

# Sentis-ToF-M100 API

## 1.0.0

Generated by Doxygen 1.8.3.1

Wed Jan 15 2014 10:12:20



# Contents

<b>1</b>	<b>Sentis-ToF-M100 camera API</b>	<b>1</b>
1.1	Introduction . . . . .	1
1.2	Installation . . . . .	1
1.2.1	Linux . . . . .	1
1.2.2	Windows . . . . .	1
1.3	Getting Started . . . . .	2
1.3.1	Network configuration . . . . .	2
1.3.2	Samples . . . . .	2
1.3.3	Linux . . . . .	2
1.3.4	Windows . . . . .	3
<b>2</b>	<b>Module Index</b>	<b>5</b>
2.1	Modules . . . . .	5
<b>3</b>	<b>Data Structure Index</b>	<b>7</b>
3.1	Data Structures . . . . .	7
<b>4</b>	<b>File Index</b>	<b>9</b>
4.1	File List . . . . .	9
<b>5</b>	<b>Module Documentation</b>	<b>11</b>
5.1	Flags macros . . . . .	11
5.1.1	Detailed Description . . . . .	11
5.1.2	Macro Definition Documentation . . . . .	11
5.1.2.1	HOLD_CONTROL_ALIVE . . . . .	11
5.1.2.2	SENTIS_NON_BLOCKING_INPUT . . . . .	11
5.2	Registers macros . . . . .	12
5.2.1	Detailed Description . . . . .	12
5.3	Image data format macros . . . . .	13
5.3.1	Detailed Description . . . . .	13
<b>6</b>	<b>Data Structure Documentation</b>	<b>15</b>
6.1	T_SENTIS_CONFIG Struct Reference . . . . .	15

6.1.1	Detailed Description	15
6.1.2	Field Documentation	15
6.1.2.1	flags	15
6.1.2.2	tcp_ip	15
6.1.2.3	tcp_port	15
6.1.2.4	udp_ip	16
6.1.2.5	udp_port	16
6.2	T_SENTIS_DATA_HEADER Struct Reference	16
6.2.1	Detailed Description	16
6.2.2	Field Documentation	16
6.2.2.1	bytesPerPixel	16
6.2.2.2	firmwareVersion	16
6.2.2.3	frameCounter	17
6.2.2.4	headerCrc16	17
6.2.2.5	headerVersion	17
6.2.2.6	imageFormat	17
6.2.2.7	imageHeight	17
6.2.2.8	imageWidth	17
6.2.2.9	ledTemp	17
6.2.2.10	mainTemp	17
6.2.2.11	nofChannels	17
6.2.2.12	reserved1	17
6.2.2.13	reserved2	17
6.2.2.14	reserved3	17
6.2.2.15	timestamp	18
<b>7</b>	<b>File Documentation</b>	<b>19</b>
7.1	include/apitypes.h File Reference	19
7.1.1	Detailed Description	20
7.1.2	DESCRIPTION	21
7.1.3	Typedef Documentation	21
7.1.3.1	T_ERROR_CODE	21
7.1.3.2	T_SENTIS_HANDLE	21
7.2	include/m100api.h File Reference	21
7.2.1	Detailed Description	22
7.2.2	DESCRIPTION	22
7.2.3	Function Documentation	22
7.2.3.1	STSclose	22
7.2.3.2	STSgetData	22
7.2.3.3	STSopen	23

7.2.3.4	STReadRegister . . . . .	23
7.2.3.5	STWriteRegister . . . . .	24
7.3	glut-3d/3d.cpp File Reference . . . . .	24
7.3.1	Detailed Description . . . . .	26
7.3.2	Macro Definition Documentation . . . . .	26
7.3.2.1	TICKS_PER_SECOND . . . . .	26
7.3.3	DESCRIPTION . . . . .	26
7.3.4	Function Documentation . . . . .	26
7.3.4.1	drawAxes . . . . .	26
7.3.4.2	getFrame . . . . .	26
7.3.4.3	initCamera . . . . .	26
7.3.4.4	stopCamera . . . . .	27
7.4	glut-distances/distances.cpp File Reference . . . . .	27
7.4.1	Detailed Description . . . . .	28
7.4.2	Macro Definition Documentation . . . . .	28
7.4.2.1	TICKS_PER_SECOND . . . . .	28
7.4.3	DESCRIPTION . . . . .	28
7.4.4	Function Documentation . . . . .	29
7.4.4.1	drawAxes . . . . .	29
7.4.4.2	getFrame . . . . .	29
7.4.4.3	initCamera . . . . .	29
7.4.4.4	stopCamera . . . . .	29
7.5	register-printer/register.cpp File Reference . . . . .	29
7.5.1	Detailed Description . . . . .	30
7.5.2	Function Documentation . . . . .	30
7.5.2.1	main . . . . .	30
7.5.3	DESCRIPTION . . . . .	30



