|  |
| --- |
| **Melexis EVK75027** |
| **Software User Manual** |
| Version 1 |

BECOM Systems GmbH

Gutheil-Schoder-Gasse 17

1230 Wien

AUSTRIA

[office.systems@becom-group.com](mailto:office.systems@becom-group.com)

<http://systems.becom-group.com>

**Melexis EVK75027** – Software User Manual

Template No.: 900-519 Rev A

Publication date: March 22, 2019

Subject to change without notice. Errors excepted.

This document is protected by copyright. All rights reserved. No part of this document may be reproduced or transmitted for any purpose in any form or by any means, electronically or mechanically, without expressly written permission by  
BECOM Systems.

Windows is a registered trademark of Microsoft.

Table of Contents

[1 General Information 7](#_Toc4171027)

[1.1 Symbols Used 7](#_Toc4171028)

[2 Overview 8](#_Toc4171029)

[3 Interfacing 9](#_Toc4171030)

[3.1 Control Interface 9](#_Toc4171031)

[3.1.1 Register read 10](#_Toc4171032)

[3.1.2 Register write 12](#_Toc4171033)

[3.1.3 Reset 13](#_Toc4171034)

[3.1.4 Flash Update 15](#_Toc4171035)

[3.1.5 Alive 16](#_Toc4171036)

[3.2 Data Interface 19](#_Toc4171037)

[3.2.1 Image header 20](#_Toc4171038)

[3.3 Secure Shell (SSH) Login 22](#_Toc4171039)

[3.3.1 Change default password 22](#_Toc4171040)

[3.4 Debug UART 22](#_Toc4171041)

[4 Evaluation Kit Features 24](#_Toc4171042)

[4.1 Basic Settings 24](#_Toc4171043)

[4.2 Image Processing Chain 24](#_Toc4171044)

[4.3 Image filtering 25](#_Toc4171045)

[4.3.1 Median Filter 25](#_Toc4171046)

[4.3.2 Bilateral filter 25](#_Toc4171047)

[4.3.3 Sliding Average Filter 25](#_Toc4171048)

[4.3.4 Frame Average Filter 25](#_Toc4171049)

[4.4 Pixel Invalidation 26](#_Toc4171050)

[4.4.1 Distance values 26](#_Toc4171051)

[4.4.2 XYZ values 26](#_Toc4171052)

[4.5 Coordinate System 26](#_Toc4171053)

[4.6 Data Format 26](#_Toc4171054)

[4.6.1 Distances and Amplitudes 27](#_Toc4171055)

[4.6.2 Distances 27](#_Toc4171056)

[4.6.3 XYZ Point Cloud 28](#_Toc4171057)

[4.6.4 Distances and XYZ Point Cloud 28](#_Toc4171058)

[4.6.5 X Coordinate and Amplitudes 29](#_Toc4171059)

[4.6.6 Test mode 29](#_Toc4171060)

[4.6.7 Raw phase data 29](#_Toc4171061)

[4.6.8 Distances and Conficence 30](#_Toc4171062)

[4.6.9 Amplitude 30](#_Toc4171063)

[4.7 ToF Modulation Frequency 30](#_Toc4171064)

[4.8 Frame Rate and Integration Time 30](#_Toc4171065)

[4.9 Distance Offset Calibration 31](#_Toc4171066)

[4.10 Trigger Modes 31](#_Toc4171067)

[4.11 Illumination Power Control 31](#_Toc4171068)

[4.12 Temperature Monitoring 31](#_Toc4171069)

[4.12.1 Illumination Temperature 31](#_Toc4171070)

[4.12.2 Illumination Over-Temperature Protection 32](#_Toc4171071)

[4.12.3 ToF Sensor Temperature 32](#_Toc4171072)

[4.12.4 Application Processor temperature/speed 32](#_Toc4171073)

[4.13 Save Registers 32](#_Toc4171074)

[4.14 Ethernet/IP Settings 32](#_Toc4171075)

[4.14.1 MAC Address 32](#_Toc4171076)

[4.14.2 IP/TCP/UDP Settings 33](#_Toc4171077)

[4.15 Device Status and Error Conditions 33](#_Toc4171078)

[4.16 Reset to Factory Default 34](#_Toc4171079)

[4.17 Firmware Update 34](#_Toc4171080)

[4.18 Firmware Recovery 35](#_Toc4171081)

[4.19 Logging 35](#_Toc4171082)

[4.20 GPOs 35](#_Toc4171083)

[5 Software 36](#_Toc4171084)

[5.1 Demo Application 36](#_Toc4171085)

[5.2 Getting Started Software Development Example 36](#_Toc4171086)

[6 Register Description 37](#_Toc4171087)

[6.1 General 37](#_Toc4171088)

[6.2 User Defined 41](#_Toc4171089)

[6.3 General (2) 41](#_Toc4171090)

[6.4 Device Update 42](#_Toc4171091)

[6.5 Filter Configuration 43](#_Toc4171092)

[6.6 Ethernet configuration 43](#_Toc4171093)

[6.7 General 3 44](#_Toc4171094)

[7 Support 46](#_Toc4171095)

[7.1 General Support 46](#_Toc4171096)

[7.2 Software Packages 46](#_Toc4171097)

[8 Firmware History 47](#_Toc4171098)

[8.1 Version Information 47](#_Toc4171099)

[8.2 Anomalies 47](#_Toc4171100)

[9 Document Revision History 48](#_Toc4171101)

[A List of Figures and Tables 49](#_Toc4171102)

© BECOM Systems GmbH 2019

All Rights Reserved.

The information herein is given to describe certain components and shall not be considered as a guarantee of characteristics.

Terms of delivery and rights of technical change reserved.

We hereby disclaim any warranties, including but not limited to warranties of non-infringement, regarding circuits, descriptions and charts stated herein.

BECOM Systems makes and you receive no warranties or conditions, express, implied, statutory or in any communication with you. BECOM Systems specifically disclaims any implied warranty of merchantability or fitness for a particular purpose.

BECOM Systems takes no liability for any damages and errors causing of the usage of this board. The user of this board is responsible by himself for the functionality of his application. He is allowed to use the board only if he has the qualification. More information is found in the General Terms and Conditions (AGB).

Information

For further information on technology, delivery terms and conditions and prices please contact BECOM Systems [www.becom-group.com](http://www.becom-group.com)

# General Information

This guide applies to the Melexis EVK75027 from BECOM Systems. Follow this guide chapter by chapter to set up and understand your product. If a section of this document only applies to certain camera parts, this is indicated at the beginning of the respective section.

**The document applies to product V0.11.x**

## Symbols Used

This guide makes use of a few symbols and conventions:

|  |  |
| --- | --- |
|  | **Warning** |
| Indicates a situation which, if not avoided, could result in minor or moderate injury and/or property damage or damage to the device. |
| ­ |  |
|  | **Caution** |
| Indicates a situation which, if not avoided, may result in minor damage to the device, in malfunction of the device or in data loss. |
|  |  |
|  | **Note** |
| Notes provide information on special issues related to the device or provide information that will make operation of the device easier. |
|  |  |
|  | **Procedures** |
| A procedure always starts with a headline   1. The number indicates the step number of a certain procedure you are expected to follow.  Steps are numbered sequentially.   This sign ⮞ indicates an expected result of your action. |
|  |  |
|  | **References** |
| This symbol ⮱ indicates a cross reference to a different chapter of this manual or  to an external document. |

# Overview

The document describes the necessary steps and settings to work with the Melexis EVK75027 and describes the firmware dependent interfaces.

The Melexis EVK75027 features the MLX75027 ToF sensor. It was designed with full user’s control of the MLX75123 in mind, which is accomplished by direct access to the companion chip’s register set.

**This document applies to firmware version 0.11.x.**

# Interfacing

The Melexis EVK75027 provides control and data interfaces via Gigabit-Ethernet.

The control interface is used to set and read the configuration of the Melexis EVK75027 via a set of registers. Refer to chapter 6 for a detailed register description.

The data interface provides a continuous stream of raw phase data, or distance and amplitude values depending on the configuration.

## Control Interface

The Melexis EVK75027 can be configured using a TCP/IP connection. For the control interface the Melexis EVK75027 is listening to the following factory default IP settings:

* **IP-Address**: 192.168.0.10
* **Subnet mask**: 255.255.255.0
* **Network protocol**: TCP
* **TCP port**: 10001

|  |  |
| --- | --- |
|  | **Note** |
| The Ethernet IP settings can be configured using the Eth0\_ registers. The changes become active on writing register Eth0Gateway1. |

Once a TCP connection has been established the Melexis EVK75027 can be configured using a dedicated set of command frames. The Melexis EVK75027 answers to each command frame with a dedicated response frame. The following table shows the currently supported command frames:

|  |  |
| --- | --- |
| Command frame | Description |
| Register Read | Used to read one or more consecutive registers |
| Register Write | Used to write one or more consecutive registers |
| Reset | Used to reset/reboot the Melexis EVK75027 |
| Flash Update | Used to either update the firmware or the boot loader |
| Alive | Used to keep the TCP control connection alive. If no command is sent for 10 seconds, the Melexis EVK75027 closes the control interface connection and waits for a new incoming connection request.  Up to 5 concurrent control connections are supported. |

Table 1: Supported command frames

The following section describes each command frame and the expected answer in detail. To be able to communicate with the Melexis EVK75027 the frame must be composed exactly as described.

