BLUETECHNIX
Embedding Ideas

Sentis-ToF-P510
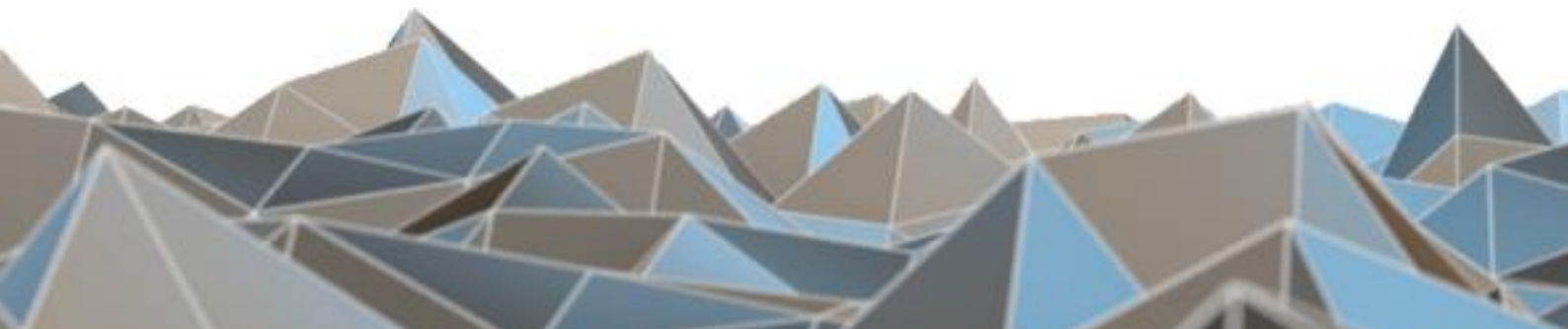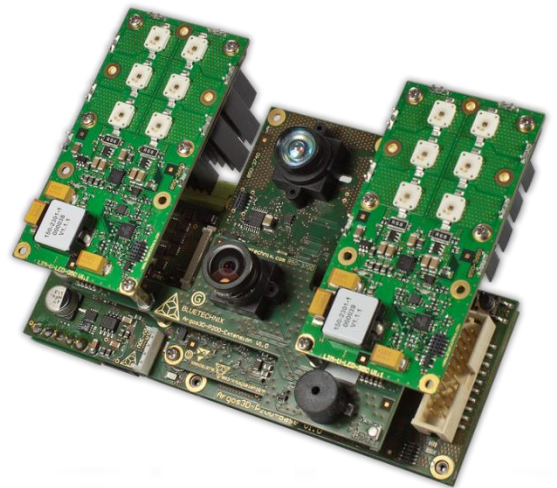
Software User Manual

Version 6

Bluetechnix

Waidhausenstraße 3/19
A-1140 Vienna
AUSTRIA

office@bluetechnix.com
www.bluetechnix.com

Sentis-ToF-P510 – Software User Manual

Document No.: 900-308 / A

Publication date: September 14, 2015

Subject to change without notice. Errors excepted.

Table of Contents

Information

For further information on technology, delivery terms and conditions and prices please contact Bluetechnix (http://www.bluetechnix.com).

Warning

Due to technical requirements components may contain dangerous substances.

1 General Information

This guide applies to the Sentis-ToF-P510 camera platform from Bluetechnix. 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.

1.1 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.

2 Overview

The document describes the necessary steps and settings to work with the Sentis-ToF-P510 and describes the firmware dependent interfaces.

This document applies to firmware version 0.7.0.

For a hardware compatibility list please refer to our support site.

Software and documentation

https://support.bluetechnix.at/index.html

3   Interfacing

The Sentis-ToF-P510 provides control and data interfaces via Gigabit-Ethernet.

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

The data interface provides a continuous stream of ToF (and color) data depending on the configuration.

> **Note**
>
> Bluetechnix provides an abstraction of control and data interfaces by means of the *BltTofApi*. If you use this API, you need not be familiar with control and data interface in detail. Refer to chapter 7.1 for the *BltTofApi*.

3.1   Control Interface

The Sentis-ToF-P510 can be configured using a TCP/IP connection. For the control interface the camera 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 camera can be configured using a dedicated set of command frames. The camera 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 Sentis-ToF-P510
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 camera closes the control interface connection and waits for a new incoming connection request. Up to 5 concurrent control connections are supported.

Table 3-1: Supported command frames

The following section describes each command frame and the expected answer in detail. To be able to communicate with the Sentis-ToF-P510 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.

3.1.1 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 3-2: Register read command frame

Note 1): For the CRC16 calculation the CRC-CCITT is used (Polynom: 0x1021, start value: 0). Please ask the Bluetechnix 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

Addr	Field	Type	Value	Description
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 3-3: Register read response frame

Note 1): For the CRC16 calculation the CRC-CCITT is used (Polynom: 0x1021, start value: 0). Please ask the Bluetechnix 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 Bluetechnix support for an implementation example of the CRC-32.

Flags

Flags	Description
Bit 0	1: Ignore DataCrc32

Table 3-4: Register read flag description

3.1.2 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 3-5: Result codes

3.1.3 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 3-6: Register write command frame

Note 1): For the CRC16 calculation the CRC-CCITT is used (Polynom: 0x1021, start value: 0). Please ask the Bluetechnix 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 Bluetechnix 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

Addr	Field	Type	Value	Description
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 3-7: Register write response frame

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

Flags

Flags	Description
Bit 0	1: Ignore DataCrc32

Table 3-8: Register write flag description

Result codes

Please refer to Table 3-5.

3.1.4 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 3-9: Reset command frame

Note 1): For the CRC16 calculation the CRC-CCITT is used (Polynom: 0x1021, start value: 0). Please ask the Bluetechnix 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 3-10: Reset response frame

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

Flags

Flags	Description
	Currently no flags defined for this command

Table 3-11: Reset flag description

Result codes

Please refer to Table 3-5.

3.1.5 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 or 0x0C	0x0B: Flash boot loader 0x0C: Flash Application 0x15: Flash lens calibration data for XYZ calculation 0x29: Flash intrinsic ToF sensor calibration file 0x2A: Flash intrinsic color sensor calibration file 0x2B: Flash extrinsic calibration file
0x04	SubCommand	Uint8	Refer to table	Indicates which flash to write to

Addr	Field	Type	Value	Description
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 3-12: Flash update command frame

Note 1): For the CRC16 calculation the CRC-CCITT is used (Polynom: 0x1021, start value: 0). Please ask the Bluetechnix 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 Bluetechnix 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	0x0B or 0x0C	0x0B: Flash boot loader 0x0C: Flash Application
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 3-13: Flash update response frame

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

Subcommand

SubCommand	Description
Ignored	For boot loader and application update

Table 3-14: Flash update subcommand description

Flags

Flags	Description
Bit 0	1: Ignore DataCrc32

Table 3-15: Flash update flag description

Result codes

Please refer to Table 3-5.

3.1.6 Read data from Flash

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	→	0x8D: Read intrinsic ToF sensor calibration file 0x8E: Read intrinsic color sensor calibration file 0x8F: Read extrinsic calibration file
0x04	SubCommand	Uint8		Ignored
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)		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 read as a binary byte stream

Table 3-16: Flash update command frame

Note 1): For the CRC16 calculation the CRC-CCITT is used (Polynom: 0x1021, start value: 0). Please ask the Bluetechnix 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 Bluetechnix 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		Same as in send command
0x04	SubCommand	Uint8		Same as in send command
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 3-17: Flash update response frame

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

Subcommand

SubCommand	Description
Ignored	

Table 3-18: Flash update subcommand description

Flags

Flags	Description
Bit 0	1: Ignore DataCrc32

Table 3-19: Flash update flag description

Result codes

Please refer to Table 3-5.

3.1.7 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

Addr	Field	Type	Value	Description
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 3-20: Alive command frame

Note 1): For the CRC16 calculation the CRC-CCITT is used (Polynom: 0x1021, start value: 0). Please ask the Bluetechnix 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 3-21: Alive response frame

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

Flags

Flags	Description
	Currently no flags defined for this command

Table 3-22: Alive flag description

Result codes:

Please refer to Table 3-5.

3.2   Data Interface

A UDP stream delivers depth and amplitude data from the Sentis-ToF-P510. Each UDP packet contains a header and up to 1400 bytes of data (Ethernet, IP, and UDP headers are not shown in Figure 3-1).
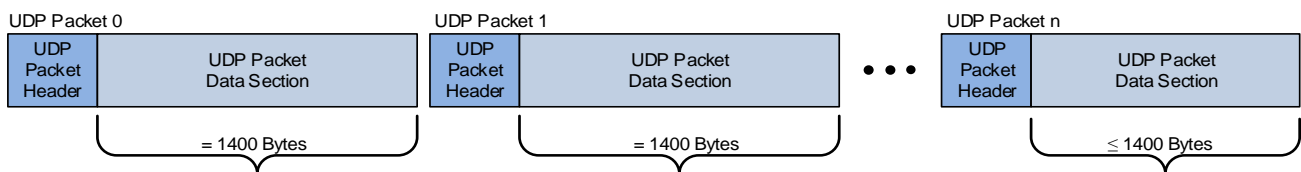
Figure 3-1: UDP streaming data format

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 camera 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 his local network interface card (NIC) where the camera 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.

3.2.1 UDP Streaming Header

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 UDP Streaming 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 3-24	Optional flags
0x14	Reserved			Reserved for future use
0x20	ImageData			Image data section

Table 3-23: UDP packet header

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

Flags

Flags	Description
Bit 0	1: Ignore DataCrc32

Table 3-24: UDP packet header flag description

3.2.2 Frame Header

The current header version is **3**.