# Chapter 1

## Sentis-ToF-M100 camera API

### 1.1 Introduction

#### Sentis-ToF-M100 camera API

This software enables network communication with the camera Sentis-ToF-M100 from Bluetechnix GmbH. It enables camera control via read and write access to the camera registers via TCP and it also provides an asynchronous reader thread for capturing image data that is sent out by the camera via UDP packets on a multicast channel.

### 1.2 Installation

#### 1.2.1 Linux

For Debian derived distributions as Ubuntu you can use the .deb package for your system architecture. If you work with other linux distributions you can find in the same folder the compiled library and the headers in a tar.gz file. The sample applications will look for these files in common system paths (/usr/, /usr/local/). If you decide to have the library somewhere else, remember to change the cmake module files in samples/ to find it.

In order to compile the sample applications you will need the following dependencies installed:

```
sudo get-apt install build-essential cmake freeglut freeglut-dev
```

#### 1.2.2 Windows

To use the library in windows you can either use the dynamic library or the static library version. The sample applications were compiled with the dynamic version, therefore we have added a bat file to easily execute them. You can also add the .dll file from windows/dll/<arch>/ to your system path (i.e. c:/windows/system32).

You will also need the Visual C++ Redistributable for your system version. The Visual C++ Redistributable Packages install runtime components that are required to run C++ applications built with Visual Studio. You can get them from the Download Center web site of Microsoft:

Visual C++ Redistributable for Visual Studio 2012: <http://www.microsoft.com/en-us/download/details.aspx?id=30679>

You need also need the 2010 version:

Microsoft Visual C++ 2010 Redistributable Package (x64): <http://www.microsoft.com/en-us/download/details.aspx?id=14632>

Microsoft Visual C++ 2010 Redistributable Package (x86): <http://www.microsoft.com/en-us/download/details.aspx?id=5555>

The windows version of the library depends on the pthreads-w32 library. You to install this library to your computer as well. You can get it from <http://www.sourceware.org/pthreads-win32/> or you can use the compiled version we have added in windows/libraries/pthreads-w32. Copy the .dll file in your system path.

To execute the samples, you will also need the freeglut utilities library for windows. You can get it from <http://freeglut.sourceforge.net/> or use the dll found in windows/libraries/freeglut/.

## 1.3 Getting Started

### 1.3.1 Network configuration

To start communicating with your Sentis-ToF-M100 camera you need to configure your network connections: By default the M100 has the local IP "192.168.0.10" as control address set in its registers. Please choose a network address in the same subnet. The camera's IP can be changed later on. For the data udp multicast connection the default address is "224.0.0.1" The ports used by the camera to send the network packets are:

tcp=10001, udp=10002.

Make sure these ports are not been used. You could change them later by setting the right camera register. Please refer to the Sentis-ToF-M100 documentation to know which registers contain the network parameters and to get more information about the camera.

The included sample applications use this configuration to create the connections with the device.

In order to be sure that the communication with the camera is correct, we recommend to execute first the register\_printer sample, as it uses a TCP connection only. If you get a list of registers and their actual values without any error, your basic network configuration is correct, if you don't get depth/image data please verify your multicast settings.

Note for windows users:

When you execute the demo apps for the first time and the windows firewall is activated, a security prompt will be displayed. Please make sure that you select the "Allow access" option for private networks. You could also change your firewall configuration in "Control Panel -> System and Security -> Firewall->Allowed Programs".