The following types are used:

* **Uint8**: 8 bit unsigned integer
* **Uint16**: 16 bit unsigned integer
* **Uint32**: 32 bit unsigned integer

|  |  |
| --- | --- |
|  | **Note** |
| Values with ‘0x’ as prefix are hexadecimal values. |

### Register read

Command frame

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Addr | Field | Type | Value | Description |
| 0x00 | Preamble | Uint16  (high byte first) | 0xA1EC | Unique identifier, start of header |
| 0x02 | ProtocolVersion | Uint8 | 0x03 | This description refers to protocol version V3.0 |
| 0x03 | Command | Uint8 | 0x03 | Command code for read registers |
| 0x04 | SubCommand | Uint8 |  | Ignored |
| 0x05 | Status | Uint8 |  | Ignored |
| 0x06 | Flags | Uint16 | Refer to table | Optional flags |
| 0x08 | Length | Uint32 (high byte first) | <# of bytes to read> | Number of bytes to read. Must be a multiple of two. The length divided by two represents the # of registers to read. |
| 0x0C | RegisterAddress | Uint16 (high byte first) | <Register Address> | Start register address for read command |
| 0x0E | HeaderData2 | Uint8 |  | Ignored |
| 0x0F | HeaderData3 | Uint8 |  | Ignored |
| 0x10 | Reserved (42 bytes) | Uint8[] |  | Ignored |
| 0x3A | DataCrc32 | Uint32 (high byte first) | 0x0 | No data present after header. |
| 0x3E | HeaderCrc16 | Uint16 (high byte first) | <CRC16 checksum> | Checksum over 60 bytes of Header: 0x02 – 0x3D 1) |

Table 2: Register read command frame

Note 1): For the CRC16 calculation the CRC-CCITT is used (Polynom: 0x1021, start value: 0). Please ask the BECOM Systems support for an implementation example of the CRC-CCITT.

Response frame

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Addr | Field | Type | Value | Description |
| 0x00 | Preamble | Uint16 (high byte first) | 0xA1EC | Unique identifier, start of header |
| 0x02 | ProtocolVersion | Uint8 | 0x03 | This document refers to protocol version V3.0 |
| 0x03 | Command | Uint8 | 0x03 | Command code for read registers |
| 0x04 | SubCommand | Uint8 |  | Ignore |
| 0x05 | Status | Uint8 | Refer to table | Result code |
| 0x06 | Flags | Uint16 | Refer to table | Optional flags |
| 0x08 | Length | Uint32 (high byte first) | <# of bytes read> | The number of bytes read (length of <Data> in bytes). The length divided by two represents the # of registers read. |
| 0x0C | RegisterAddress | Uint16 (high byte first) | <Register Address> | Start register address of read data |
| 0x0E | HeaderData2 | Uint8 |  | Ignored |
| 0x0F | HeaderData3 | Uint8 |  | Ignored |
| 0x10 | Reserved (42 bytes) | Uint8[] |  | Ignored |
| 0x3A | DataCrc32 | Uint32 (high byte first) | <CRC32 checksum> | Checksum over <Data> 2) |
| 0x3E | HeaderCrc16 | Uint16 (high byte first) | <CRC16 checksum> | Checksum over 60 bytes of Header: 0x02 – 0x3D 1) |
| 0x40 | Data | Uint16[] (high byte first) | <result data> | Result: One or more 16 bit values |

Table 3: Register read response frame

Note 1): For the CRC16 calculation the CRC-CCITT is used (Polynom: 0x1021, start value: 0). Please ask the BECOM Systems support for an implementation example of the CRC-CCITT.

Note 2): For the CRC32 calculation the CRC-32 is used (Polynom: 0x04C11DB7, start value: 0xFFFFFFFF). Please ask the BECOM Systems support for an implementation example of the CRC-32.

Flags

|  |  |
| --- | --- |
| Flags | Description |
| Bit 0 | 1: Ignore DataCrc32 |

Table 4: Register read flag description

Result codes

|  |  |
| --- | --- |
| Status | Description |
| 0x00 | Ok |
| 0x0D | Invalid handle (internal error) |
| 0x0F | Illegal write: The Address is not valid or the register is not write-enabled |
| 0x10 | Illegal read: The Address is not valid (deprecated, replaced by 17) |
| 0x11 | Register end reached |
|  |  |
| 0xFA | Length exceeds maximum file size (not enough memory for file download) |
| 0xFB | HeaderCrc16 mismatch |
| 0xFC | DataCrc32 mismatch |
| 0xFD | Length invalid: Cannot be equal 0 |
| 0xFE | Length invalid: Cannot be grater 0 |
| 0xFF | Unknown command |

Table 5: Result codes

### Register write

Command frame

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Addr | Field | Type | Value | Description |
| 0x00 | Preamble | Uint16 (high byte first) | 0xA1EC | Unique identifier, start of header |
| 0x02 | ProtocolVersion | Uint8 | 0x03 | This document refers to version V3.0 |
| 0x03 | Command | Uint8 | 0x04 | Command code for write registers |
| 0x04 | SubCommand | Uint8 |  | Ignored |
| 0x05 | Status | Uint8 |  | Ignored |
| 0x06 | Flags | Uint16 | Refer to table | Optional flags |
| 0x08 | Length | Uint32 (high byte first) | <# of bytes to write> | The number of bytes to write. Must be a multiple of two and match length of <Data> in bytes. The length divided by two represents the # of registers to write. |
| 0x0C | RegisterAddress | Uint16 (high byte first) | <Register Address> | Start register address for write command |
| 0x0E | HeaderData2 | Uint8 |  | Ignored |
| 0x0F | HeaderData3 | Uint8 |  | Ignored |
| 0x10 | Reserved (42 bytes) | Uint8[] |  | Ignored |
| 0x3A | DataCrc32 | Uint32 (high byte first) | <CRC32 checksum> | Checksum over <Data> 2) |
| 0x3E | HeaderCrc16 | Uint16 (high byte first) | <CRC16 checksum> | Checksum over 60 bytes of Header: 0x02 – 0x3D 1) |
| 0x40 | Data | Uint16[] (high byte first for each register value) | <data to write> | One or more 16 bit values in a stream that should be written |

Table 6: Register write command frame

Note 1): For the CRC16 calculation the CRC-CCITT is used (Polynom: 0x1021, start value: 0). Please ask the BECOM Systems support for an implementation example of the CRC-CCITT.

Note 2): For the CRC32 calculation the CRC-32 is used (Polynom: 0x04C11DB7, start value: 0xFFFFFFFF). Please ask the BECOM Systems support for an implementation example of the CRC-32.

Response frame

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Addr | Field | Type | Value | Description |
| 0x00 | Preamble | Uint16  (high byte first) | 0xA1EC | Unique identifier, start of header |
| 0x02 | ProtocolVersion | Uint8 | 0x03 | This document refers to version V3.0 |
| 0x03 | Command | Uint8 | 0x04 | Command code for write registers |
| 0x04 | SubCommand | Uint8 |  | Ignored |
| 0x05 | Status | Uint8 | Refer to table | Result code |
| 0x06 | Flags | Uint16 | Refer to table | Optional flags |
| 0x08 | Length | Uint32 (high byte first) | 0 | No <Data> present |
| 0x0C | RegisterAddress | Uint8 (high byte first) | <Register Address> | Same as in sent command |
| 0x0E | HeaderData2 | Uint8 |  | Ignored |
| 0x0F | HeaderData3 | Uint8 |  | Ignored |
| 0x10 | Reserved (42 bytes) | Uint8[] |  | Ignored |
| 0x3A | DataCrc32 | Uint32 (high byte first) | 0x0 | No data present after header. |
| 0x3E | HeaderCrc16 | Uint16 (high byte first) | <CRC16 checksum> | Checksum over 60 bytes of Header: 0x02 – 0x3D 1) |

Table 7: Register write response frame

Note 1): For the CRC16 calculation the CRC-CCITT is used (Polynom: 0x1021, start value: 0). Please ask the BECOM Systems support for an implementation example of the CRC-CCITT.

Flags

|  |  |
| --- | --- |
| Flags | Description |
| Bit 0 | 1: Ignore DataCrc32 |

Table 8: Register write flag description

Result codes

Please refer to Table 3‑5.

### Reset

Command frame

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Addr | Field | Type | Value | Description |
| 0x00 | Preamble | Uint16 (high byte first) | 0xA1EC | Unique identifier, start of header |
| 0x02 | ProtocolVersion | Uint8 | 0x03 | This document refers to version V3.0 |
| 0x03 | Command | Uint8 | 0x07 | Command code for reset |
| 0x04 | SubCommand | Uint8 |  | Ignored |
| 0x05 | Status | Uint8 |  | Ignored |
| 0x06 | Flags | Uint16 | Refer to table | Optional flags |
| 0x08 | Length | Uint32 (high byte first) | 0x0 | No <Data> present |
| 0x0C | HeaderData0 | Uint8 |  | Ignored |
| 0x0D | HeaderData1 | Uint8 |  | Ignored |
| 0x0E | HeaderData2 | Uint8 |  | Ignored |
| 0x0F | HeaderData3 | Uint8 |  | Ignored |
| 0x10 | Reserved (42 bytes) | Uint8[] |  | Ignored |
| 0x3A | DataCrc32 | Uint32 (high byte first) | 0x0 | No data present after header. |
| 0x3E | HeaderCrc16 | Uint16 (high byte first) | <CRC16 checksum> | Checksum over 60 bytes of Header: 0x02 – 0x3D 1) |

Table 9: Reset command frame

Note 1): For the CRC16 calculation the CRC-CCITT is used (Polynom: 0x1021, start value: 0). Please ask the BECOM Systems support for an implementation example of the CRC-CCITT.