The frame data itself is split into a 64 byte frame header and the frame data section. The format of the frame data depends on the selected image format and is described in chapter 4.3. Below you can find the format of the 64 byte frame 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)	0x00A0	Width of the image in pixels.
0x06	ImageHeight	Uint16 (high byte first)	0x0078	Height of the image in pixels.
0x08	NofChannels	Uint8		Nof data channels. Depends on the image format
0x09	BytesPerPixel	Uint8	0x02	Bytes per pixel of the ToF 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.
0x12	Reserved			
0x1A	MainTemp	Uint8		TIM (**T**oF **I**mage sensor **M**odule) temperature in °C + 50. Decrement this field by 50 to get the current TIM temperature. A value of 0xFF means sensor error.
0x1B	LEDtemp	Uint8		LIM (**Li**ght **M**odule) Temperature in °C + 50. Decrement this field by 50 to get the current LIM temperature. A value of 0xFF means sensor error.
0x1C	FirmwareVersion	Uint16 (high byte first)		Content of the register *FirmwareInfo*
0x1E	Magic	Uint16 (high byte first)	0x3331 or 0xCC32	0x3331: Magic bytes indicating header version 3.1. Default. 0xCC32: Magic bytes indicating header version 3.2. Used only in combination with JPEG encoded color channel.
0x20	IntegrationTime	Uint16 (high byte first)		ToF Integration time in us.
0x22	ModFreq	Uint16 (high byte first)		ToF Modulation frequency with resolution 10 kHz (e.g., a value of 0x1234 means frequency 46.6 MHz)
0x24	Temp3	Uint8		Base Board Temperature sensor value in °C + 50. Decrement this field by 50 to get the current temperature. A value of 0xFF means sensor error.
0x25	ColorMode	Uint8		Pixel format of color sensor frame: 0…No color data in frame 1…RGB565 2…JPEG *Is always zero, unless ImageFormat is 2 or 6.*

Addr	Field	Type	Value	Description
0x26	ColorSensorWidth	Uint16 (high byte first)		Width in pixels of color sensor frame. *Is zero, unless ImageFormat is 2 or 6.* *Can be zero with ImageFormat 2 or 6 (=no sensor data available).*
0x28	ColorSensorHeight	Uint16 (high byte first)		Height in pixels of color sensor frame. *Is zero, unless ImageFormat is 2 or 6.* *Can be zero with ImageFormat 2 or 6 (=no sensor data available)*
0x2A	SequenceNumber	Uint8		ToF sequence number of the data frame
0x2B	Reserved			
0x2C	ColorChannelLength	Uint32 (high byte first)		Length of color channel data in bytes. Variable for JPEG color mode. Fixed for RGB565 color mode (= ColorSensorWidth * ColorSensorHeight * 2)
0x30	Reserved			
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 3-25: Frame header

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

3.3   Color Sensor RTP Stream

A UDP-based RTP stream with video data from the CMOS color image sensor is available in different resolutions and frame rates. The video stream is either H.264 (a.k.a. MPEG-4 Part 10 or AVC - Advanced Video Coding) Baseline profile encoded, or raw video (YUV420 format).

For configuration of the stream, please refer to registers **ColorStreamParams** and **Eth0UdpRtpStream***.

> **!**
>
> **Note**
> The RTP stream is completely asynchronous and independent of the data interface.
>
> If the data interface is configured to image data formats that deliver color sensor data, the
>
> RTP stream cannot be enabled.

The color stream is disabled by factory default. Other default settings are:

- **IP-Address**: Multicast address 224.0.0.1

- **UDP port**: 10004

- **Resolution**: QCIF

- **Frame rate**: 15 fps

- **Codec**: AVC (H.264)

Note

! On any firmware update, the RTP stream is stopped automatically.

3.3.1 Web server

The configuration of the RTP color sensor stream receiver is accomplished via the Session Description Protocol (SDP). The camera runs a tiny web server which is available at the camera's IP address and standard TCP port 80. Navigating to this IP address with a web browser will automatically trigger the download of the .sdp file.

Note

! The .sdp file is re-generated on each change of the RTP color sensor stream's configuration.

3.4 Manual frame triggers

The default mode of the Sentis-ToF-P510 is video mode, where the camera streams continuously with configured frame rate. To use manual frame triggering, the video mode must be disabled in register *Mode0*.

A frame can be triggered by either

- Hardware trigger or
- Software trigger: See register *Mode0*.

Both will trigger the capturing of as many sequences on the ToF sensor, as is configured in register *NofSequ*, as well as a transition to low on the trigger output.

If color data is transferred via the data interface, the color sensor operates at either 15 or 30 frames per second in the background (depending on register settings in *ColorStreamParams*). A color image is assigned to each captured ToF image by means of both internal capture timestamps. If no color image is available (if the ToF sensor frame rate is higher than the color sensor frame rate), an empty color image is transferred.

3.5 External Illumination Interface

An external illumination unit (ToF-Flash) is supported. It features

- The differential modulation signal according to the selected modulation frequency
- 1-Wire communication line between the external illumination module (ToF-Flash) and the camera
- Supported protocol: <u>OWIRE V1</u> (OWIRE V3 is not supported).

Note

! If the Sentis-ToF-P510 firmware detects an external illumination unit via the 1-Wire line, the over-temperature-protection monitors the external units. Internal illumination is turned off in this case (<u>LIMs >= V2.0 are required!</u>)

Please consult the Hardware User Manual and the Bluetechnix Support if you want to you use a ToF-Flash with the Sentis-ToF-P510.

Please refer to the Hardware User Manual for detailed information on the interface.

3.6   GPIOs

The camera features one general-purpose input/output. Please see the register description in chapter 5.2 for more information.

There is also a buzzer available for the camera being able to send an acoustic signal.

3.7   PoE power control

If the camera is powered via a PoE connection (and not via VAUX), the firmware is able to detect the PoE power class it is currently connected to.

It will automatically adjust its current consumption to the maximum allowed power.

The following classes can be detected (see register *PoEStatus*):

Power class	Max. allowed power	Action
PoE	13 W	Power limitation to 13 W [1]
PoE+	25.5 W	Power limitation to 25.5 W
PoE++	not specified	No power limitation
VAUX supply	not specified	No power limitation

Table 3-26: Supported power supplies

Note 1. PoE mode does not allow for any illumination, so there is no reasonable usage of the camera with this power supply.

The power is limited by limiting the max. allowed FITP (Frame rate-Integration Time-Product), i.e., by limiting either the frame rate or the integration time (or both). Please see chapter 0 for details.

The detected PoE class can be overridden by means of register **PoEOverride**.

3.8   Secure Shell (SSH) Login

The Sentis-ToF-P510 camera features an OpenSSH server listening to TCP port 22.

	Root account	User account
Username	root	user
Default password	root	user

Table 3-27: Default login credentials

3.8.1   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 camera reboot: `cp /etc/shadow /mnt/settings`

3.9 Debug UART

Note

! Using the Debug UART is optional.

The Sentis-ToF-P510 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 micro-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 3-28: Debug UART settings

The Debug UART also allows to log in to the camera's Linux OS. Please see chapter 3.8 for the default login and how to change it.

4 Camera Features

4.1 Basic Settings

The Sentis-ToF-P510 comes up according to the reset (default) values as described in the register description chapter (refer to chapter 6).

Each camera has been pre-configured with a factory-default register map.

4.2 ToF Image Processing Chain

The following flow diagram shows the image processing chain of the Sentis-ToF-P510 for the ToF sensor data. Filters can be applied individually to distance and amplitude data. XYZ point cloud data is calculated from distance data on demand.
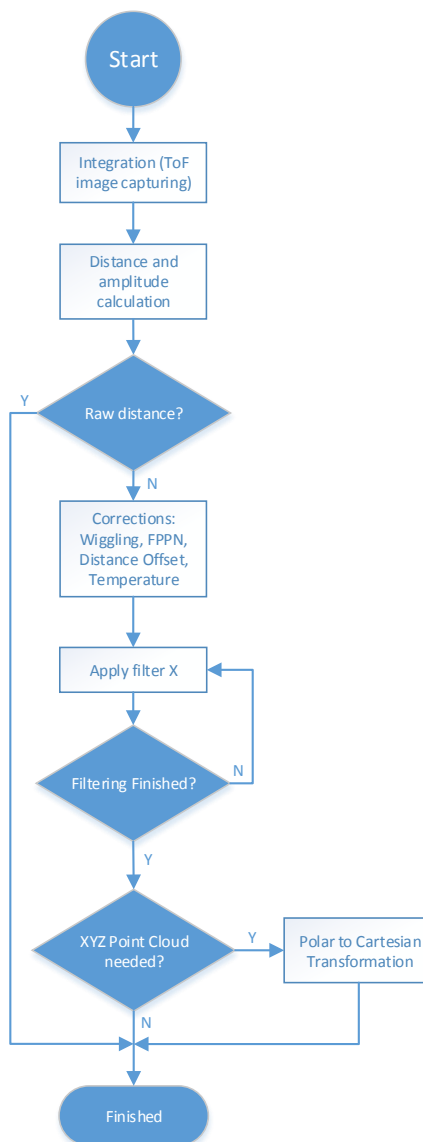
Figure 4-1: Image processing flow

4.3 ToF Image Filters

After the distance and amplitude calculation, filters can be applied to the distance and amplitude data. Each of the filters 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** (distance data) and **ImgProcConfig2** (amplitude data) 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

Note that the Frame Average filter for amplitude data can only be enabled if the Frame Average filter is enabled for distance data.

4.3.1 Median Filter

A 3x3 median filter can be applied.

Register: **FilterMedianConfig**

The number of iterations is configurable.

4.3.2 Bilateral filter

Registers: **FilterBilateralConfig, FilterBilateralConfig2**

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

4.3.3 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.

4.3.4 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 camera 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.

4.4   ToF Image Enhancements

4.4.1   Pixel invalidation

The Sentis-ToF-P510 provides an on-board check for invalid pixels:

- Underexposed pixels: The amplitude is too low for the distance value to be trustworthy. The Sentis-ToF-P510 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 Sentis-ToF-P510 sets the pixel distance to the minimum value. The threshold is set via register *ConfidenceThresHigh*.

- Invalid pixels: The Sentis-ToF-P510 features an additional amplitude check called ACF (auto correlation function) Plausibility Check. It detects inconsistent pixels e.g. in case of fast movement in the scene.

4.4.1.1   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.

For inconsistent pixels, the distance value is set to 0x0001.

4.4.1.2   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.

If the ACF plausibility check classified the pixel's distance as invalid, the X value of the appropriate pixel is set to 1. Y and Z values are set to 0.

4.4.2   Temperature compensation

The camera firmware continuously monitors temperatures of LIMs, TIM, and base board, and corrects the measured distance with cubic polynomials.

