMPERATURE     200     //°C
#define   LUT_LINES_NUM           121     //Should be calculated as
                                          //ceil((LUT_MAX_TEMPERATURE-LUT_MIN_TEMPERATURE)/LUT_STEP) + 1;
int lutLines = LUT_LINES_NUM;
static float lut1[LUT_LINES_NUM];
static float lut2[LUT_LINES_NUM];


float Ta;
unsigned char slaveAddress;
static int eeMLX90640[832];
static int mlx90640Frame[834];
paramsMLX90640 mlx90640;
int status;
```

status = MLX90640_DumpEE (slaveAddress, eeMLX90640);

status = MLX90640_ExtractParameters(eeMLX90640, &mlx90640);

lutLines = MLX90640_GenerateLUTs(LUT_MIN_TEMPERATURE, LUT_MAX_TEMPERATURE, LUT_STEP, pParam, lut1, lut2);

4.1.2. void MLX90640_LookUpTo(uint16_t *frameData, const paramsMLX90640 *params, float emissivity, float tr, float tMin, float tStep, uint16_t lutLines, float *lut1, float *lut2, float *result)

This function uses a look-up table to calculate the object temperatures for all 768 pixel in the frame all based on the frame data read from a MLX90640 device, the extracted parameters for that particular device and the emissivity defined by the user. The allocated memory should be at least 768 words for proper operation.

- *uint16_t *frameData* – pointer to the MCU memory location where the user wants the frame data to be stored
- *paramsMLX90640 *params* – pointer to the MCU memory location where the already extracted parameters for the MLX90640 device are stored
- *float emissivity* – emissivity defined by the user. The emissivity is a property of the measured object
- *float tr* – reflected temperature defined by the user. If the object emissivity is less than 1, there might be some temperature reflected from the object. In order for this to be compensated the user should input this reflected temperature. The sensor ambient temperature could be used, but some shift depending on the enclosure might be needed. For a MLX90640 in the open air the shift is -8°C.
- *float tMin* – The minimum temperature used to generate the look-up table
- *float tStep* – The temperature step in °C used to generate the look-up table
- *uint16_t lutLines* – the number of lines in the generated look-up table
- *float *lut1* – pointer to the MCU memory location where the user has generated look-up table 1
- *float *lut2* – pointer to the MCU memory location where the user has generated look-up table 2
- *float *result* – pointer to the MCU memory location where the user wants the object temperatures data to be stored

Example:

1. *Look-up the object temperatures for all the pixels in a frame, object emissivity is 0.95 and the reflected temperature is the sensor ambient temperature:*

```
#define   LUT_MIN_TEMPERATURE     -40     //°C
#define   LUT_STEP                2       //°C
#define   LUT_MAX_TEMPERATURE     200     //°C
#define   LUT_LINES_NUM           121     //Should be calculated as
```

```
                                                    //ceil((LUT_MAX_TEMPERATURE-LUT_MIN_TEMPERATURE)/LUT_STEP)+1;
#define TA_SHIFT 8                                  //the default shift for a MLX90640 device in open air

int lutLines = LUT_LINES_NUM;

static float lut1[LUT_LINES_NUM];

static float lut2[LUT_LINES_NUM];


float Ta;

unsigned char slaveAddress;

static int eeMLX90640[832];

static int mlx90640Frame[834];

paramsMLX90640 mlx90640;

int status;

status = MLX90640_DumpEE (slaveAddress, eeMLX90640);

status = MLX90640_ExtractParameters(eeMLX90640, &mlx90640);

lutLines = MLX90640_GenerateLUTs(LUT_MIN_TEMPERATURE, LUT_MAX_TEMPERATURE, LUT_STEP, pParam, lut1, lut2);

status = MLX90640_GetFrameData (0x33, mlx90640Frame);

tr = MLX90640_GetTa(mlx90640Frame, &mlx90640) – TA_SHIFT;

MLX90640_LookUpTo(mlx90640Frame, &mlx90640, emissivity, tr, LUT_MIN_TEMPERATURE, LUT_STEP, lutLines, lut1, lut2, &mlx90640To[i][0]);

//The object temperatures for all 768 pixels in a frame are stored in the mlx90640To array
```

5. Disclaimer

The information furnished by Melexis herein ("Information") is believed to be correct and accurate. Melexis disclaims (i) any and all liability in connection with or arising out of the furnishing, performance or use of the technical data or use of the product(s) as described herein ("Product") (ii) any and all liability, including without limitation, special, consequential or incidental damages, and (iii) any and all warranties, express, statutory, implied, or by description, including warranties of fitness for particular purpose, non-infringement and merchantability. No obligation or liability shall arise or flow out of Melexis' rendering of technical or other services.
The Information is provided "as is" and Melexis reserves the right to change the Information at any time and without notice. Therefore, before placing orders and/or prior to designing the Product into a system, users or any third party should obtain the latest version of the relevant information to verify that the information being relied upon is current.
Users or any third party must further determine the suitability of the Product for its application, including the level of reliability required and determine whether it is fit for a particular purpose.
The Information is proprietary and/or confidential information of Melexis and the use thereof or anything described by the Information does not grant, explicitly or implicitly, to any party any patent rights, licenses, or any other intellectual property rights.
This document as well as the Product(s) may be subject to export control regulations. Please be aware that export might require a prior authorization from competent authorities.
The Product(s) are intended for use in normal commercial applications. Unless otherwise agreed upon in writing, the Product(s) are not designed, authorized or warranted to be suitable in applications requiring extended temperature range and/or unusual environmental requirements. High reliability applications, such as medical life-support or life-sustaining equipment are specifically not recommended by Melexis.
The Product(s) may not be used for the following applications subject to export control regulations: the development, production, processing, operation, maintenance, storage, recognition or proliferation of 1) chemical, biological or nuclear weapons, or for the development, production, maintenance or storage of missiles for such weapons: 2) civil firearms, including spare parts or ammunition for such arms; 3) defense related products, or other material for military use or for law enforcement; 4) any applications that, alone or in combination with other goods, substances or organisms could cause serious harm to persons or goods and that can be used as a means of violence in an armed conflict or any similar violent situation.
The Products sold by Melexis are subject to the terms and conditions as specified in the Terms of Sale, which can be found at https://www.melexis.com/en/legal/terms-and-conditions.

This document supersedes and replaces all prior information regarding the Product(s) and/or previous versions of this document.

Melexis NV © - No part of this document may be reproduced without the prior written consent of Melexis. (2019)

ISO/TS 16949 and ISO14001 Certified

6. Revision history table

12/06/2019	Initial release

Table 1