Response frame

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Addr | Field | Type | Value | Description |
| 0x00 | Preamble | Uint16 (high byte first) | 0xA1EC | Unique identifier, start of header |
| 0x02 | ProtocolVersion | Uint8 | 0x03 | This document refers to version V3.0 |
| 0x03 | Command | Uint8 | 0x07 | Command code for reset |
| 0x04 | SubCommand | Uint8 |  | Ignored |
| 0x05 | Status | Uint8 | Refer to table | Result code |
| 0x06 | Flags | Uint16 | Refer to table | Optional flags |
| 0x08 | Length | Uint32 (high byte first) | 0x0 | No <Data> present |
| 0x0C | HeaderData0 | Uint8 |  | Ignored |
| 0x0C | HeaderData1 | Uint8 |  | Ignored |
| 0x0E | HeaderData2 | Uint8 |  | Ignored |
| 0x0F | HeaderData3 | Uint8 |  | Ignored |
| 0x10 | Reserved (42 bytes) | Uint8[] |  | Ignored |
| 0x3A | DataCrc32 | Uint32 (high byte first) | 0x0 | No data present after header. |
| 0x3E | HeaderCrc16 | Uint16 (high byte first) | <CRC16 checksum> | Checksum over 60 bytes of Header: 0x02 – 0x3D 1) |

Table 10: Reset response frame

Note 1): For the CRC16 calculation the CRC-CCITT is used (Polynom: 0x1021, start value: 0). Please ask the BECOM Systems support for an implementation example of the CRC-CCITT.

Flags

|  |  |
| --- | --- |
| Flags | Description |
|  | Currently no flags defined for this command |

Table 11: Reset flag description

Result codes

Please refer to Table 3‑5.

### Flash Update

Command frame

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Addr | Field | Type | Value | Description |
| 0x00 | Preamble | Uint16 (high byte first) | 0xA1EC | Unique identifier, start of header |
| 0x02 | ProtocolVersion | Uint8 | 0x03 | This document refers to protocol version V3.0 |
| 0x03 | Command | Uint8 | 🡪 | 0x0B: Flash boot loader  0x0C: Flash Application  0x21: Flash Lens Calibration File  0x22: Wiggling Calibration File |
| 0x04 | SubCommand | Uint8 | Refer to table | Indicates which flash to write to |
| 0x05 | Status | Uint8 |  | Ignored |
| 0x06 | Flags | Uint16 | Refer to table | Optional flags 3) |
| 0x08 | Length | Uint32 (high byte first) | <# of bytes to write> | The size of the binary file to flash |
| 0x0C | FlashAddress | Uint32 (high byte first) | <Flash Address> | Ignored |
| 0x10 | Reserved (42 bytes) | Uint8[] |  | Ignored |
| 0x3A | DataCrc32 | Uint32 (high byte first) | <CRC32 checksum> | Checksum over <Data> 2) |
| 0x3E | HeaderCrc16 | Uint16 (high byte first) | <CRC16 checksum> | Checksum over 60 bytes of Header: 0x02 – 0x3D 1) |
| 0x40 | Data | Uint8[] | <binary loader file> | The file to flash as a binary byte stream |

Table 12: Flash update command frame

Note 1): For the CRC16 calculation the CRC-CCITT is used (Polynom: 0x1021, start value: 0). Please ask the BECOM Systems support for an implementation example of the CRC-CCITT.

Note 2): For the CRC32 calculation the CRC-32 is used (Polynom: 0x04C11DB7, start value: 0xFFFFFFFF). Please ask the BECOM Systems support for an implementation example of the CRC-32.

Note 3): The DataCrc32 is mandatory, the appropriate flag must be set to 0.

Response frame

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Addr | Field | Type | Value | Description |
| 0x00 | Preamble | Uint16 (high byte first) | 0xA1EC | Unique identifier, start of header |
| 0x02 | ProtocolVersion | Uint8 | 0x03 | This document refers to protocol version V3.0 |
| 0x03 | Command | Uint8 |  | Identical to command frame |
| 0x04 | SubCommand | Uint8 | Refer to table | Indicates which flash to write to |
| 0x05 | Status | Uint8 | Refer to table | Result code |
| 0x06 | Flags | Uint16 | Refer to table | Optional flags |
| 0x08 | Length | Uint32 (high byte first) | 0x0 | No <Data> present |
| 0x0C | HeaderData0 | Uint8 |  | Ignored |
| 0x0D | HeaderData1 | Uint8 |  | Ignored |
| 0x0E | HeaderData2 | Uint8 |  | Ignored |
| 0x0F | HeaderData3 | Uint8 |  | Ignored |
| 0x10 | Reserved (42 bytes) | Uint8[] |  | Ignored |
| 0x3A | DataCrc32 | Uint32 (high byte first) | 0x0 | No data present after header. |
| 0x3E | HeaderCrc16 | Uint16 (high byte first) | <CRC16 checksum> | Checksum over 60 bytes of Header: 0x02 – 0x3D 1) |

Table 13: Flash update response frame

Note 1): For the CRC16 calculation the CRC-CCITT is used (Polynom: 0x1021, start value: 0). Please ask the BECOM Systems support for an implementation example of the CRC-CCITT.

Subcommand

|  |  |
| --- | --- |
| SubCommand | Description |
| Ignored | Always ignored |

Table 14: Flash update subcommand description

Flags

|  |  |
| --- | --- |
| Flags | Description |
| Bit 0 | 1: Ignore DataCrc32 |

Table 15: Flash update flag description

Result codes

Please refer to Table 3‑5.

### Alive

Command frame

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Addr | Field | Type | Value | Description |
| 0x00 | Preamble | Uint16 (high byte first) | 0xA1EC | Unique identifier, start of header |
| 0x02 | ProtocolVersion | Uint8 | 0x03 | This document refers to protocol version V3.0 |
| 0x03 | Command | Uint8 | 0xFE | Command code for ‘Alive message |
| 0x04 | SubCommand | Uint8 |  | Ignored |
| 0x05 | Status | Uint8 |  | Ignored |
| 0x06 | Flags | Uint16 | Refer to table | Optional flags |
| 0x08 | Length | Uint32 | 0x0 | No <Data> present |
| 0x0C | HeaderData0 | Uint8 |  | Ignored |
| 0x0D | HeaderData1 | Uint8 |  | Ignored |
| 0x0E | HeaderData2 | Uint8 |  | Ignored |
| 0x0F | HeaderData3 | Uint8 |  | Ignored |
| 0x10 | Reserved (42 bytes) | Uint8[] |  | Ignored |
| 0x3A | DataCrc32 | Uint32 (high byte first) | 0x0 | No data present after header. |
| 0x3E | HeaderCrc16 | Uint16 (high byte first) | <CRC16 checksum> | Checksum over 60 bytes of Header: 0x02 – 0x3D 1) |

Table 16: Alive command frame

Note 1): For the CRC16 calculation the CRC-CCITT is used (Polynom: 0x1021, start value: 0). Please ask the BECOM Systems support for an implementation example of the CRC-CCITT.

Response frame

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Addr | Field | Type | Value | Description |
| 0x00 | Preamble | Uint16 (high byte first) | 0xA1EC | Unique identifier, start of header |
| 0x02 | ProtocolVersion | Uint8 | 0x03 | This document refers to protocol version V3.0 |
| 0x03 | Command | Uint8 | 0xFE | Command code for ‘Alive message’ |
| 0x04 | SubCommand | Uint8 |  | Indicates which flash to write to |
| 0x05 | Status | Uint8 | Refer to table | Result code |
| 0x06 | Flags | Uint16 | Refer to table | Optional flags |
| 0x08 | Length | Uint32 (high byte first) | 0x0 | No <Data> present |
| 0x0C | HeaderData0 | Uint8 |  | Ignored |
| 0x0D | HeaderData1 | Uint8 |  | Ignored |
| 0x0E | HeaderData2 | Uint8 |  | Ignored |
| 0x0F | HeaderData3 | Uint8 |  | Ignored |
| 0x10 | Reserved (42 bytes) | Uint8[] |  | Ignored |
| 0x3A | DataCrc32 | Uint32 (high byte first) | 0x0 | No data present after header. |
| 0x3E | HeaderCrc16 | Uint16 (high byte first) | <CRC16 checksum> | Checksum over 60 bytes of Header: 0x02 – 0x3D 1) |

Table 17: Alive response frame

Note 1): For the CRC16 calculation the CRC-CCITT is used (Polynom: 0x1021, start value: 0). Please ask the BECOM Systems support for an implementation example of the CRC-CCITT.

Flags

|  |  |
| --- | --- |
| Flags | Description |
|  | Currently no flags defined for this command |

Table 18: Alive flag description

Result codes:

Please refer to Table 3‑5.

## Data Interface

A UDP stream delivers distance and amplitude data from the Melexis EVK75027. Each UDP packet contains a header and by default 1400 bytes of data (Ethernet, IP, and UDP headers are not shown in Figure 3‑1). This amount is configurable using register Eth0UdpPacketSize.