4.5 Camera Coordinate System

The default coordinate system starts pixel numbering in the upper left corner of the pixel array, seen from the camera's point of view. Also note the directions of X, Y, and Z coordinates (In XYZ image modes).
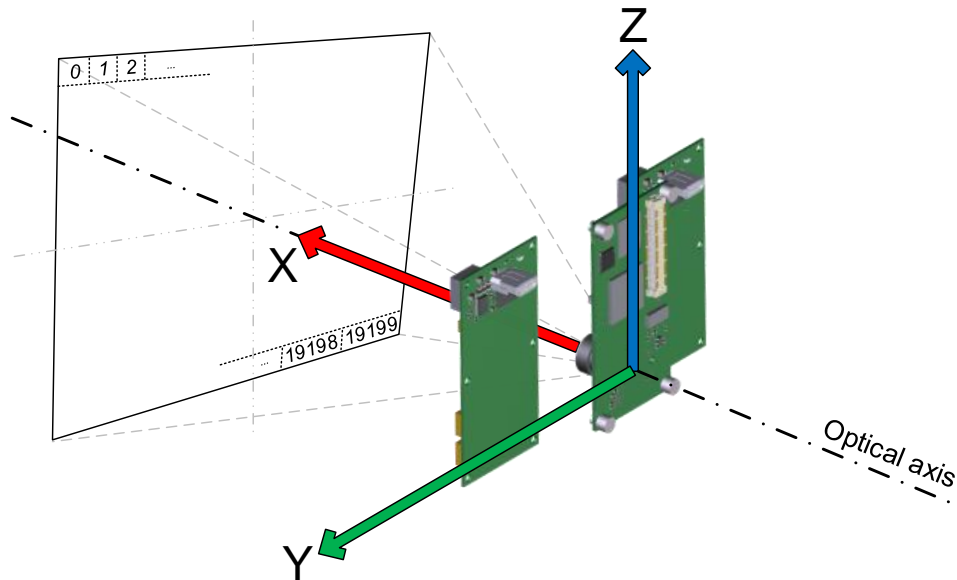

Figure 4-2: Sentis-ToF-P510 Default Coordinate System

Note

If you are working with a *BltTofApi* library, or the *BltTofSuite*, the camera coordinate system is automatically converted. Please refer to *BltTofApi* documentation for the coordinate system. (See chapter 5).

4.6 Camera Data Format

The camera provides up to four data channels via its data interface. The meaning of each data channel depends on the selected data format. The factory default setting provides an array of distance data and an array of amplitude data.

Alternatively, a 3D XYZ point cloud can be provided. Refer to chapter 4.4 for a description of the coordinate system of the camera.

Color data from the CMOS image sensor can also be selected.

A channel can carry one of the following data types:

- Distance data from the ToF sensor, in millimeters, as 16-bit unsigned (Uint16) values. Resolution is always 160x120 pixels.

- Amplitude data from the ToF sensor, as 16-bit unsigned (Uint16) values. Resolution is always 160x120 pixels.

- X coordinate values, in millimeters, as 16-bit signed (Int16) values. No negative values allowed. Resolution is always 160x120.

- Y coordinate values, in millimeters, as 16-bit signed (Int16) values. Resolution is always 160x120.

- Z coordinate values: Same as Y.

- Color sensor data in RGB565 format (16 bits per pixel). The resolution is set via register **ColorStreamParams**. The format is depicted in Table 4-1. Note that from one color value, B0, G0, and R0 are the least significant bits. B4, G5, and R4 are the most significant bits.

- Color sensor data in JPEG format. The resolution is set via register **ColorStreamParams**, as is the selection of JPEG instead of RGB565 format.

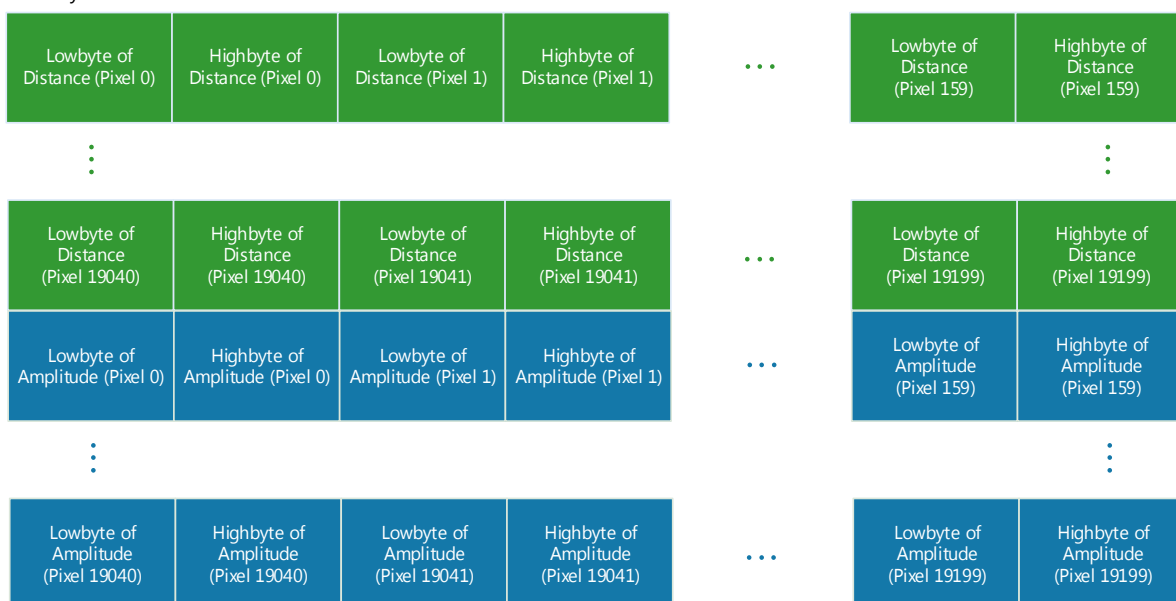| High byte								Low byte							
R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0

Table 4-1: RGB565 format

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.

4.6.1 Distances and Amplitudes

In this mode the distances and amplitudes will be transferred in progressive mode, first the distance array (channel 0), then the amplitude array (channel 1). The stream starts always with pixel #0.

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

First Byte in Stream

Lowbyte of Distance (Pixel 0)	Highbyte of Distance (Pixel 0)	Lowbyte of Distance (Pixel 1)	Highbyte of Distance (Pixel 1)	...	Lowbyte of Distance (Pixel 159)	Highbyte of Distance (Pixel 159)

⋮

Lowbyte of Distance (Pixel 19040)	Highbyte of Distance (Pixel 19040)	Lowbyte of Distance (Pixel 19041)	Highbyte of Distance (Pixel 19041)	...	Lowbyte of Distance (Pixel 19199)	Highbyte of Distance (Pixel 19199)
Lowbyte of Amplitude (Pixel 0)	Highbyte of Amplitude (Pixel 0)	Lowbyte of Amplitude (Pixel 1)	Highbyte of Amplitude (Pixel 1)	...	Lowbyte of Amplitude (Pixel 159)	Highbyte of Amplitude (Pixel 159)

⋮

| Lowbyte of Amplitude (Pixel 19040) | Highbyte of Amplitude (Pixel 19040) | Lowbyte of Amplitude (Pixel 19041) | Highbyte of Amplitude (Pixel 19041) | ... | Lowbyte of Amplitude (Pixel 19199) | Highbyte of Amplitude (Pixel 19199) |

Last Byte in Stream

Figure 4-3: Data stream of Distance and Amplitude data

4.6.2 Distances, Amplitudes and RGB565 Color

In this mode distances and amplitudes from the ToF sensor, as well as the color sensor image will be transferred in progressive mode, first the distance array (channel 0), then the amplitude array (channel 1), then the color sensor data (channel 2). The stream starts always with pixel #0.

The **distances** are coded in **millimeters** as **Uint16**, the **amplitudes** also as **Uint16**. The **color** sensor pixels are coded in **RGB565** (raw) or **JPEG** compressed format.

The resolution, format, and frame rate of the color sensor are set via register **ColorStreamParams**.

Channel 2 (which carries the color data) may have zero length, if no color data is available for the current frame. In this case, fields *ColorSensorWidth*, *ColorSensorHeight*, and *ColorSensorLength* of the frame header are set to zero (refer to section 3.2.2).

In the case the color data is transferred in JPEG format, frame header field *ColorSensorLength* contains the length of the JPEG-compressed image in bytes.

Figure 4-4: Data stream of Distance, Amplitude, RGB565 Color data

4.6.1 Distances and Color

In this mode distances from the ToF sensor and the color sensor image will be transferred in progressive mode, first the distance array (channel 0), then the color sensor data (channel 1). The stream starts always with pixel #0.

The **distances** are coded in **millimeters** as **Uint16**, the **color** sensor pixels are coded in **RGB565** (raw) or **JPEG** compressed format.

The resolution, format, and frame rate of the color sensor are set via register ***ColorStreamParams***.

Channel 1 (which carries the color data) may have zero length, if no color data is available for the current frame. In this case, fields *ColorSensorWidth*, *ColorSensorHeight*, and *ColorSensorLength* of the frame header are set to zero (refer to section 3.2.2).

In the case the color data is transferred in JPEG format, frame header field *ColorSensorLength* contains the length of the JPEG-compressed image in bytes.
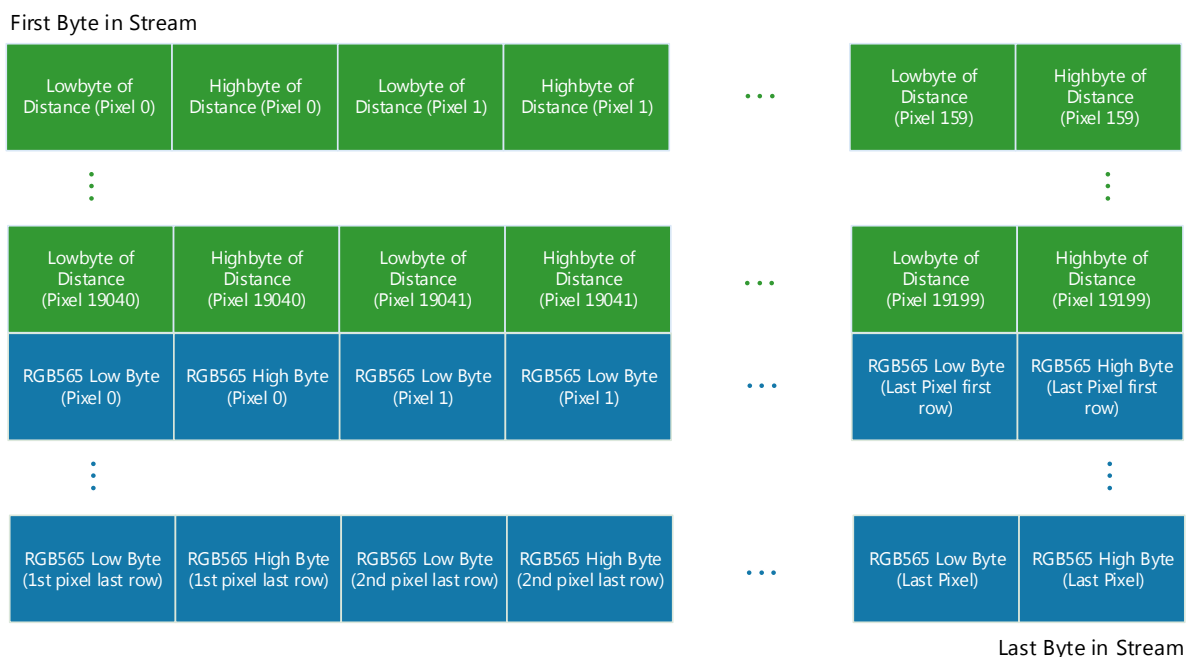