### 1.3.2 Samples

We have included three different samples with this package:

- Register\_printer: This application reads out all the general registers of the camera and prints their contents. This helps you to check how the camera is configured.
- Glut-3d: OpenGL viewer that displays 3d point cloud images obtained by the M100 camera using the XYZ+I format. The intensity values are dynamically mapped to a grey-scale ramp bounded by the maximal intensity.
- Glut-distances: A simple OpenGL viewer that plots depth data received from the camera to a regular grid.

The Glut applications depend on the freeglut utilities library. Please read the installation section to know how to get it.

To get more information about how the samples work please refer to sample files in this document.

### 1.3.3 Linux

For linux we include a CMakeList.txt file which will let you compile the samples easily. (cmake must be installed) Create a build directory (i.e inside the sample folder) and run cmake:



```
cd samples
mkdir build
cd build
cmake ..
make
```

After compiling you will find the executables under the build/ directory.

### 1.3.4 Windows

In windows we have already included executable files for the application samples. Please use the include .bat files to execute them or make sure you have the library dependencies installed in your system.

The API and some samples use pthread-w32 and freeglut as mentioned before. You will find in windows/libraries/ the binary library files used when this m100 API was built (pthreadVC2.dll and freeglut.dll). Copy the correct version for your system in c:/windows/system32. Remember also to install the m100.dll file you find in windows/bin/ folder to the system32/ directory.

If you want to build the binaries from the sources open in windows/samples\_vs/ the Visual Studio project (Visual Studio must be installed in your system). The project is ready to find the .lib files from its location. It is configured to generate Debug and Release version for 32 and 64 bits. Check the Configuration Manager to select the configuration you need before build. After building if you want to execute or debug the sample applications, you must install the .dll dependency files in your system as described above.



# Chapter 2

## Module Index

### 2.1 Modules

Here is a list of all modules:

Flags macros . . . . .	11
Registers macros . . . . .	12
Image data format macros . . . . .	13



## Chapter 3

# Data Structure Index

### 3.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">T_SENTIS_CONFIG</a>	
Configuration structure . . . . .	15
<a href="#">T_SENTIS_DATA_HEADER</a>	
The structure contains all members of the frame header . . . . .	16



## Chapter 4

# File Index

### 4.1 File List

Here is a list of all documented files with brief descriptions:

<a href="#">include/apitypes.h</a> . . . . .	19
<a href="#">include/m100api.h</a> . . . . .	21
<a href="#">glut-3d/3d.cpp</a> . . . . .	24
<a href="#">glut-distances/distances.cpp</a> . . . . .	27
<a href="#">register-printer/register.cpp</a> . . . . .	29





## Chapter 5

# Module Documentation

### 5.1 Flags macros

API Flags.

#### Macros

- `#define HOLD_CONTROL_ALIVE 0x00000001`
- `#define SENTIS_NON_BLOCKING_INPUT 0x00000001`

#### 5.1.1 Detailed Description

API Flags. The following macros define the values of the flags that can be use in the API functions

#### 5.1.2 Macro Definition Documentation

##### 5.1.2.1 `#define HOLD_CONTROL_ALIVE 0x00000001`

Indicate that the control connection will be opened until the handle is closed.

##### 5.1.2.2 `#define SENTIS_NON_BLOCKING_INPUT 0x00000001`

Indicate that the STSgetData function should return without waiting if the frame queue is empty.

## 5.2 Registers macros

General Register Definitions.