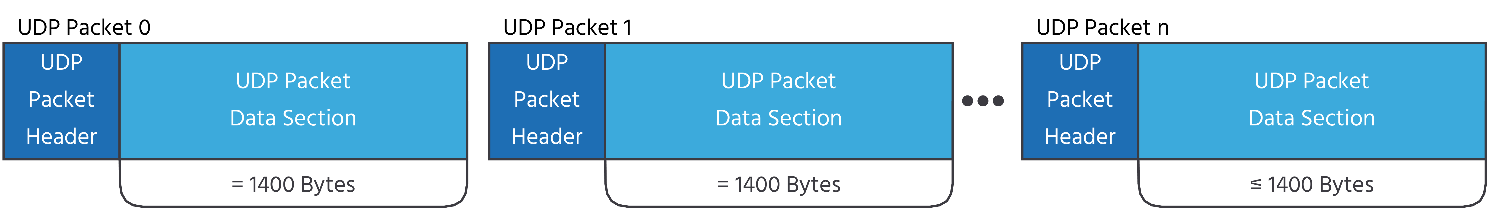


Figure 3‑1: UDP streaming data format

The following types are used in the data streaming protocol:

* **Uint8**: 8 bit unsigned integer
* **Uint16**: 16 bit unsigned integer
* **Uint32**: 32 bit unsigned integer

|  |  |
| --- | --- |
|  | **Note** |
| Values with ‘0x’ as prefix are hexadecimal values. |

The UDP streaming is enabled by factory default. The Melexis EVK75027 streams to the following IP settings:

* **IP-Address**: Multicast address 224.0.0.1
* **UDP port**: 10002

|  |  |
| --- | --- |
|  | **Note** |
| The UDP stream settings can be configured using the Eth0\_ registers. |

As multicast is used more than one can receive the stream within the same subnet at the same time. The client has to join the appropriate multi cast group and open the port 10002 on its local network interface card (NIC) where the Melexis EVK75027 is connected to. The receiver should receive the stream and interpret it as the following protocol description shows.

|  |  |
| --- | --- |
|  | **Note** |
| Be aware that a multicast stream may slow down your Ethernet network as the stream must be spread to all active links of switches/hubs and routers.  Also, be aware that firewalls may block multicast UDP packets! |

The current protocol version is **1**.

Each image transmitted on the UDP stream is split into packets of max. 1432 bytes length (except the last which may be smaller). Each packet consists of a 32 byte packet header and up to 1400 bytes of image data section (refer to Figure 3‑1).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Addr | Field | Type | Value | Description |
| 0x00 | Version | Uint16  (high byte first) | 0x0001 | Protocol version |
| 0x02 | FrameCounter | Uint16 (high byte first) |  | Continuous frame counter. On an overrun it restarts at 0. |
| 0x04 | PacketCounter | Uint16 (high byte first) |  | Actual packet #. The frame data must be recomposed in order of the packet #. |
| 0x06 | DataLength | Uint16 (high byte first) |  | Length of the image data section of the current packet. |
| 0x08 | FrameSize | Uint32 (high byte first) |  | Size of the image data. It may be used to calculate the expected # of packets for a frame. |
| 0x0C | PacketCRC32 | Uint16 (high byte first) |  | CRC32 checksum over the entire packet (pos 0 to pos n) 1) |
| 0x10 | Flags | Uint32 | Refer to Table 20 | Optional flags |
| 0x14 | Reserved |  |  | Reserved for future use |
| 0x20 | ImageData |  |  | Image data section |

Table 19: UDP packet header

Note 1): For the CRC32 calculation the CRC-32 is used (Polynom: 0x04C11DB7, start value: 0xFFFFFFFF). Please ask the BECOM Systems support for an implementation example of the CRC-32.

Flags

|  |  |
| --- | --- |
| Flags | Description |
| Bit 0 | 1: Ignore DataCrc32 |

Table 20: UDP packet header flag description

### Image header

The image data itself is split into a 64 byte image header and the image data section. The format of the image data depends on the selected image format and is described in chapter 4.3. Below you can find the format of the 64 byte image header.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Addr | Field | Type | Value | Description |
| 0x00 | Reserved | Uint16 | 0xFFFF |  |
| 0x02 | HeaderVersion | Uint16  (high byte first) | 0x0003 | Current header version |
| 0x04 | ImageWidth | Uint16  (high byte first) |  | Width of the image in pixels. Depends on binning settings. |
| 0x06 | ImageHeight | Uint16  (high byte first) |  | Height of the image in pixels. Depends on binning settings. |
| 0x08 | NofChannels | Uint8 |  | Nof data channels. Depends on the image format |
| 0x09 | BytesPerPixel | Uint8 | 0x02 | Bytes per pixel of the image data. |
| 0x0A | ImageFormat | Uint16  (high byte first) |  | The content is the same as in the register *ImageDataFormat*). |
| 0x0C | Timestamp | Uint32 (high byte first) |  | Timestamp of the actual image in µs |
| 0x10 | FrameCounter | Uint16  (high byte first) |  | Continuous frame counter. On an overrun it restarts at 0. |
|  |  |  |  |  |
| 0x1A | MainTemp | Uint8 |  | ToF sensor temperature in °C + 50. Decrement this field by 50 to get the current ToF sensor temperature. |
| 0x1B | LEDtemp | Uint8 |  | Average LEDs temperature in °C + 50. Decrement this field by 50 to get the current average temperature of LEDs. |
| 0x1C | FirmwareVersion | Uint16  (high byte first) |  | Content of the register *FirmwareInfo* |
| 0x1E | MagicV31 | Uint16 (high byte first) | 0x3331 | These magic bytes indicate that header version is 3.1 |
| 0x20 | IntegrationTime | Uint16 (high byte first) |  | Integration time in us. |
| 0x22 | ModFreq | Uint16 (high byte first) |  | Modulation frequency with resolution 10 kHz (e.g., a value of 0x1234 means frequency 46.6 MHz) |
| 0x24 | Temp3 | Uint8 |  | Sensor board temperature sensor in °C + 50. Decrement this field by 50 to get the current temperature.  A value of 0xFF means sensor error. |
|  |  |  |  |  |
| 0x30 | RawPhaseContent | Uint32 (high byte first) |  | **If ImageFormat(3:10) == 24:**  Bits 0…3: Phase 0  Bits 4…7: Phase 1  Bits 8…11: Phase 2  Bits 12…15: Phase 3  Bits 16…19: Phase 4  Bits 20…23: Phase 5  Bits 24…27: Phase 6  Bits 28…31: Phase 7  Each 4 bits contain a content number which describes the raw phase:  0…1 common mode bit + 11 signed bits, aligned to LSB  1…12 signed bits, aligned to LSB  2…1 common mode bit + 11 unsigned bits, aligned to LSB  3…12 unsigned bits, aligned to LSB |
|  |  |  |  |  |
| 0x3E | CRC16 | Uint16  (high byte first) |  | CRC16 checksum over the header without the first two bytes and the CRC16 checksum itself (addr 0x02 to addr 0x3D) 1) |

Table 21: Image data header

Note 1): For the CRC16 calculation the CRC-CCITT is used (Polynom: 0x1021, start value: 0). Please ask the BECOM Systems support for an implementation example of the CRC-CCITT.

## Secure Shell (SSH) Login

The Melexis EVK75027 features an OpenSSH server listening to TCP port 22.

|  |  |  |
| --- | --- | --- |
|  | Root account | User account |
| Username | root | user |
| Default password | root | user |

Table 22: Default login credentials

### Change default password

|  |  |
| --- | --- |
|  | Change default password   1. Log in via SSH, e.g., ssh root@192.168.0.10 2. Type passwd root or passwd user 3. Supply the new password for two times 4. Copy the file /etc/shadow (containing the encrypted passwords) to the non-volatile settings partition, to be restored again on next reboot: cp /etc/shadow /mnt/settings |

## Debug UART

|  |  |
| --- | --- |
|  | **Note** |
| Using the Debug UART is optional. |

The Melexis EVK75027 features a debug UART, which is the primary debug interface for the boot loader as well as the Linux kernel.

The Debug UART is available via a Mini-USB-connector, with a UART-to-USB converter behind. To be able to access the serial terminal via the Debug UART, you need an appropriate driver installed in your OS for the FTDI FT234 device.

|  |  |
| --- | --- |
|  | **Windows OS Device Driver Download** |
|  | ⮱ http://www.ftdichip.com/Drivers/VCP.htm |

Most Linux distributions come with an appropriate driver and create a device node /dev/ttyUSB... dynamically.

Additionally, one needs a serial terminal emulator, e.g., Minicom or C-Kermit for Linux, or TeraTerm for Windows OS. The emulator has to be configured with the following settings:

|  |  |
| --- | --- |
| Baud rate | 115200 |
| Data bits | 8 |
| Parity | none |
| Stop bits | 1 |
| Flow control | none |

Table 23: Debug UART settings

The Debug UART also allows to log in to the evaluation kit’s Linux OS. Please see chapter 3.3 for the default login and how to change it.

# Evaluation Kit Features

## Basic Settings

The Melexis EVK75027 comes up according to the reset (default) values as described in the register description section (refer to chapter 6).

## Image Processing Chain

The following flow diagram shows the image processing chain of the Melexis EVK75027 for the distance data. For the amplitude data currently no post processing will be performed.