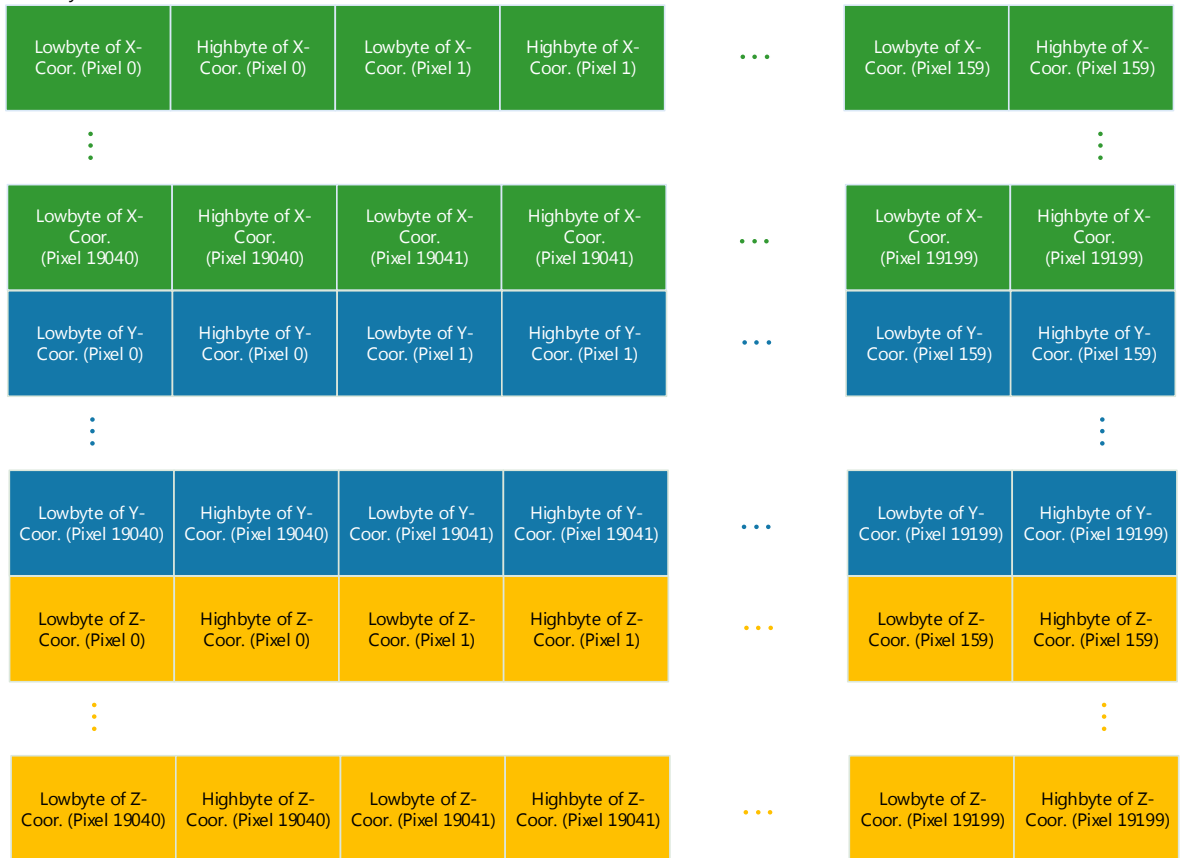Figure 4-5: Data stream of distance and RGB565 data

4.6.2 XYZ Point Cloud

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 stream starts always with pixel #0.

The **coordinates** are coded in **millimeters** as **Int16.**
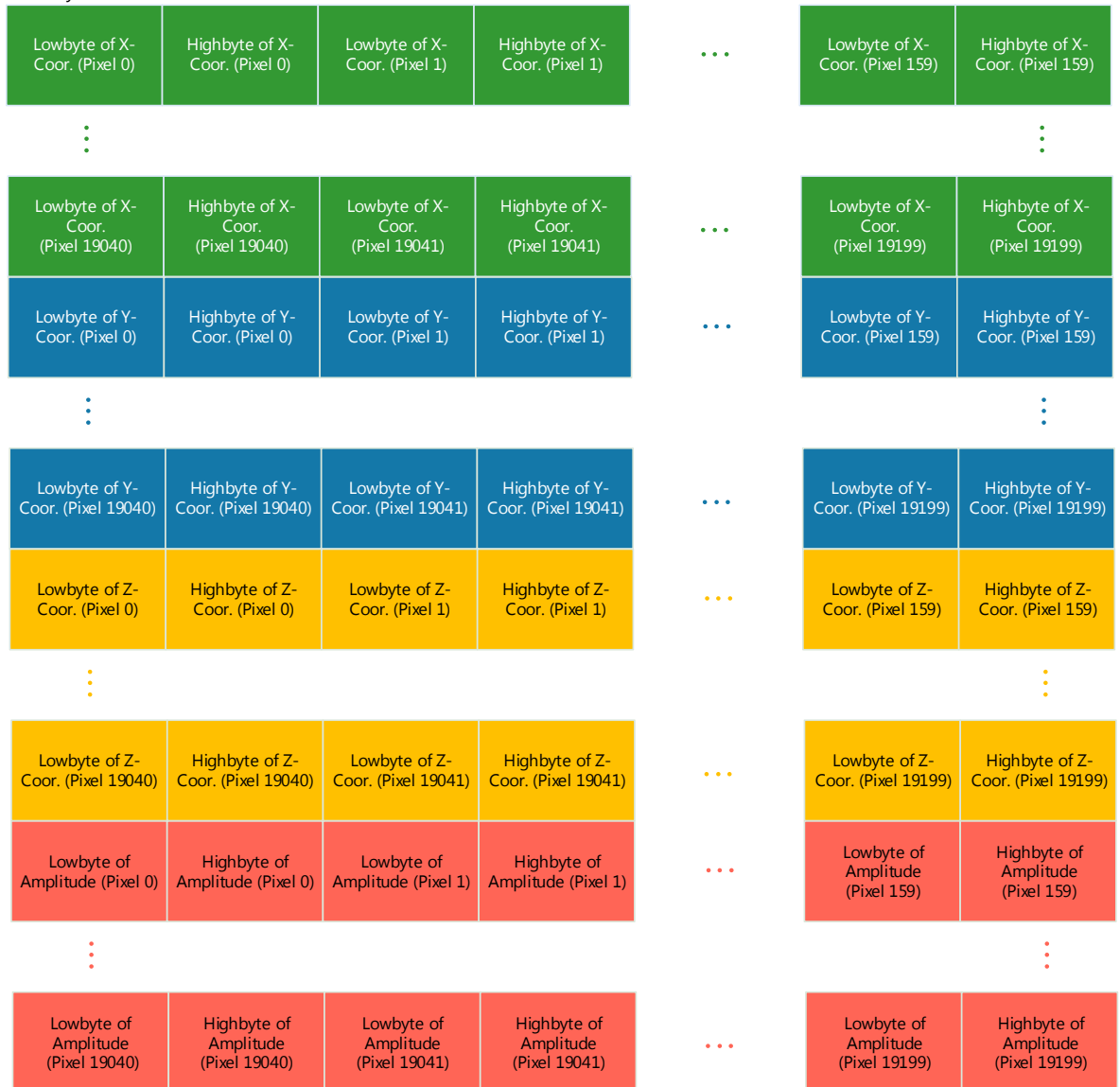
First Byte in Stream

Figure 4-6: Data stream of XYZ Point Cloud

4.6.3 XYZ Point Cloud and Amplitudes

In this mode the XYZ point cloud and the amplitude will be transferred in progressive mode, first the X coordinate array (channel 0), then the Y (channel 1) and Z (channel 2) coordinate array, then the amplitude array (channel 3). The stream starts always with pixel #0.

The **coordinates** are coded in **millimeters** as **Int16** the **amplitudes** as **Uint16.**

First Byte in Stream

Figure 4-7: Data stream of XYZ Point Cloud and Amplitude

4.6.4 Distances and XYZ Point Cloud

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 stream starts always with pixel #0.

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

4.6.5 X Coordinate and Amplitudes

In this mode a single coordinate array, more specifically, the one belonging to the optical axis of the camera (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**.

4.6.6   Distances

In this mode a single array with distances is transferred in channel 0. The stream starts always with pixel #0.

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

4.6.7   Test mode

In this mode four arrays with test data are transferred in progressive order.

- Channel 0:   Uint16 value = Pixel Index

- Channel 1:   Uint16 value always constant '0xbeef'

- Channel 2:   Uint16 value = (Pixel Index)^2

- Channel 3:   Uint16 value always constant '0x0000'

4.6.8   Raw Distances and Amplitudes

In this mode, raw distance values are transferred in channel 0, and amplitude values are transferred in channel 1.

Raw distance values are forwarded as retrieved from the TIM. No conversion to millimeters and no corrections are performed on the values. Filter settings for raw distance values are ignored as well.

4.6.9   Distance, Amplitudes, and Confidences

In this mode, the distances, amplitudes, and confidences will be transferred in progressive mode, first the distance array (channel 0), then the amplitude array (channel 1) and confidence array (channel 2). The stream starts always with pixel #0.

The **distances** are coded in **millimeters** as **Uint16**. The **amplitudes** pixels are in **Uint16** format. The **confidence** pixels are in Uint8 format.

The confidence is a measure of the confidence of the corresponding depth value. It is an exponential function of a target amplitude to the current amplitude. A value of 255 (maximum value) means 100% confidence, whereas a value of 0 (minimum value) means a confidence of 0%.