### Macros

- #define **Mode0** 0x0001
- #define **Status** 0x0003
- #define **ImageDataFormat** 0x0004
- #define **IntegrationTime** 0x0005
- #define **DeviceType** 0x0006
- #define **DeviceInfo** 0x0007
- #define **FirmwareInfo** 0x0008
- #define **ModulationFrequency** 0x0009
- #define **FrameRate** 0x000A
- #define **HardwareConfiguration** 0x000B
- #define **SerialNumberLowWord** 0x000C
- #define **SerialNumberHighWord** 0x000D
- #define **FrameCounter** 0x000E
- #define **ConfidenceThresLow** 0x0010
- #define **ConfidenceThresHig** 0x0011
- #define **Mode1** 0x0019
- #define **CalculationTime** 0x001A
- #define **LedboardsTemp** 0x001B
- #define **MainboardTemp** 0x001C
- #define **LinearizationAmplitude** 0x001D
- #define **LinearizationPhaseShift** 0x001E
- #define **MaxLedTemp** 0x0024
- #define **HorizontalFov** 0x0026
- #define **VerticalFov** 0x0027
- #define **TriggerDelay** 0x002B
- #define **BootloaderStatus** 0x002C
- #define **TemperatureCompensationGradient** 0x002D
- #define **ApplicationVersion** 0x002E
- #define **DistCalibGradient** 0x002F
- #define **DistCalibOffset** 0x0030
- #define **CmdExec** 0x0033
- #define **CmdExecResult** 0x0034
- #define **FactoryMacAddr2** 0x0035
- #define **FactoryMacAddr1** 0x0036
- #define **FactoryMacAddr0** 0x0037
- #define **FactoryYear** 0x0038
- #define **FactoryMonthDay** 0x0039
- #define **FactoryHourMinute** 0x003A
- #define **FactoryTimezone** 0x003B

### 5.2.1 Detailed Description

General Register Definitions.

## 5.3 Image data format macros

Image data format configurations.

### Macros

- #define **DEPTH\_AMP\_DATA** 0x0000
- #define **XYZ\_COORDS\_DATA** 0x0010
- #define **XYZ\_AMP\_DATA** 0x0020
- #define **PHASES\_0\_270\_DATA** 0x0038
- #define **PHASES\_270\_0\_DATA** 0x0040

### 5.3.1 Detailed Description

Image data format configurations.



## Chapter 6

# Data Structure Documentation

### 6.1 T\_SENTIS\_CONFIG Struct Reference

Configuration structure.

```
#include <apitypes.h>
```

#### Data Fields

- unsigned short [udp\\_port](#)
- unsigned short [tcp\\_port](#)
- unsigned [flags](#)
- char const \* [tcp\\_ip](#)
- char const \* [udp\\_ip](#)

#### 6.1.1 Detailed Description

Configuration structure.

This structure is used for configuring the API and for setting the connection parameters. It is used in the STSOpen function

#### 6.1.2 Field Documentation

##### 6.1.2.1 unsigned T\_SENTIS\_CONFIG::flags

Bit level set unsigned for flags. Used for forcing a specific behavior of the connection.

##### 6.1.2.2 char const\* T\_SENTIS\_CONFIG::tcp\_ip

Pointer to the string with the tcp ip address in network pointed format.

##### 6.1.2.3 unsigned short T\_SENTIS\_CONFIG::tcp\_port

The tcp port to use.

#### 6.1.2.4 char const\* T\_SENTIS\_CONFIG::udp\_ip

Pointer to the string with the udp multicast group ip address in network pointed format.

#### 6.1.2.5 unsigned short T\_SENTIS\_CONFIG::udp\_port

The udp port to use.

The documentation for this struct was generated from the following file:

- [include/apitypes.h](#)

## 6.2 T\_SENTIS\_DATA\_HEADER Struct Reference

The structure contains all members of the frame header.

```
#include <apitypes.h>
```

### Data Fields

- unsigned short [reserved1](#)
- unsigned short [headerVersion](#)
- unsigned short [imageWidth](#)
- unsigned short [imageHeight](#)
- unsigned char [nofChannels](#)
- unsigned char [bytesPerPixel](#)
- unsigned short [imageFormat](#)
- unsigned [timestamp](#)
- unsigned short [frameCounter](#)
- char [reserved2](#) [8]
- unsigned char [mainTemp](#)
- unsigned char [ledTemp](#)
- unsigned short [firmwareVersion](#)
- char [reserved3](#) [30]
- unsigned short [headerCrc16](#)

### 6.2.1 Detailed Description

The structure contains all members of the frame header.

It will be used for receiving and reading the configuration of the image frame within STSgetData.

### 6.2.2 Field Documentation

#### 6.2.2.1 unsigned char T\_SENTIS\_DATA\_HEADER::bytesPerPixel

Number of bytes per pixel.