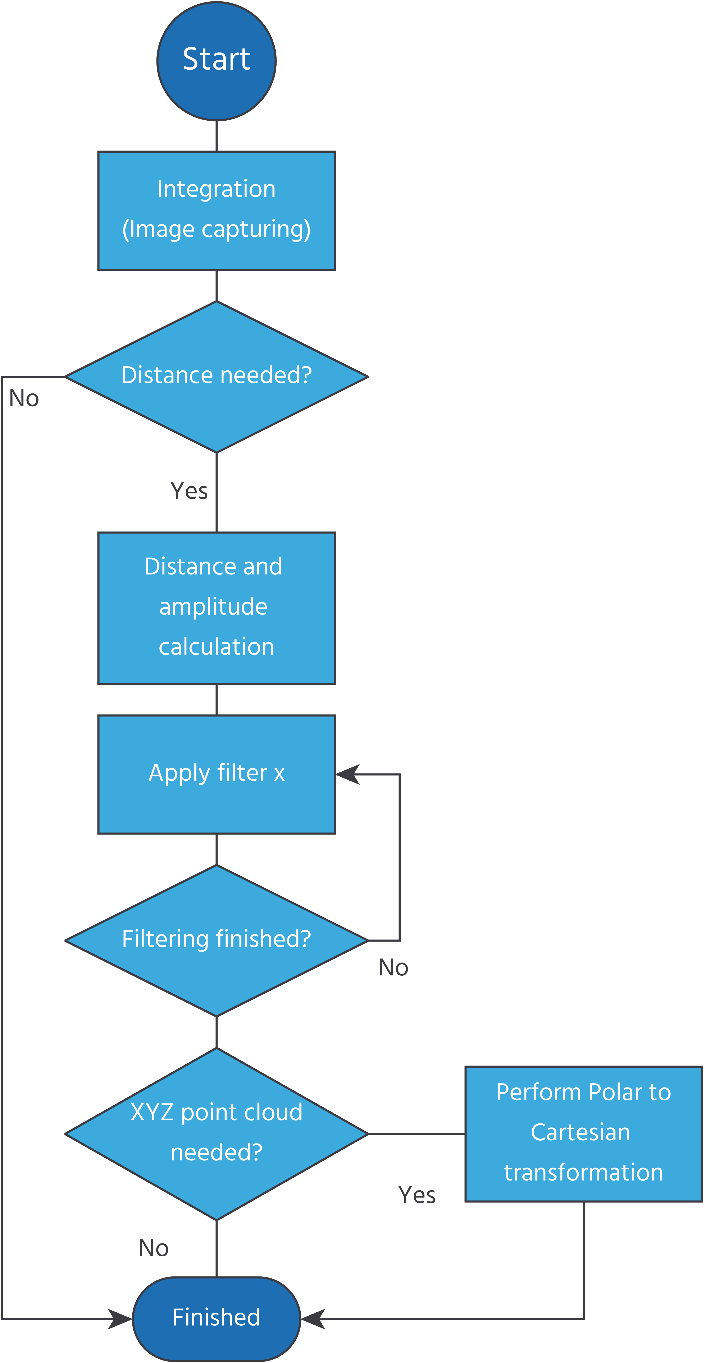


Figure 4‑1: Image processing flow

## Image filtering

After the distance and amplitude calculation some filters can be applied to the distance data. The amplitude data will be left unchanged. Each of the filter provides one or more configuration parameters. The iteration count for each filter can also be configured. The filters can be enabled or disabled by writing the ImgProcConfig and ***ImgProcConfig2*** registers. Enabling more than one filter is possible but each added filter reduces the maximum achievable frame rate (as does the number of iterations).

The filters are applied in the following order:

1. Frame Average filter
2. Sliding Average filter
3. Median filter
4. Bilateral filter

### Median Filter

A 3x3 median filter can be applied.

Register: FilterMedianConfig

The number of iterations is configurable.

### Bilateral filter

Registers: FilterBilateralConfig, FilterBilateralConfig2

Configuration options are R (Width of range kernel), S (Width of spatial kernel), number of iterations, and window size.

### Sliding Average Filter

Register: FilterSLAFconfig

A sliding average filter over up to 255 frames can be applied. The number of frames is configurable. Only the distance data will be averaged. The amplitude data will be left unchanged. An increasing number of frames will not decrease the frame rate but may add blurring effects.

### Frame Average Filter

Register: ***FilterFrameAverageConfig***

A frame average filter over up to 15 frames can be applied. The number of frames is configurable. Only the distance data will be averaged. The amplitude data is left unchanged.

The frame rate of the data interface will be divided by the number of configured frames to be averaged, e.g., if the evaluation kit is configured to 40 frames per second, and the frame average filter with number 4 is used, the resulting output frame rate will be 10.

## Pixel Invalidation

The Melexis EVK75027 provides an on-board check for invalid pixels:

* Underexposed pixels: The amplitude is too low for the distance value to be trustworthy. The Melexis EVK75027 sets the pixel distance to the maximum value. The threshold is set via register ***ConfidenceThresLow***.
* Overexposed pixels: The amplitude is too high for the distance value to be trustworthy. The Melexis EVK75027 sets the pixel distance to the minimum value. The threshold is set via register ***ConfidenceThresHigh***.

### Distance values

If the amplitude of the reflected signal is below a threshold (underexposure) the distance value of the appropriate pixel will be set to 0xFFFF. If the amplitude is too high (overexposure) the distance value will be set to 0x0000.

### XYZ values

If the amplitude of the reflected signal is below a threshold (underexposure) the X value of the appropriate pixel is set to 32767 (0x7FFF), i.e., the largest positive Int16 value. Y and Z values are set to 0.

If the amplitude of the reflected signal is above a threshold (overexposure) the X of the appropriate pixel is set to 0. Y and Z values are set to 0 as well.

## Coordinate System

The default coordinate system starts pixel numbering in the upper right corner of the pixel array, seen from the evaluation kit’s point of view.

## Data Format

The evaluation kit provides up to 8 data channels. The meaning of each data channel depends on the selected data format.

The stream usually starts with pixel #0. The maximum array size for one channel is 640x480 pixels. The output size can be decreased by binning settings for some modes.

Which image format will be transferred can be selected by the register ImageDataFormat. The following sections describe each of the supported formats in detail. Only the data section which contains the image data of the transferred frame will be described. For information about the packet format and meta-data please refer to chapter 3.2.

### Distances and Amplitudes

Format number for ***ImageDataFormat*** register: 0

In this mode the distances and amplitudes will be transferred in progressive mode, first the distance array, then the amplitude array.

The **distances** are coded in **millimeters** as **Uint16**, the **amplitudes** also as **Uint16**.



Figure 4‑2: Data stream example of Distance and Amplitude data for 320x240 pixel

### Distances

Format number for ***ImageDataFormat*** register: 12

In this mode a single array with distances is transferred.

The **distances** are coded in **millimeters** as **Uint16**.

### XYZ Point Cloud

Format number for ***ImageDataFormat*** register: 3

In this mode the XYZ point cloud will be transferred in progressive mode, first the X coordinate array (channel 0) then the Y (channel 1) and Z (channel 2) coordinate array.

The **coordinates** are coded in **millimeters** as **Int16.**



Figure 4‑3: Data stream of 160x120 pixel XYZ Point Cloud

|  |  |
| --- | --- |
|  | **Note** |
| XYZ data will not be reliable until a proper Lens Calibration is programmed onto the evaluation kit. After programming the Lens Calibration, set the lens identifier (= horizontal opening angle; default: 110) in register HardwareConfiguration. |

### Distances and XYZ Point Cloud

Format number for ***ImageDataFormat*** register: 9

In this mode the distances and the XYZ point cloud will be transferred in progressive mode, first the distances array (channel 0), then X (channel 1), Y (channel 2), and Z (channel 3) coordinate arrays.

The **distances** are coded in millimeters as Uint16. The **coordinates** are coded in **millimeters** as **Int16**.

|  |  |
| --- | --- |
|  | **Note** |
| XYZ data will not be reliable until a proper Lens Calibration is programmed onto the evaluation kit. After programming the Lens Calibration, set the lens identifier (= horizontal opening angle; default: 110) in register HardwareConfiguration. |

### X Coordinate and Amplitudes

Format number for ***ImageDataFormat*** register: 10

In this mode a single coordinate array, more specifically, the one belonging to the optical axis of the evaluation kit (X), is transferred in channel 0, as well as the amplitudes (channel 1).

**Coordinate** values are coded in **millimeters** as **Int16**. The **amplitudes** are coded as **Uint16**.

|  |  |
| --- | --- |
|  | **Note** |
| XYZ data will not be reliable until a proper Lens Calibration is programmed onto the evaluation kit. After programming the Lens Calibration, set the lens identifier (= horizontal opening angle; default: 110) in register HardwareConfiguration.  Each binning mode requires a separate Lens Calibration to be programmed! |

### Test mode

Format number for ***ImageDataFormat*** register: 11

In this mode four arrays with test data are transferred in progressive order. Each array has size 640x480 pixels.

* First array: Uint16 value = Pixel Index
* Second array: Uint16 value always constant ‘0xbeef’
* Third array: Uint16 value = (Pixel Index)2
* Fourth array: Uint16 value always constant ‘0x0000’

### Raw phase data

Format number for ***ImageDataFormat*** register: 24

In this mode the raw phase data will be transferred in progressive mode. There are 1 to 8 channels that contain phase data. How many channels will be actually transferred depends on register NofPhases.

### Distances and Conficence

Format number for ***ImageDataFormat*** register: 26

In this mode the distances and confidence data will be transferred in progressive mode, first the distance array, then the confidence array.

This mode is designed to offer increased framerate at full VGA resolution with a predefined configuration:

* FPPN correction is applied if enabled
* Wiggling correction is applied if enabled
* Distance offset is applied
* Temperature compensation is applied
* Confidence data corresponds to amplitude data reduced to 8 bit resolution.
* UDP message size is fixed to 51232 byte, frame header is sent as a separate 96 byte message

Any other image processing options (e.g. filters) are ignored.

Both arrays have size 640x480 pixels. Binning must be disabled.

The **distances** are coded in **millimeters** as **Uint16**, confidence as **Uint8**.

### Amplitude

Format number for ***ImageDataFormat*** register: 27

In this mode a single array with amplitudes is transferred.

The **amplitudes** are sent as **Uint16**.

## ToF Modulation Frequency

The modulation frequency of the illumination is set to 40 MHz per default. Other modulation frequencies can be set using register ModulationFrequency. Be aware that this also changes the ambiguity range of the camera. On writing these registers, if inexact values are supplied, the camera searches for the next possible modulation frequency automatically.

## Frame Rate and Integration Time

The frame rate and the integration time of the ToF sensor can be set by using the registers Framerate and IntegrationTime.