First Byte in Stream

Lowbyte of Distance (Pixel 0)	Highbyte of Distance (Pixel 0)	Lowbyte of Distance (Pixel 1)	Highbyte of Distance (Pixel 1)	⋯	Lowbyte of Distance (Pixel 159)	Highbyte of Distance (Pixel 159)
Lowbyte of Distance (Pixel 19040)	Highbyte of Distance (Pixel 19040)	Lowbyte of Distance (Pixel 19041)	Highbyte of Distance (Pixel 19041)	⋯	Lowbyte of Distance (Pixel 19199)	Highbyte of Distance (Pixel 19199)
Lowbyte of Amplitude (Pixel 0)	Highbyte of Amplitude (Pixel 0)	Lowbyte of Amplitude (Pixel 1)	Highbyte of Amplitude (Pixel 1)	⋯	Lowbyte of Amplitude (Pixel 159)	Highbyte of Amplitude (Pixel 159)
Lowbyte of Amplitude (Pixel 19040)	Highbyte of Amplitude (Pixel 19040)	Lowbyte of Amplitude (Pixel 19041)	Highbyte of Amplitude (Pixel 19041)	⋯	Lowbyte of Amplitude (Pixel 19199)	Highbyte of Amplitude (Pixel 19199)
Confidence Pixel 0 (1 Byte)	Confidence Pixel 1 (1 Byte)	Confidence Pixel 2 (1 Byte)	Confidence Pixel 3 (1 Byte)	⋯	Confidence Pixel 318 (1 Byte)	Confidence Pixel 319 (1 Byte)
Confidence Pixel 18880 (1 Byte)	Confidence Pixel 18881 (1 Byte)	Confidence Pixel 18882 (1 Byte)	Confidence Pixel 18883 (1 Byte)	⋯	Confidence Pixel 19198 (1 Byte)	Confidence Pixel 19199 (1 Byte)

Last Byte in Stream

Figure 4-8: Data stream of Distances, Amplitudes, and Confidences

4.6.10 Distances, Amplitudes, Confidences, and Color

In this mode distances and amplitudes from the ToF sensor, confidence values, as well as the color sensor image will be transferred in progressive mode, first the distance array (channel 0), then the amplitude array (channel 1), then the confidence array (channel 2), then the color sensor data (channel 3). The stream starts always with pixel #0.

The **distances** are coded in **millimeters** as **Uint16**, the **amplitudes** also as **Uint16**. The confidence values are in **Uint8** format. The **color** sensor pixels are coded in **RGB565** (raw) or **JPEG** compressed format.

The resolution, format, and frame rate of the color sensor are set via register *ColorStreamParams*.

Channel 3 (which carries the color data) may have zero length, if no color data is available for the current frame. In this case, fields *ColorSensorWidth*, *ColorSensorHeight*, and *ColorSensorLength* of the frame header are set to zero (refer to section 3.2.2).

In the case the color data is transferred in JPEG format, frame header field *ColorSensorLength* contains the length of the JPEG-compressed image in bytes.

4.6.11 Color

This data format is only used for the UDP Color Sensor Stream (see chapter 4.15.1 for more information). This data format cannot be set via register *ImageDataFormat*. When this data format is transferred, field *ImageFormat* of the frame header (→ chapter 3.2.2) is set to value 22 (0x16).

In this mode the color sensor image is transferred in channel #0. The **color** sensor pixels are coded in **RGB565** (raw) or **JPEG** compressed format. In the case the color data is transferred in JPEG format, frame header field *ColorSensorLength* contains the length of the JPEG-compressed image in bytes.

The resolution, format, and frame rate of the color sensor are configured via register *ColorStreamParams*.

4.7 ToF Modulation Frequency

The modulation frequency of the illumination is set to 20 MHz per default. Other modulation frequencies can be set using the register *ModulationFrequency* (for sequence 1: *ModFreqSeq1*). Be aware that this also changes the ambiguity range of the camera.

The following modulation frequencies can be selected:

Index	Frequency
0	5 MHz
1	7.5 MHz
2	10 MHz
3	15 MHz
4	20 MHz
5	25 MHz
6	30 MHz

Table 4-2: Pre-defined modulation frequencies

The register can be written with the frequency in 10-kHz-steps (frequency in Hz/10000) or the index. On a register read, the currently selected modulation frequency (again, in 10-kHz-steps), is returned.

Other frequencies between 5 MHz and 30 MHz can be set as well, but as there is no calibration data available for other frequencies than the indexed ones it may result in unpredictable distance values.

4.8 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 (for sequence 1: IntTimeSeq1)*. The ToF sensor's integration time is limited by hardware to 24ms.

The color sensor is operating at ca. 15 or ca. 30 frames per second. If color data is transferred via the camera's data interface, the frame rate is set automatically depending on register *Framerate*. If color data is transferred via RTP stream, the sensor's frame rate is defined in register *ColorStreamParams*. The color sensor always works with auto exposure control (AEC) and auto gain control (AGC) on.

The maximum frame rate for the ToF sensor is 160 fps but may be reduced by increasing its integration time. 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:

$$FITP = t_{INT}\ [ms] \cdot fps\ \left[\frac{1}{s}\right] \cdot 4$$

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.

The FITP is also used for the camera's power control, if it is powered by PoE (see chapter 0). If the camera register **Integration Time** or **Framerate** is modified, the entered value is used to calculate the FITP. Should the FITP violate the maximum allowed power, then the value is automatically corrected before it is applied. The corrected value can be read back via the same register. Also, if any correction of integration time or frame rate took place because of PoE constraints, Bit[10] of the **Status** register will be set.

4.9 Sequencing

The Sentis-ToF-P510 allows the configuration of up to 2 sequences on the TIM. These sequences are captured immediately after each other by the TIM. Each sequence can be configured with individual integration time and modulation frequency.

If the camera's video mode is disabled (see register **Mode0**), and the camera is manually triggered, each trigger will generate as many sequences as currently configured.

The number of sequences can be configured in register **NofSequ**.
For sequence 0, use registers **IntegrationTime** and **ModulationFrequency** to set integration time and modulation frequency, respectively.
For sequence 1, use registers **IntTimeSeq1** and **ModFreqSeq1**.

Each sequence is processed and streamed by the camera as one individual frame. Each frame header contains the sequence number in field *SequenceNumber*.

4.10 Illumination Pre-Heating

The camera supports pre-heating of the LEDs on the LIMs immediately before capturing. Pre-heating is accomplished by applying a defined modulation signal to the LEDs.

Pre-heating is enabled by setting a pre-heating time. For each sequence, an individual pre-heating time can be configured.

Sequence 0 pre-heating time is configured in register **IllPreheatingTime**.
Sequence 1 pre-heating time is configured in register **IllPreheatingTimeSeq1**.

Common parameters for pre-heating are the frequency of the modulation signal (register **IllPreheatingFreq**) and the duty cycle of the modulation signal (register **IllPreheatingDutyCycle**).

4.11 Distance Offset Calibration

Each Sentis-ToF-P510 is offset-calibrated out of factory.

For each pre-defined modulation frequency, there is an absolute offset in millimeters all distances measured with this frequency are corrected with. The absolute offsets are stored in registers **DistOffset0** (for 5 MHz modulation frequency) to **DistOffset6** (for 30 MHz). You can modify these values by direct register writes.

The camera has also a built-in offset calibration function, which is described in the following procedure. You let the camera know the real distance and the camera will calculate the correct absolute offset. It uses a square of 4x4 pixels in the center of the distance image.

Procedure

Offset Calibration

1. Place your camera co-planar in front of a uniform white target.
2. Avoid extreme environmental light conditions or avoid light completely.
3. Set the desired modulation frequency using register **ModulationFrequency**.
4. Check the amplitude in the center of the image and increase/decrease the **IntegrationTime** until the amplitude is about 3000-5000 in the center.
5. Measure the real distance in millimeters from the camera to the white target. Write this value into register **RealWorldXCoordinate**.
6. Write decimal value 19 into register **CalibrationCommand** and wait until the CalibrationExtended register, Bits 0..7, read decimal value 161 (= finished).
   ➤ The appropriate **DistOffsetX** register is updated.
7. If you want to keep the setting, don't forget to save registers to flash (Chapter 4.16).

4.12 ToF Automatic Exposure Control (AEC)

The Sentis-ToF-P510 provides an automatic exposure control feature which controls the integration time according to the currently observed scene. The AEC is disabled by default and must be enabled in the register **Mode1**.

The AEC is controlled through dedicated registers, which are listed in chapter 5.5.

The AEC algorithm also supports weighing. One may assign specific weights to each of 25 areas into which the sensor area is divided. These weights are inputs to calculate the current amplitude. Please see Figure 4-9: AEC weighing areas for an illustration. Each area's weight is a Uint8 value and can range from 0% (0x0) to 100% (0xf).
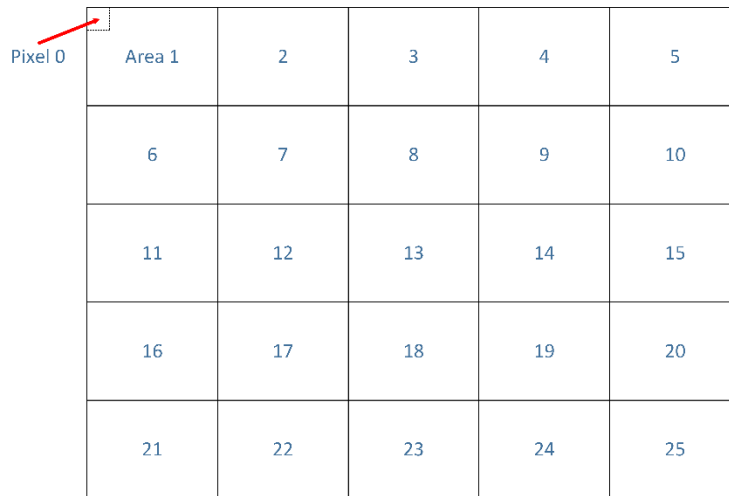
Figure 4-9: AEC weighing areas

4.13 Manual Frame Trigger

There are two types of manual trigger. To enable the manual trigger, the video mode must be disabled in register *Mode0,* Bit[0].

A manual trigger will start capturing of as many sequences as currently configured in register *NofSequ*.

4.13.1 Hardware Trigger

The camera provides an extension connector where a hardware trigger can be applied. Please refer to 3.4 for more information.

4.13.2 Software Trigger

In addition to the hardware trigger a software trigger is available. To start a frame capturing by software, set the appropriate bit (Bit[4]) in register *Mode0*.

4.14 Over Temperature Protection

The Sentis-ToF-P510 firmware has a built-in monitoring for over-temperature condition of the LIMs. If the LIM temperature exceeds 70°C, the camera 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 allowed temperature for the LEDs can be changed in register *MaxLedTemp*. However, we do not recommend to set values larger than 70°C, because it drastically reduces the lifetime of the LEDs.

4.15 Color Sensor

Data from the color sensor can be transmitted in three different ways (The numbers give the priority):

1. Color channel: Send color data as part of the data streaming protocol, along with data from the ToF sensor. Select an appropriate mode in register **ImageDataFormat**. Select resolution, data format (RGB565 or JPEG compressed), and frame rate of the color sensor in register **ColorStreamParams**.

2. UDP Color Sensor Stream: Send color data with the data streaming protocol, but independent from ToF sensor data. A separate destination IP address and destination UDP port can be configured.

3. RTP Color Sensor Stream: Independent from data interface. Use register **ColorStreamParams** for enabling, and controlling resolution, content (H.264 or raw YUV), and frame rate.

See Table 4-3 for a list of available resolutions/frame rates.

Resolution	Frame rates	Horiz. Opening Angle[1]	Vert. Opening Angle[1]
176x144	15, 30	TBD	TBD
320x240	15, 30	103°	77°
640x480	15, 30	103°	77°
720x480	15, 30	103°	69°
720x576	15, 30	95°	76°
1024x768	15, 30	TBD	TBD
1280x720	15, 30	103°	58°
1920x1080	15, 30 [2]	74°	42°

Table 4-3: Color sensor resolution and frame rate options, plus opening angles

Note 1. The specified opening angles are approximate values only. They are valid only with the default lens. Please consult the Hardware User Manual for details.

Note 2. Depending on the camera's configuration, the maximum value may not be reached.

The pixel orientation of the color image sensor is identical to Figure 4-2.

4.15.1 UDP Color Sensor Stream

In use cases where a continuous color sensor stream is required, but only sporadic ToF sensor images, the camera features a second UDP stream that is dedicated to send color sensor data only. It is compliant to the data interface format in chapter 3.2 and hence compatible to tools and libraries provided by Bluetechnix.

The UDP color sensor stream is enabled in register **ColorStreamParams**, bit 2. The resolution, frame rate, and data format (raw RGB565 or JPEG compressed) can also be configured with this register.

The destination for this stream can be configured independently of the ToF data stream, using the following registers:

- Use registers **Eth0UdpColorStreamIp0**, **Eth0UdpColorStreamIp1** to set the destination IP address.

- Use register **Eth0UdpColorStreamPort** to set the destination UDP port.

> **Note**
>
> If any image data format with color sensor data is selected in register **ImageDataFormat**, the UDP Color Sensor Stream is automatically disabled, and the color sensor data is transmitted in the configured channel along with data from the ToF sensor.

It is recommended to use this stream in favor of the RTP Color Sensor Stream.

4.15.2 RTP Time Stamp

As defined in RFC 1889 by the IETF, each RTP packet header contains a 32-bit timestamp value. RTP uses a clock rate of 90 kHz, i.e., this timestamp increments by 90000 each second.

The Sentis-ToF-P510 uses the same internal time base for this RTP timestamp and for the ToF data stream timestamps (which are part of the image header, see chapter 3.2). This is required for the receiver, in order to relate each received RTP and ToF frame on a time line.

Note that the resolution of both time stamps is different: Whereas the ToF data stream's resolution is 1us (1MHz clock), the RTP stream's resolution is 11.1us (90kHz). To convert the RTP time stamp value to microseconds as well, use the formula

$$T_{2D} = \frac{100 \, T_{RTP}}{9}$$

4.16 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.

4.17 Ethernet/IP Settings

4.17.1 MAC Address

A dedicated Ethernet MAC address from Bluetechnix MAC address pool is assigned to each Sentis-ToF-P510 by factory default. This MAC address is saved in the OTP and cannot be changed by the user.

The user is allowed to assign the camera 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.

4.17.2 IP/TCP/UDP Settings

The IP Settings of the Sentis-ToF-P510 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.

4.18 Reset to Factory Default

The Sentis-ToF-P510 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*.

Alternatively, a factory reset is executed via the camera's reset signal. (Please consult the Hardware User Manual for details.) It must be active until the firmware is completely booted and the data stream is present.

4.19 Firmware Update

The Sentis-ToF-P510 is capable of updating its firmware (as well as the boot loader). The update procedure is executed using dedicated TCP/IP command frames over the control interface connection.

Bluetechnix provides the "Downloader" tool within the BltTofSuite for updating the camera firmware over Ethernet.

> **Note**
>
> On any firmware update, the RTP Stream is stopped automatically.
>
> In order to complete a firmware update, a complete reboot of the camera is required. The camera will NOT reboot automatically.

4.19.1 Firmware Recovery

If a new firmware fails to load for 3 times, the Sentis-ToF-P510 boot loader will recover the old firmware automatically.

You can visibly detect this issue if the color LED flashes red and yellow for three times consecutively, without flashing green in between. After a firmware recovery, Bit[8] of the **Status** register is set.

The camera 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 camera.

4.20 Logging

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

Log data may be retrieved using the Secure Shell login (see chapter 3.8) and can be found at `/mnt/logs/messages*` files. Newest log data is contained in file `messages`.

4.21 Error Indication

The Sentis-ToF-P510 indicates detected errors mainly in the **Status** register:

- Bit 3: Indicates a temperature measurement error on at least one LIM. The bit is automatically cleared if the error disappears.

- Bit 4: Indicates a temperature measurement error on the TIM. The bit is automatically cleared if the error disappears.

- Bit 8: There were errors during firmware boot, the previous firmware was discovered. This bit is not set again after a reboot.

- Bit 11: The status of at least one LIM could not be retrieved, or there is an error condition in one of the LIM status registers **Lim1Status** and **Lim2Status**.

- <u>Bit 13</u>: Error starting the RTP color sensor stream, or error on retrieving color sensor data (if register *ImageDataFormat* is set to either "Distances and Color" or "Distances, Amplitudes, and Color" mode). This bit is cleared on disabling the RTP stream (register *ColorStreamParams*, bit 0).

- <u>Bit 14</u>: Indicates a temperature measurement error on the base board. The bit is automatically cleared if the error disappears.

- <u>Bit 15</u>: Indicates a communication error with the TIM, or an error with triggering the TIM. This bit is automatically cleared if the error disappears.

5   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.

5.1   General registers

Addr (hex)	Register Name	Default Value (hex)	R/W	Description
0001	Mode0	0001	R/W	Bit[0]: 0…Manual Mode, 1…Video Mode Bit[4]: 1…Manual Trigger (self-clearing bit) Bit[6]: 1…Clear status register (self-clearing bit)
0003	Status	0040	R	Bit[2]: 1…Ongoing Calibration Bit[3]: 1…LIM temperature sensor error (Temperature sensors not found, or temporary temperature reading error) Bit[4]: 1…TIM 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…LIM over-temperature Bit[10]: 1…Frame rate or integration time was limited due to PoE constraints Bit[11]: 1…Error condition reported by one of the LIMs, see registers **Lim1Status** and **Lim2Status** for details Bit[13]: 1…Color sensor stream error Bit[14]: 1…Base board temperature sensor error Bit[15]: 1…TIM error (I2C communication, triggering)
0004	ImageDataFormat	0000	R/W	Bit[3:10]: 　0 … Distances and Amplitudes 　1 … Distances, Amplitudes, and Confidences 　2 … Distances, Amplitudes and Color 　3 … XYZ Point Cloud 　4 … XYZ Point Cloud and Amplitudes 　6 … Distances and Color 　9 … Distances and XYZ Point Cloud 　10 … X Coordinate and Amplitudes 　11 … Test mode 　12 … Distances 　13 … Raw Distances and Amplitudes 　21 … Distances, Amplitudes, Confidences, and Color
0005	IntegrationTime	05DC	R/W	Integration Time [µs] (min: 1, max: 24000)
0006	DeviceType	B320	R	Hardware specific identification
0007	DeviceInfo		R	Bit[0-3]: PCB Revision 0…Unknown 1…V2.1

Addr (hex)	Register Name	Default Value (hex)	R/W	Description
				2…V2.2 3…V2.3
0008	FirmwareInfo		R	Bit[0-5]: Non Functional Revision Bit[6-10]: Minor Revision Bit[11-15]: Major Revision
0009	ModulationFrequency	07D0	R/W	Modulation frequency index: 0…5 MHz 1…7.5 MHz 2…10 MHz 3…15 MHz 4…20 MHz 5…25 MHz 6…30 MHz or a modulation frequency in multiples of 10kHz (may take a while to apply and no calibration data will be available)
000A	Framerate	0028	R/W	ToF frame rate [Hz]
000B	HardwareConfiguration		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 for the current modulation frequency. Exactly one sequence must be configured in register *NofSequ*. 16…Clear FPPN calibration data for the current modulation frequency. Always takes modulation frequency of first sequence! 19…Calibrate DistOffset for the current modulation frequency. Exactly one sequence must be configured in register *NofSequ*.
0010	ConfidenceThresLow	012C	R/W	Amplitude threshold for valid distance data
0011	ConfidenceThresHigh	3A98	R/W	Amplitude threshold for valid distance data
0019	Mode1	0000	R/W	Bit[3]: 0..AEC Off, 1..AEC On
001B	LedboardTemp		R	Temperature of LIMs in 0.01[°C] (FFFF: Sensor not available).
001C	MainboardTemp		R	Temperature of ToF sensor in 0.01[°C] (FFFF: Sensor not available).
0020	RealWorldXcoordinate	0000	R/W	Distance to the calibration target [mm].
0021	CalibrationExtended	0000	R	Bit[0-7]: Status/error 0…Idle 19…FPPN calibration 20…Erasing flash 21…DistOffset calibration 161…Operation done 246…Wrong image mode (Need distance) or Mode0 setting (Need video mode) 248.. Invalid modulation frequency 255…Generic error Bit[10]: 1…Error occurred Bit[12]: 1…No FPPN Calibration data in NVM

Addr (hex)	Register Name	Default Value (hex)	R/W	Description
				Bit[14]: 1…No Lens Calibration data in NVM
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 LIM temperature 0.01[°C]
0026	HorizontalFov	2)	R	Horizontal field of view in 0,01[°]
0027	VerticalFov	2)	R	Vertical field of view in 0,01[°]
002B	TriggerDelay	0000	R/W	Delay between input trigger assertion (either software or hardware) and output trigger assertion (image capturing, trigger output) [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 * x^3 + b/10000 * x^2 + c/1000 * x
0030	TempCompGradient2Lim		R/W	Factor 'b' of the illumination temperature compensation function: y [mm] = a/100000 * x^3 + b/10000 * x^2 + c/1000 * x
0032	TimVersion		R	Version of the firmware on the TIM Bit[0-5]: Non Functional Revision Bit[6-10]: Minor Revision Bit[11-15]: Major Revision
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 this register 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
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 temperature compensation function: y [mm] = a/100000 * x^3 + b/10000 * x^2 + c/1000 * x
003D	BuildYearMonth		R	Firmware Build date/time

Addr (hex)	Register Name	Default Value (hex)	R/W	Description
				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]
0043	TimSerialLow		R	Serial Number of the TIM, low word
0044	TimSerialHigh		R	Serial Number of the TIM, high word
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
0047	RgbLedColor	0300	R/W	RGB565 color value of RGB LED
0048	Lim1Status	0000	R	Status word of LIM #1 or external illumination. Bits[0..4]: Overcurrent on LED segments 0..4 Bits[8..12]: Open load on LED segments 0..4 Bit[14]: Temperature sensor error Bit[15]: Could not read status word (communication failure)
0049	Lim2Status	0000	R	Status word of LIM #2 (Bit description see register *Lim1Status*)
004A	TempCompGradientTim		R/W	Factor 'c' of the ToF sensor temperature compensation function: y [mm] = a/100000 * x³ + b/10000 * x² + c/1000 * x
004B	TempCompGradient2Tim		R/W	Factor 'b' of the ToF sensor temperature compensation function: y [mm] = a/100000 * x³ + b/10000 * x² + c/1000 * x
004C	TempCompGradient3Tim		R/W	Factor 'a' of the ToF sensor temperature compensation function: y [mm] = a/100000 * x³ + b/10000 * x² + c/1000 * x

Table 5-1: General registers

Note 2): The content depends on the mounted lens and the calibration data and represents the real viewing angles.