The combination of frame rate and integration time influences the input current as well as the dissipated heat and will be characterized by the *“Frame rate Integration Time Product”* (FITP) which has been defined as follows:

|  |  |
| --- | --- |
|  | **Caution** |
| Be careful in setting different integration times and frame rate combinations. Not all combinations are possible! Without appropriate cooling the device may be damaged! Refer to the Hardware User Manual for more information. |

## Distance Offset Calibration

There is an absolute offset in millimeters all distances are corrected with. The absolute offset is stored in register ***DistOffset0***. You can modify this value by a register write.

The evaluation kit has also a built-in offset calibration function, which is described in the following procedure. You let the evaluation kit know the real distance and it will calculate the correct absolute offset. It uses a square of 4x4 pixels in the center of the distance image. See the register description of registers ***RealWorldXCoordinate*** and ***CalibrationCommand***.

## Trigger Modes

The default mode of the MLX75027 is “video mode”, where the kit streams continuously with configured frame rate. To use manual frame triggering, you have to put the MLX75027 into “manual trigger mode” via register Mode0 bit 0.

In this mode, a frame capture on the ToF sensor is triggered via register Mode0, where the application processor sends a trigger signal to the MLX75027 .

## Illumination Power Control

The output power of the illumination can be controlled using register ***IllPower***. The register takes a percent value between 0 and 100, whereas 1 is the least output power and 100 the highest. A value of 0 disables the illumination completely.

## Temperature Monitoring

### Illumination Temperature

The firmware constantly reads the values of the temperature sensor on the illumination board. The temperature value is provided via register LedboardTemp as well as in each image header (see chapter 3.2).

If no LED temperature could be read, Bit[3] in the Status register is set.

### Illumination Over-Temperature Protection

The Melexis EVK75027 firmware has a built-in monitoring for over-temperature condition of the illumination board. If this temperature exceeds 70°C, the evaluation kit will automatically stop illumination and streaming, until temperature is below 68°C.

During over-temperature condition, Bit[9] of the Status register is set.

The maximum temperature can be set via register MaxLedTemp.

### ToF Sensor Temperature

The value of MLX75027 built in temperature sensor is provided in register MainboardTemp, as well as in each image header (see chapter 3.2).

The value of the temperature sensor near the MLX75027 ToF sensor is provided in register BaseboardTemp, as well as in each image header (see chapter 3.2).

### Application Processor temperature/speed

Via register ProcessorStatus, the temperature of the processor on the Melexis EVK75027 as well as its current clock speed can be read out. Users can detect insufficient cooling of the processor this way.

|  |  |
| --- | --- |
|  | **Warning** |
| If the temperature goes above 80°C, the cooling is insufficient. The processor will automatically decrease its clock speed in this case (default is ca. 1 GHz). |
|  |  |

## Save Registers

The entire register map can be saved into the flash using the register CmdExec. It will be restored from flash after a reboot or power cycle. Use this feature to save a user specific configuration.

## Ethernet/IP Settings

### MAC Address

A dedicated Ethernet MAC address from BECOM Systems MAC address pool is assigned to each Melexis EVK75027 by default. This MAC address is saved in the OTP and cannot be changed by the user.

The user is allowed to assign the Melexis EVK75027 another MAC address using the registers Eth0Mac0 to Eth0Mac2. Be aware that in order to make the changes persistent you have to save the register map to flash using register CmdExec, otherwise the changes will be lost on a reboot or power cycle.

If the register map in the flash will be cleared the factory default MAC address from OTP will be loaded.

### IP/TCP/UDP Settings

The IP Settings of the Melexis EVK75027 can be changes via the Eth0\_\* registers. A change of the IP settings (IP address, subnet mask, default gateway) will take effect on writing the latter one. Port settings will take effect immediately. UDP destination IP addresses will take effect immediately. Please see the register description for details.

To make the changes persistent you have to save the register map by writing a dedicated value to the CmdExec register.

## Device Status and Error Conditions

The device offers several registers that indicate the current status and error conditions. Please consult Table 24 for a complete list of available flags.

Those flags are separated into two groups: Status flags and error flags.

Status flags indicate a specific status of the device. Status flags are set and cleared by the firmware depending on the state. They do not indicate errors.

Error flags indicate an error condition that is present currently or was present in the past. Error flags are only set by the firmware, they are never cleared automatically. The operator may clear all error flags by writing bit 6 of register ***Mode0***.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Flag name | Register | Bit No. | Error flag | Status flag | Description |
| Ongoing Calibration | Status | 2 |  | X | The device is performing an operator-initiated calibration procedure currently. |
| Illumination temperature error | Status | 3 | X |  | There was an error reading the temperature sensor on the illumination board. |
| ToF sensor temperature error | Status | 4 | X |  | There was an error reading the temperature sensor near the MLX75027 . |
| Calibration data missing | Status | 5 |  | X | Some calibration data is missing (was not uploaded or calibrated). Please consult registers ***CalibStatus*** and ***CalibStatus2*** for more information. |
| Factory Regmap was loaded | Status | 6 |  | X | The device does not have a user register map stored, and so the factory default settings are currently loaded. |
| Previous firmware version was restored | Status | 8 |  | X | The programmed firmware could not be booted correctly for 3 times, so the previous firmware version was restored and is now booted. The reason is either that the last firmware update failed or was incomplete, or the device was power-cycled too often without booting fully in between. |
| Illumination over-temperature | Status | 9 |  | X | The current temperature of the illumination board exceeds the value of register ***MaxLedTemp***. Illumination has been stopped automatically and will continue after cool-down. This is an indication that your cooling measures of the illumination board are insufficient! |
| Illumination error | Status | 11 | X |  | The Laser/LED switch on the illumation board reports error(s). |
| Sensor board temperature error | Status | 14 | X |  | There was an error reading the temperature sensor on the sensor board. |
| Capture timeout | Status2 | 2 |  | X | There are currently no frames received by the i.MX CSI. Errors during transmission; |
| Application processor too hot | Status2 | 3 |  | X | The current temperature of the i.MX6 application processor exceeds 80˚C. At this temperature, the processor is clocked down automatically (ARM cores as well as GPU). This is an indication that your cooling measures of the processor module are insufficient! |

Table 24: Status and Error Flags

## Reset to Factory Default

The Melexis EVK75027 can be reset to the factory default register settings by deleting the saved register map. This can be done by writing a dedicated value to the register ***CmdExec***.

## Firmware Update

The Melexis EVK75027 firmware is capable of updating the evaluation kit’s firmware (as well as the boot loader). The update procedure is executed using dedicated TCP/IP command frames over the control interface connection.

BECOM Systems provides a .NET based tool for updating the Melexis EVK75027 firmware over Ethernet. Please refer to our support site.

|  |  |
| --- | --- |
|  | **BltTofSuite Downloader** |
| ⮱ https://support.bluetechnix.at/index.html |

|  |  |
| --- | --- |
|  | **Note** |
| In order to complete a firmware update, a complete reboot of the evaluation kit is required. The evaluation kit will NOT reboot automatically. |

## Firmware Recovery

If a new firmware fails to load for 3 times, the Melexis EVK75027 boot loader will recover the old firmware automatically.

After a firmware recovery, Bit[8] of the Status register is set.

The Melexis EVK75027 evaluation kit also features a firmware load attempt counter, in register BootStatus. It is usually 1 (first boot attempt successful). It will lose its value if power is completely removed from the evaluation kit.

## Logging

The evaluation kit automatically saves log messages to a dedicated partition in the internal flash.

Log data may be retrieved using the Secure Shell login (see chapter 3.3) and can be found at /mnt/logs/messages\* files. Newest log data is contained in file messages.

## GPOs

The camera features 3 general-purpose outputs (GPO0, GPO1 and GPO2). Please see the register description (0x00D0) in chapter 6.1 for more information.

|  |  |  |  |
| --- | --- | --- | --- |
| Pin | Name | Pin | Name |
| 1 | ECSPI3\_MISO/DISP0\_DAT2/GPIO4\_23 – **GPO2** | **2** | Vin (12 V) |
| 3 | ECSPI3\_MOSI/DISP0\_DAT1/GPIO4\_22 | **4** | GND |
| 5 | ECSPI3\_SCLK/DISP0\_DAT0/GPIO4\_21 | **6** | GND |
| 7 | ECSPI3\_MOSI/DISP0\_DAT1/GPIO4\_22 | **8** | 3.3 V |
| 9 | ECSPI3\_MISO/DISP0\_DAT2/GPIO4\_23 | **10** | GND |
| 11 | I2C2.SDA | **12** | GND |
| 13 | I2C2.SCL | **14** | GPIO.2\_23 |
| 15 | GPIO.2\_24 - **GPO0** | **16** | GPIO.2\_25 - **GPO1** |

Table 4‑4: Interface Board, Multi-IO connector

# Software

## Demo Application

For the first evaluation of the kit and to evaluate different settings and configurations a .NET demo application for Microsoft Windows is provided: BLT-ToF-Suite. The demo application can be downloaded from our support web site.

|  |  |
| --- | --- |
|  | **Software and documentation** |
| ⮱ https://support.bluetechnix.at/index.html |

## Getting Started Software Development Example

To facilitate the integration of the Melexis EVK75027 into your own application, a C library implementing the “BltToFAPI” is provided. Please refer to our support site.

|  |  |
| --- | --- |
|  | **Software and documentation** |
| ⮱ https://support.bluetechnix.at/index.html |

# Register Description

|  |  |
| --- | --- |
|  | **Note** |
| Some critical registers are password protected. To enable the functionality a specific value must be written to the CmdEnablePasswd register in advance to enable the functionality. This should prevent from accidentally executing certain functions. |