5.2 Registers for GPIO Control

Addr (hex)	Register Name	Default Value (hex)	R/W	Description
00D0	IOstate0	00--	R/W	Bit[0]: … state of IN_0 (only R) Bit[8]: … state of OUT_0 (R/W)
00D1	Buzzer	0000	R/W	Bit[0]: 1 … Enable buzzer

Table 5-2: Registers for GPIO Control

5.3   Registers for Color Image

Addr (hex)	Register Name	Default Value (hex)	R/W	Description

00E0	ColorStreamParams	0022	R/W	This register configures various options to transmit data from the color sensor.
				
				Bit[0]: 1…Enable autonomous RTP stream
				Bit[1]: Color channel format (for image modes with color sensor data, and for the autonomous UDP data stream)
				    0 … RGB565
				    1 … JPEG compressed
				Bit[2]: 1…Enable autonomous UDP data stream
				Bit[4..6]: Color sensor resolution
				    0 … QCIF (176x144)
				    1 … QVGA (320x240)
				    2 … VGA (640x480)
				    3 … NTSC (720x480)
				    4 … PAL (720x576)
				    5 … XGA (1024x768)
				    6 … 720P (1280x720)
				    7 … 1080P (1920x1080)
				Bit[8]: Color sensor frame rate
				    0 … 15 fps
				    1 … 30 fps
				Bit[13]: RTP stream format
				    0 … H.264-in-RTP video
				    1 … Raw-in-RTP video
				    (YCbCr 4:2:0)

Table 5-3: Color Image Registers

5.4   More General Registers

Addr (hex)	Register Name	Default Value (hex)	R/W	Description
00C1	DistOffset0		R/W	An offset for distance values when operating at modulation frequency with index 0 (5 MHz)
00C2	DistOffset1		R/W	An offset for distance values when operating at modulation frequency with index 1 (7.5 MHz)
00C3	DistOffset2		R/W	An offset for distance values when operating at modulation frequency with index 2 (10 MHz)
00C4	DistOffset3		R/W	An offset for distance values when operating at modulation frequency with index 3 (15 MHz)
00C5	DistOffset4		R/W	An offset for distance values when operating at modulation frequency with index 4 (20 MHz)

Addr (hex)	Register Name	Default Value (hex)	R/W	Description
00C6	DistOffset5		R/W	An offset for distance values when operating at modulation frequency with index 5 (25 MHz)
00C7	DistOffset6		R/W	An offset for distance values when operating at modulation frequency with index 6 (30 MHz)
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
010A	TempCompGradientBaseboard		R/W	Factor 'c' of the ToF base board temperature compensation function: y [mm] = a/100000 * x³ + b/10000 * x² + c/1000 * x + u
010B	TempCompGradient2Baseboard		R/W	Factor 'b' of the ToF base board temperature compensation function: y [mm] = a/100000 * x³ + b/10000 * x² + c/1000 * x + u
010C	TempCompGradient3Baseboard		R/W	Factor 'a' of the ToF base board temperature compensation function: y [mm] = a/100000 * x³ + b/10000 * x² + c/1000 * x + u
010D	BaseboardTemp		R	Temperature of baseboard in 0,01[°C] (FFFF: Sensor not available).
0110	IllPreheatingTime	0	R/W	Sequence 0 illumination preheating 0 … No pre-heating 100 - 30000 … Time for illumination pre-heating in microseconds

Table 5-4: More General registers

5.5 Registers for Sequencing

Addr (hex)	Register Name	Default Value (hex)	R/W	Description
0120	NofSequ	1	R/W	Number of sequences recorded by the ToF sensor without wait time in between
0121	IntTimeSeq1	05DC	R/W	Integration time to be used for recording sequence 1 NOTE: Sequence 0 integration time is set via register *IntegrationTime*
0128	ModFreqSeq1	07D0	R/W	Modulation frequency to be used for recording sequence 1 Modulation frequency index: 0.. 5 MHz

Addr (hex)	Register Name	Default Value (hex)	R/W	Description
				1.. 7,5 MHz 2.. 10 MHz 3.. 15 MHz 4.. 20 MHz 5.. 25 MHz 6.. 30 Mhz or a modulation frequency in multiples of 10kHz NOTE: Sequence 0 modulation frequency is set via register ***ModulationFrequency***

Table 5-5: Registers for Sequencing

5.6   Registers for Illumination Preheating

Addr (hex)	Register Name	Default Value (hex)	R/W	Description
0150	IllPreheatingFreq	0064	R/W	Frequency of the pre-heating modulation signal in multiples of 10kHz Allowed values: 100…1500 (1MHz…15MHz)
0151	IllPreheatingDutyCycle	0032	R/W	Duty cycle of the pre-heating modulation signal in percent Allowed values: 10…90
0152	IllPreheatingTimeSeq1	0000	R/W	Sequence 1 illumination preheating 0 … No pre-heating 100 - 30000 … Time for illumination pre-heating in microseconds NOTE: Sequence 0 illumination preheating time is set via register **IllPreheatingTime**

Table 5-6: Registers for Illumination Pre-heating

5.7   Registers for Automatic Exposure Control

Addr (hex)	Register Name	Default Value (hex)	R/W	Description
01A9	AecAvgWeight0	4444	R/W	Bit[15-12]: Weight for average, area 1 Bit[11-8]: Weight for average, area 2 Bit[7-4]: Weight for average, area 3 Bit[3-0]: Weight for average, area 4
01AA	AecAvgWeight1	44CC	R/W	Bit[15-12]: Weight for average, area 5 Bit[11-8]: Weight for average, area 6 Bit[7-4]: Weight for average, area 7 Bit[3-0]: Weight for average, area 8
01AB	AecAvgWeight2	C44C	R/W	Bit[15-12]: Weight for average, area 9

Addr (hex)	Register Name	Default Value (hex)	R/W	Description
				Bit[11-8]: Weight for average, area 10 Bit[7-4]: Weight for average, area 11 Bit[3-0]: Weight for average, area 12
01AC	AecAvgWeight3	FC44	R/W	Bit[15-12]: Weight for average, area 13 Bit[11-8]: Weight for average, area 14 Bit[7-4]: Weight for average, area 15 Bit[3-0]: Weight for average, area 16
01AD	AecAvgWeight4	CCC4	R/W	Bit[15-12]: Weight for average, area 17 Bit[11-8]: Weight for average, area 18 Bit[7-4]: Weight for average, area 19 Bit[3-0]: Weight for average, area 20
01AE	AecAvgWeight5	4444	R/W	Bit[15-12]: Weight for average, area 21 Bit[11-8]: Weight for average, area 22 Bit[7-4]: Weight for average, area 23 Bit[3-0]: Weight for average, area 24
01AF	AecAvgWeight6	4000	R/W	Bit[15-12]: Weight for average, area 25
01B0	AecAmpTarget	02BC	R/W	Auto exposure target amplitude value to which the controller is controlling to
01B1	AecTintStepMax	0021	R/W	Auto exposure maximum change of integration time percentage. The relative change of the integration time will be lower than this percentage
01B2	AecTintMax	2710	R/W	Auto exposure maximum integration time the controller calculates
01B3	AecKp	0028	R/W	Proportional part of the auto exposure controller in percent
01B4	AecKi	000F	R/W	Integral part of the auto exposure controller in percent
01B5	AecKd	0000	R/W	Differential part of the auto exposure controller in percent

Table 5-7: Registers for automatic exposure control

5.8 Registers for Test Commands

Addr (hex)	Register Name	Default Value (hex)	R/W	Description
01C0	TestConfig	0000	R/W	Bit[1]: 1… Watchdog Test Bit[2]: 1…Simulate color sensor module failure Bit[3]: 1…Simulate ToF sensor module failure Bit[4]: 1…Manually power off camera with UVLO detector Writing this register must be preceded by writing 0x5E6B into register CmdEnablePasswd (0x0022)

Table 5-8: Registers for Test Commands

5.9 Registers for 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

Addr (hex)	Register Name	Default Value (hex)	R/W	Description
				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 5-9: Registers for device update

5.10 Registers for Filter Configuration

Addr (hex)	Register Name	Default Value (hex)	R/W	Description
01E0	ImgProcConfig	28C0	R/W	Bit[0]: 1…enable Median Filter for Distance Image Bit[3]: 1…enable Bilateral Filter for Distance Image Bit[4]: 1…enable Sliding Average for Distance Image Bit[6]: 1…enable wiggling compensation for Distance Image Bit[7]: 1…enable FPPN compensation for Distance Image Bit[10]: 1…enable FrameAverage Filter for Distance Image Bit[11]: 1…enable temperature compensation for Distance Image Bit[13]: 1…enable offsets via registers *DistOffsetX* (0x00C1 onwards) for Distance Image
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[0-7]: Number of Frames
01E9	ImgProcConfig2	0000	R/W	Bit[0]: 1… enable Median Filter for Amplitude Image Bit[1]: 1… enable Bilateral Filter for Amplitude Image Bit[2]: 1… enable Sliding Average Filter for Amplitude Image Bit[3]: 1… enable FrameAverage Filter for Amplitude Image. Bit 10 of register *ImgProcConfig* must be set.