## General

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Addr (hex) | Register Name | Default Value (hex) | R/W | Description |
| 0001 | Mode0 | 0001 | R/W | Bit[4]: 1…Manual Trigger (self-clearing bit)  Bit[6]: 1…Clear error bits in Status and Status2 registers (self-clearing bit) |
| 0003 | Status | 0060 | R | Bit[2]: 1…Ongoing Calibration  Bit[3]: 1…Illumination temperature sensor error  Bit[4]: 1…ToF sensor temperature sensor error  Bit[5]: 1…Calibration data missing  Bit[6]: 1…Factory Regmap was loaded  Bit[8]: 1…Previous firmware version was restored  Bit[9]: 1…Illumination over-temperature  Bit[11]: 1…Illumination error |
| 0004 | ImageDataFormat | 0000 | R/W | Bits[3:10]:  0…2 bytes distance data / 2 bytes amplitude data  3… X/Y/Z coordinates (2 bytes signed for each coordinate)  4… X/Y/Z coordinates and amplitude data (2 bytes signed for each coordinate, 2 bytes unsigned for the amplitude value)  9… distance data and X/Y/Z coordinates (2 bytes unsigned for the distance value, 2 bytes signed for each coordinate)  10… Optical axis coordinate and amplitude data (2 bytes signed for the coordinate, 2 bytes unsigned for amplitude data)  11…4 channels Test mode: Arithmetic functions (2 bytes ascending index; 2 bytes constant 0xbeef; 2 bytes  ascending squared index; 2 bytes constant 0x0000)  12…2 bytes distance data  13… 2 bytes raw distance data; 2 bytes amp data  23… 2 bytes distance data, 2 bytes amp data, 2 bytes balance data  24…Raw phase data; 1 to 8 channels (variable); 2 bytes per pixel  26… 2 bytes distance data, 1 byte amp data at specific settings  27…2 bytes amplitude data |
| 0005 | IntegrationTime | 00FA | R/W | Integration Time [µs] |
| 0006 | DeviceType | 31FF | R | Hardware specific identification |
| 0008 | FirmwareInfo |  | R | Bit[0-5]: Non Functional Revision  Bit[6-10]: Minor Revision  Bit[11-15]: Major Revision |
| 0009 | ModulationFrequency | 0FA0 | R/W | Modulation frequency in multiples of 10kHz |
| 000A | Framerate | 000F | R/W | ToF frame rate [Hz] |
| 000B | HardwareConfiguration | 006E | R/W | Lens opening angle identifier. |
| 000C | SerialNumberLowWord |  | R | Lower 16bit of the 32bit Serial Number |
| 000D | SerialNumberHighWord |  | R | Higher 16bit of the 32bit Serial Number |
| 000E | FrameCounter |  | R | Frame Counter (increments on every captured frame) |
| 000F | CalibrationCommand | 0000 | R/W | Bit[0:7]: Cmd code  13…FPPN calibration of the current modulation frequency  16…Clear FPPN calibration data for current modulation frequency  19…Calibrate DistOffset of the current modulation frequency |
| 0010 | ConfidenceThresLow |  | R/W | Amplitude threshold for valid distance data |
| 0011 | ConfidenceThresHigh |  | R/W | Amplitude threshold for valid distance data |
| 001B | LedboardTemp |  | R | Average temperature of illumination in 0,01[°C] (FFFF: Temperature not available). |
| 001C | MainboardTemp |  | R | Temperature of ToF chip in 0,01[°C] (FFFF: Temperature not available). |
| 0020 | RealWorldXcoordinate | 0000 | R/W | Distance to the calibration target [mm]. |
| 0021 | CalibStatus | 0000 | R | Bit[0-7]: Status/error  0…Idle  19…FPPN calibration  20…Erasing flash  21…DistOffset calibration  161…Operation done  255.. Generic error  252.. Out of memory  246.. Wrong image mode (Need distance)  244.. RealWorldXCoordinate value zero or too large  Bit[10]: 1…Error occurred  Bit[12]: 1…No FPPN Calibration data in NVM for current modulation frequency  Bit[14]: 1…No Lens Calibration data in NVM for current ***HardwareConfiguration*** setting |
| 0022 | CmdEnablePasswd | 0000 | R/W | Set a password for critical operations:  0x4877: Register map flash operations (register ***CmdExec*** 0x0033)  0x5E6B: Test commands (register ***TestConfig*** 0x01C0) |
| 0024 | MaxLedTemp | 1B58 | R/W | Maximum tolerable illumination temperature 0.01[°C] |
| 0026 | HorizontalFov |  | R | Horizontal field of view in 0,01[°].The content depends on the mounted lens and the calibration data and represents the real viewing angles. |
| 0027 | VerticalFov |  | R | Vertical field of view in 0,01[°].The content depends on the mounted lens and the calibration data and represents the real viewing angles. |
| 002B | TriggerDelay | 0000 | R/W | Delay between trigger assertion and image capturing [ms] |
| 002C | BootStatus | 4000 | R | Bit[14-15]: Firmware Load Counter. This counter is reset by the firmware. It counts the boot attempts. |
| 002D | TempCompGradientLim |  | R/W | Factor ‘c’ of the illumination temperature compensation function: y [mm] = a/100000 \* x³ + b/10000 \* x² + c/1000 \* x |
| 0030 | TempCompGradient2Lim |  | R/W | Factor ‘b’ of the illumination temperature compensation function: y [mm] = a/100000 \* x³ + b/10000 \* x² + c/1000 \* x |
| 0033 | CmdExec | 0000 | R/W | Initiate an operation:  0xC2AE…Clear RegMap in flash  0x9E20…Read RegMap from flash  0x909A…Read factory RegMap  0xDD9E…Save RegMap in flash  Writing these commands must be preceded by writing 0x4877 into register CmdEnablePasswd (0x0022). |
| 0034 | CmdExecResult | 0000 | R | Result code of the operation initiated using CmdExec  1…Success  Other…Error  *This register is cleared to 0x0 on read.* |
| 0035 | FactoryMacAddr2 |  | R | Hi byte and byte 4 of the MAC address stored in OTP flash |
| 0036 | FactoryMacAddr1 |  | R | Byte 3 and 2 of the MAC address stored in OTP flash |
| 0037 | FactoryMacAddr0 |  | R | Byte 1 and low byte of the MAC address stored in OTP flash |
| 0038 | FactoryYear |  | R | Production year (stored in OTP flash) |
| 0039 | FactoryMonthDay |  | R | Bit[0-7]: Production day (stored in OTP flash)  Bit[8-15]: Production month (stored in OTP flash) |
| 003A | FactoryHourMinute |  | R | Bit[0-7]: Production hour (stored in OTP flash)  Bit[8-15]: Production minute (stored in OTP flash) |
| 003B | FactoryTimezone |  | R | Production time zone (stored in OTP flash) |
| 003C | TempCompGradient3Lim |  | R/W | Factor ‘a’ of the illumination temperature compensation function: y [mm] = a/100000 \* x³ + b/10000 \* x² + c/1000 \* x |
| 003D | BuildYearMonth |  | R | Firmware Build date/time  Bit[14-4]: Year  Bit[3-0]: Month |
| 003E | BuildDayHour |  | R | Firmware Build day/hour  Bit[9-5]: Day  Bit[4-0]: Hour |
| 003F | BuildMinuteSecond |  | R | Firmware Build date/time  Bit[11-6]: Minute  Bit[5-0]: Second |
| 0040 | UpTimeLow |  | R | Lower 16 bit of uptime in [s] |
| 0041 | UpTimeHigh |  | R | Higher 16 bit of uptime in [s] |
| 0046 | ProcessorStatus |  | R | Bit[0:7]…Temperature of the processor in °C (0xFF: Sensor not available)  Bit[8:15]…Processor speed in 10-MHz-steps |
| 004A | TempCompGradientTim |  | R/W | Factor ‘c’ of the TIM temperature compensation function: y [mm] = a/100000 \* x³ + b/10000 \* x² + c/1000 \* x |
| 004B | TempCompGradient2Tim |  | R/W | Factor ‘b’ of the TIM temperature compensation function: y [mm] = a/100000 \* x³ + b/10000 \* x² + c/1000 \* x |
| 004C | TempCompGradient3Tim |  | R/W | Factor ‘a’ of the TIM temperature compensation function: y [mm] = a/100000 \* x³ + b/10000 \* x² + c/1000 \* x |
| 00C1 | DistOffset0 |  | R/W | Offset for distance values in millimeters |
| 00D0 | IOstate0 |  | R/W | Bit[9]: … state of GPO0 (R/W)  Bit[10]: … state of GPO1 (R/W)  Bit[11]: … state of GPO2 (R/W) |
| 00FD | Latency |  | R | Latency from capture to stream complete in milliseconds |
| 00FE | Status2 |  | R | Bit 1: 1… Capture error  Bit 2: 1… Capture timeout  Bit 3: 1… Application processor too hot, running at decreased GPU+CPU clock speed |

## User Defined

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Addr (hex) | Register Name | Default Value (hex) | R/W | Description |
| 0100 | UserDefined0 | 0 | R/W | For any purpose |
| 0101 | UserDefined1 | 0 | R/W | For any purpose |
| 0102 | UserDefined2 | 0 | R/W | For any purpose |
| 0103 | UserDefined3 | 0 | R/W | For any purpose |
| 0104 | UserDefined4 | 0 | R/W | For any purpose |
| 0105 | UserDefined5 | 0 | R/W | For any purpose |
| 0106 | UserDefined6 | 0 | R/W | For any purpose |
| 0107 | UserDefined7 | 0 | R/W | For any purpose |
| 0108 | UserDefined8 | 0 | R/W | For any purpose |
| 0109 | UserDefined9 | 0 | R/W | For any purpose |

## General (2)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Addr (hex) | Register Name | Default Value (hex) | R/W | Description |
| 010A | TempCompGradientBaseboard |  | R/W | Factor ‘c’ of the sensor board temperature compensation function: y [mm] = a/100000 \* x³ + b/10000 \* x² + c/1000 \* x + u |
| 010B | TempCompGradient2Baseboard |  | R/W | Factor ‘b’ of the sensor board temperature compensation function: y [mm] = a/100000 \* x³ + b/10000 \* x² + c/1000 \* x + u |
| 010C | TempCompGradient3Baseboard |  | R/W | Factor ‘a’ of the sensor board temperature compensation function: y [mm] = a/100000 \* x³ + b/10000 \* x² + c/1000 \* x + u |
| 010D | BaseboardTemp |  | R | Temperature of the sensor board in 0,01[°C] (FFFF: Sensor not available). |
| 0118 | CalibStatus2 |  | R | Bit[0]: … No wiggling calibration data in NVM |
| 0119 | BinnFlipMirror | 0000 | R/W | Bits[0..1]:  0…No binning  1…2x2 binning  2…4x4 binning  3…8x8 binning |
| 0159 | IllPower |  | R/W | Illumination Power in percent. 0…100 |
| 01C0 | TestConfig | 0000 | R/W | Bit[1]: 1… Watchdog Test  Writing this register must be preceded by writing 0x5E6B into register CmdEnablePasswd (0x0022) |

Table 25: General registers

## Device Update

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Addr (hex) | Register Name | Default Value (hex) | R/W | Description |
| 01D1 | FileUpdateStatus | 0000 | R | 0…idle  3…out\_of\_memory  6…file crc error  8…erasing flash  9…flashing  11…erasing failed  12…flashing failed  14…update success  16…header version conflict  18…wrong fw identifier  20…data inconsistent  21…in progress  255…protocol violation |

Table 26: Registers for device update

## Filter Configuration

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Addr (hex) | Register Name | Default Value (hex) | R/W | Description |
| 01E0 | ImgProcConfig | 28C0 | R/W | Bit[0]: 1…enable Median Filter  Bit[3]: 1…enable Bilateral Filter  Bit[4]: 1…enable Sliding Average  Bit[6]: 1…enable Wiggling compensation  Bit[7]: 1…enable FPPN compensation  Bit[10]: 1…enable FrameAverage Filter  Bit[11]: 1…enable Temperature compensation  Bit[13]: 1…enable offset via register DistOffset0 |
| 01E1 | FilterMedianConfig | 0001 | R/W | Bit[0-7]: Nr. of Median Iterations |
| 01E4 | FilterBilateralConfig | 13DE | R/W | Bit[0-5]: Sigma R (Width of range kernel)  Bit[6-11]: Sigma S (Width of spatial kernel)  Bit[12-15]: Nr. of iterations |
| 01E5 | FilterSlafConfig | 0005 | R/W | Bit[0-7]: Window size |
| 01E6 | FilterBilateralConfig2 | 0003 | R/W | Bit[0-5]: Square size (=> Window size = square size x square size) |
| 01E7 | FilterFrameAverageConfig | 0002 | R/W | Bit[0-3]: Number of Frames |

Table 27: Register for filter configuration

## Ethernet configuration

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Addr (hex) | Register Name | Default Value (hex) | R/W | Description |
| 0240 | Eth0Config | 0006 | R/W | Bit[0]: 1.. Enable DHCP  Bit[1]: 1.. Enable UDP streaming  Bit[2]: 1.. Ignore CRC for UDP streaming |
| 0241 | Eth0Mac2 |  | R/W | Byte 5 (=High byte) and byte 4 of MAC address  Writing this register has no immediate effect. |
| 0242 | Eth0Mac1 |  | R/W | Byte 3 and byte 2 of MAC address  Writing this register has no immediate effect. |
| 0243 | Eth0Mac0 |  | R/W | Byte 1 and byte 0 (=Low byte) of MAC address  **Writing this register will update the network configuration with the new MAC address.** |
| 0244 | Eth0Ip0 | 000A | R/W | Low word of IP address  Writing this register has no immediate effect (see register 0x0249). |
| 0245 | Eth0Ip1 | C0A8 | R/W | High word of IP address  Writing this register has no immediate effect (see register 0x0249). |
| 0246 | Eth0Snm0 | FF00 | R/W | Low word of subnet mask  Writing this register has no immediate effect (see register 0x0249). |
| 0247 | Eth0Snm1 | FFFF | R/W | High word of subnet mask  Writing this register has no immediate effect (see register 0x0249). |
| 0248 | Eth0Gateway0 | 0001 | R/W | Low word of gateway  Writing this register has no immediate effect (see register 0x0249). |
| 0249 | Eth0Gateway1 | C0A8 | R/W | High word of gateway  **Writing this register will update the network configuration with new IP address, subnet mask and gateway.** |
| 024B | Eth0TcpCtrlPort | 2711 | R/W | Port for TCP control interface |
| 024C | Eth0UdpStreamIp0 | 0001 | R/W | Low word of IP address for UDP stream  Writing this register has no immediate effect. |
| 024D | Eth0UdpStreamIp1 | E000 | R/W | High word of IP address for UDP stream  **Writing this register will update the network configuration with the new UDP stream address.** |
| 024E | Eth0UdpStreamPort | 2712 | R/W | Port for UDP streaming |
| 0259 | Eth0UdpPacketSize | 0578 | R/W | Packet size for UDP data interface |

Table 28: Registers for Ethernet configuration

## General 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Addr (hex) | Register Name | Default Value (hex) | R/W | Description |
| 0570 | ArticleNrPart1 |  | R | First part of the article number (###-\*\*\*\*-\*) |
| 0571 | ArticleNrPart2 |  | R | Second part of the article number **(\*\*\*-####-\*)** |
| 0572 | DeviceRevisionMajor |  | R | Third part of the article number (\*\*\*-\*\*\*\*-#) Also: Major part of the revision number |
| 0573 | DeviceRevisionMinor |  | R | Bit[0-7]: … nonfunctional part of the revision number  Bit[8-15]: … minor number of the revision number |
| 0574 | NofPhases | 0003 | R/W | Number of phases to be captured |
| 0575 | AtanLUTwidth | 000A | R/W | Atan LUT width in bit |

# Support

## General Support

General support for products can be found at BECOM Systems’ support site

|  |  |
| --- | --- |
|  | **Support Link** |
| ⮱ https://support.bluetechnix.at/index.html |

## Software Packages

Software packages and software downloads are for registered customers only

|  |  |
| --- | --- |
|  | **Software Package** |
| ⮱ https://support.bluetechnix.at/index.html |

# Firmware History

## Version Information

|  |  |  |  |
| --- | --- | --- | --- |
| Firmware Version | Status | Release date | Changes |
| 0.11.0 | X-Grade | Feb 2019 |  |

Table 29: Overview Melexis EVK75027 firmware changes

## Anomalies

|  |  |  |
| --- | --- | --- |
| Applies to | Date | Description |
|  |  |  |

Table 30: Overview Melexis EVK75027 firmware anomalies

# Document Revision History

|  |  |  |
| --- | --- | --- |
| Version | Date | Document Revision |
| 1 | 2019 03 13 | Initial version of the document |

Table 31: Document revision history

1. List of Figures and Tables

**Figures**

[Figure 3‑1: UDP streaming data format 19](#_Toc4171103)

[Figure 4‑1: Image processing flow 24](#_Toc4171104)

[Figure 4‑2: Data stream example of Distance and Amplitude data for 320x240 pixel 27](#_Toc4171105)

[Figure 4‑3: Data stream of 160x120 pixel XYZ Point Cloud 28](#_Toc4171106)

[Table 4‑4: Interface Board, Multi-IO connector 35](#_Toc4171107)

**Tables**

[Table 1: Supported command frames 9](#_Toc4171108)

[Table 2: Register read command frame 10](#_Toc4171109)

[Table 3: Register read response frame 11](#_Toc4171110)

[Table 4: Register read flag description 11](#_Toc4171111)

[Table 5: Result codes 12](#_Toc4171112)

[Table 6: Register write command frame 12](#_Toc4171113)

[Table 7: Register write response frame 13](#_Toc4171114)

[Table 8: Register write flag description 13](#_Toc4171115)

[Table 9: Reset command frame 14](#_Toc4171116)

[Table 10: Reset response frame 14](#_Toc4171117)

[Table 11: Reset flag description 14](#_Toc4171118)

[Table 12: Flash update command frame 15](#_Toc4171119)

[Table 13: Flash update response frame 16](#_Toc4171120)

[Table 14: Flash update subcommand description 16](#_Toc4171121)

[Table 15: Flash update flag description 16](#_Toc4171122)

[Table 16: Alive command frame 17](#_Toc4171123)

[Table 17: Alive response frame 17](#_Toc4171124)

[Table 18: Alive flag description 17](#_Toc4171125)

[Table 19: UDP packet header 20](#_Toc4171126)

[Table 20: UDP packet header flag description 20](#_Toc4171127)

[Table 21: Image data header 22](#_Toc4171128)

[Table 22: Default login credentials 22](#_Toc4171129)

[Table 23: Debug UART settings 23](#_Toc4171130)

[Table 24: Status and Error Flags 34](#_Toc4171131)

[Table 25: General registers 42](#_Toc4171132)

[Table 26: Registers for device update 43](#_Toc4171133)

[Table 27: Register for filter configuration 43](#_Toc4171134)

[Table 28: Registers for Ethernet configuration 44](#_Toc4171135)

[Table 29: Overview Melexis EVK75027 firmware changes 47](#_Toc4171136)

[Table 30: Overview Melexis EVK75027 firmware anomalies 47](#_Toc4171137)

[Table 31: Document revision history 48](#_Toc4171138)