Table 5-10: Register for filter configuration

5.11 Registers for 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
0250	PoEStatus		R	Bit[0..2]: Detected PoE power class 1: PoE (13W limitation) 2: PoE+ (25.5W limitation) 3: PoE++ (no power limitation) 7: VAUX (no power limitation)
0251	PoEOverride	0000	R/W	Bit[0..1]: Override PoE power class 0: Don't override 1: 13W limitation (PoE) 2: 25.5W limitation (PoE+) 3: No limitation (PoE++) **Note: This limit is only checked against if integration time (0x0005) or frame rate**

Addr (hex)	Register Name	Default Value (hex)	R/W	Description
				(0x000a) is set. No immediate action is taken when this register is written.
0252	Eth0UdpRtpStreamIp0	0001	R/W	Low word of destination IP address for RTP color sensor stream Writing this register has no immediate effect.
0253	Eth0UdpRtpStreamIp1	E000	R/W	High word of destination IP address for RTP color sensor stream **Writing this register will update the RTP color sensor stream address.**
0254	Eth0UdpRtpStreamPort	2714	R/W	UDP port for RTP color sensor stream
0256	Eth0UdpColorStreamIp0	0001	R/W	Low word of destination IP address for Color sensor UDP data stream
0257	Eth0UdpColorStreamIp1	E000	R/W	High word of destination IP address for Color sensor UDP data stream
0258	Eth0UdpColorStreamPort	2716	R/W	UDP port for Color sensor UDP data stream

Table 5-11: Registers for Ethernet configuration

6 Firmware History

6.1 Version Information

Firmware Version	Status	Release date	Changes
0.1.0	X-Grade	May 2014	

0.2.0	X-Grade	Dec 2014	• [Bug] Factory reset deletes calibrated distance offsets [1]
			• [Feature Request] Support new image mode: DIST-COLOR
			  ➔ See register **ImageDataFormat**
			• [Usability] XYZ Modes: Change X value of underexposed pixels (32767), invalid pixels (1), and overexposed pixels (0), Y = Z = 0!
			• [Feature Missing] Color sensor: Add two more resolutions: QCIF and XGA
			  ➔ See register **ColorStreamParams**
			• [Bug] Color sensor: Auto exposure not turned on again after resolution changed
			• [Feature Request] Change installed library from M100API to BltTofApi (EthLib)
			• [Feature Request] Don't lock application to a single core, give control to Linux scheduler
			• [Feature Request] New Image mode 2: DIST-AMP-COLOR (including color sensor raw stream in 3rd channel)
			  ➔ See register **ImageDataFormat**
			• [Feature Request] Color sensor: Implement YUV420-over-RTP mode
			  ➔ See register **ColorStreamParams**
			• [Feature Request] Read vertical and horizontal FOV register values from lens calibration file, instead of calculating them
			  ➔ See registers **HorizontalFOV** and **VerticalFOV**
			• [Feature Missing] Temperature compensation for ToF sensor required
			  ➔ See registers **TempCompGradientTim**, **TempCompGradient2Tim**, **TempCompGradient3Tim**
			• [Usability] Change pixel invalidation priority: INVALID, then OVER/UNDEREXPOSED.
			• Image Header was extended due to new image data formats.
			
			[1] Please contact Bluetechnix support for information how to store factory defaults.

0.4.0	X-Grade	Feb 2015	• [Feature request] Add base board temperature compensation.
			  ➔ New registers **TempCompGradientBaseboard**, **TempCompGradient2Baseboard**, **TempCompGradient3Baseboard**
			• [Feature Request] Change expansion factor for all temperature compensation gradient x^3 registers from 1/10000 to 1/100000
			  ➔ See registers **TempCompGradient3Tim**, **TempCompGradient3Lim**

0.5.0	X-Grade	May 2015	• Remove libbta_eth.so, it will be installed by Pxxx Linux Development Package

Firmware Version	Status	Release date	Changes
			• Change order of applying filters: 1. Average, 2. Sliding Average, 3. Median, 4. Bilateral • Support RGB LED ➔ See new register **RgbLedColor** • Support Pxxx Linux Development Package
0.6.0	X-Grade	Jul 2015	• Detect base board revision and disable Factory Reset function on base board version 1.0.x. ➔ New Register *DeviceInfo* • Fix calculation of horizontal and vertical field of view (registers *HorizontalFov* and *VerticalFov*) • Fix slightly incorrect depth image correction with FPPN calibration data

0.7.0	X-Grade	Sep 2015	• Support up to 2 sequences with individual integration times and modulation frequencies
			  ➔ See chapter 4.9
			• Support trigger chip and enable S/W and H/W manual trigger again
			  ➔ Chapters 3.4 and 4.13
			• Support illumination pre-heating
			  ➔ See chapter 4.10
			• Add a stand-alone UDP stream with color sensor data
			  ➔ See chapters 4.15.1 and 4.6.11
			• Support image filters for amplitude data
			  ➔ See register *ImgProcConfig2*
			• Add image data formats with a confidence channel
			  ➔ See chapters 4.6.9 and 4.6.10, register *ImageDataFormat*
			• Add JPEG compression for color sensor – available in image data formats (ToF and color data) and in separate color-only UDP stream
			  ➔ See register *ColorStreamParams*
			• Change control of color sensor framerate: Via *ColorStreamParams* register bit 8 only – no more influence of ToF sensor framerate on color sensor framerate
			• Add test commands for simulation of color sensor failure, ToF sensor failure, under-voltage
			  ➔ See register *TestConfig*
			• Add internal monitoring of the TIM and a restart procedure in case of a detected failure
			• Support buzzer
			  ➔ See register *Buzzer*
			• Add flash read and write of (proprietary) lens parameter configurations – not read by the camera internally, but by BltTofSuite
			  ➔ See chapters 3.1.5 and 3.1.6
			• Improve internal error-checking
			  ➔ See register *Status*
			• Add plausibility check for TIM and base board temperature sensor values
			• Use frame counter from TIM instead of generating it in firmware
			• Improve overall image processing/compensation performance by means of multi-threading

Table 6-1: Overview Sentis-ToF-P510 firmware changes

Note

Please refer to our support site for additional information about product changes.

6.2 Anomalies

Applies to	Date	Description
0.1.0, 0.2.0	2014 05 26	ToF sensor values are not initialized correctly in rare cases. Users may detect this issue if UDP frames are sent with 5.0fps instead of register setting. Temporary fix: Reboot camera.
0.1.0	2014 05 26	PoE class detection sometimes wrong. Users may detect his issue using PoEStatus register. Temporary fix: Override class with PoEOverride register if camera is powered with PoE.
0.5.0	2015 07 07	On base board version 1.0.x (170-0033), due to a hardware fault, on each boot a factory default reset is executed
0.5.0	2015 07 07	HorizontalFov and VerticalFov registers are calculated wrongly.
0.5.0	2015 07 07	Depth data correction with FPPN data is inexact

Table 6-2: Firmware anomalies

7 Software

7.1 BltTofApi

SDK for ToF products: [Bluetechnix 'Time of Flight' API](#)

In order to create a common interface for our products we define the interfaces between a ToF device and an application. The main part of this model is the *BltTofApi* which is written in C for platform independency.

The library which provides this API for Ethernet-based devices as the Sentis-ToF-P510 is the *BtaEthLib* (*BltTofApi* Ethernet Library).

Please visit our support Wiki to get information and to download the SDK.

Bluetechnix 'Time of Flight' API

https://support.bluetechnix.at/wiki/ (Section Software)

7.2 MATLAB SDK

MATLAB SDK for ToF products: [BltTofApi Matlab SDK](#)

The MATLAB SDK is able to access the *BltTofApi* interface and will therefore be compatible with any device with an existing library implementing the *BltTofApi*.

Bluetechnix 'Time of Flight' API Matlab SDK

https://support.bluetechnix.at/wiki/ (Section Software)

7.3 BltTofSuite

For the first evaluation of the camera and to evaluate different settings and configurations a .NET demo application for Microsoft Windows is provided: *BltTofSuite*. The demo application can be downloaded from our support web site.

Software and documentation

https://support.bluetechnix.at/index.html

8   Support

8.1   General Support

General support for products can be found at Bluetechnix' support site

Support Link

⇨   https://support.bluetechnix.at/index.html

8.2   Software Downloads

Camera support packages are available for registered customers only. Please contact Bluetechnix support if you do not yet have an account.

Software Download Portal

⇨   https://support.bluetechnix.at/software/

8.3   Related Products

- TIM$^{\mu P}$-19kS3-Spartan 6
- LIM^{U}-LED-850

8.4   Camera Development Package

The camera offers the possibility to bring your own application onto the Sentis-ToF-P510.

The Sentis-ToF-P510 is based on an embedded ARM Linux system based on the i.MX6 Quad-core processor from Freescale Inc.

Please contact Bluetechnix support for more information.

9 Document Revision History

Version	Date	Document Revision
1	2014 05 23	Preliminary version of the document
2	2014 12 15	Added firmware changes from 0.1.0 to 0.2.0 firmware version (Chapter 6.1) Updated register description (Chapters 5.1, 5.2) Added data format description for modes with RGB565 channel (Chapters 0 and 4.6.1) Updated pixel invalidation description (Chapter 4.4) Added offset calibration procedure (Chapter 0)
3	2015 02 19	Added firmware changes from 0.2.0 to 0.4.0 firmware version (Chapter 6.1) Updated register description (Chapters 5.1, 5.2)
4	2015 05 15	Added firmware changes from 0.4.0 to 0.5.0 firmware version (Chapter 6.1) Updated register description (Chapter 5.1)
5	2015 07 08	- Added firmware changes from 0.5.0 to 0.6.0 firmware version (Chapter 6.1) - Fixed device type for this device in chapter 5.1 - Added 1 register to chapter 5.1
6	2015 09 11	- Made 3.1.2 (Result codes) a dedicated chapter - Updated chapter 3.1.5 (Flash update) - Added chapter 3.1.6 (Read from flash – new function) - Updated chapter 3.2.2 (Frame header) - Added chapter 3.4 (Manual frame triggers) - Updated chapter 3.5 (External Illumination) - Updated chapter 3.6 (Buzzer) - Updated chapter 4.2 (Image Processing Chain) - Updated chapter 4.3 (Order of filters) - Added chapter 4.4.2 (Temperature compensation) - Updated chapter 4.6 (JPEG format) - Added chapters 4.6.8, 4.6.9, 4.6.10, 4.6.11 (New image data formats) - Added chapters 4.9, 4.10 (Sequencing, Illumination Pre-heating) - Added a chapter on manual frame triggers: 4.13 (including new Snapshot function) - Added chapter 4.15.1 (New independent UDP color sensor stream) - Added chapter 4.21 (Error indication) - Updated chapters 5.2, 5.3, 5.4, 5.10, 5.11 (Register description) - Added chapters 5.5, 5.6 (Register description) - Updated chapter 6.1 with firmware changes on V0.7.0 firmware - Revised chapters 7 and 8

Table 9-1: Revision history

A Index

B   List of Figures and Tables

Figures

Tables