#### 6.2.2.2 unsigned short T\_SENTIS\_DATA\_HEADER::firmwareVersion

Version of the firmware loaded in the camera.

6.2.2.3 unsigned short T\_SENTIS\_DATA\_HEADER::frameCounter

Frame counter

6.2.2.4 unsigned short T\_SENTIS\_DATA\_HEADER::headerCrc16

Crc16 value of the header

6.2.2.5 unsigned short T\_SENTIS\_DATA\_HEADER::headerVersion

Reserved short.

6.2.2.6 unsigned short T\_SENTIS\_DATA\_HEADER::imageFormat

Format of the image data. Value of the imageFormat register.

6.2.2.7 unsigned short T\_SENTIS\_DATA\_HEADER::imageHeight

Height of the image.

6.2.2.8 unsigned short T\_SENTIS\_DATA\_HEADER::imageWidth

Width of the image

6.2.2.9 unsigned char T\_SENTIS\_DATA\_HEADER::ledTemp

Temperature of the led lamp.

6.2.2.10 unsigned char T\_SENTIS\_DATA\_HEADER::mainTemp

Temperature of the main board.

6.2.2.11 unsigned char T\_SENTIS\_DATA\_HEADER::nofChannels

Number of channels.

6.2.2.12 unsigned short T\_SENTIS\_DATA\_HEADER::reserved1

Reserved 2 bytes.

6.2.2.13 char T\_SENTIS\_DATA\_HEADER::reserved2[8]

Reserved 8 bytes

6.2.2.14 char T\_SENTIS\_DATA\_HEADER::reserved3[30]

Reserved 30 bytes

#### 6.2.2.15 unsigned T\_SENTIS\_DATA\_HEADER::timestamp

Unix timestamp

The documentation for this struct was generated from the following file:

- [include/apitypes.h](#)

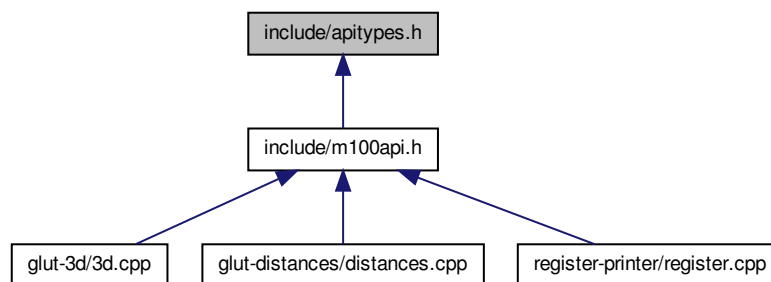


## Chapter 7

# File Documentation

### 7.1 include/apitypes.h File Reference

This graph shows which files directly or indirectly include this file:



#### Data Structures

- struct [T\\_SENTIS\\_CONFIG](#)  
*Configuration structure.*
- struct [T\\_SENTIS\\_DATA\\_HEADER](#)  
*The structure contains all members of the frame header.*

#### Macros

- #define [HOLD\\_CONTROL\\_ALIVE](#) 0x00000001
- #define [SENTIS\\_NON\\_BLOCKING\\_INPUT](#) 0x00000001
- #define **Mode0** 0x0001
- #define **Status** 0x0003
- #define **ImageDataFormat** 0x0004
- #define **IntegrationTime** 0x0005
- #define **DeviceType** 0x0006
- #define **DeviceInfo** 0x0007
- #define **FirmwareInfo** 0x0008
- #define **ModulationFrequency** 0x0009

- `#define FrameRate 0x000A`
- `#define HardwareConfiguration 0x000B`
- `#define SerialNumberLowWord 0x000C`
- `#define SerialNumberHighWord 0x000D`
- `#define FrameCounter 0x000E`
- `#define ConfidenceThresLow 0x0010`
- `#define ConfidenceThresHig 0x0011`
- `#define Mode1 0x0019`
- `#define CalculationTime 0x001A`
- `#define LedboardsTemp 0x001B`
- `#define MainboardTemp 0x001C`
- `#define LinearizationAmplitude 0x001D`
- `#define LinearizationPhasseShift 0x001E`
- `#define MaxLedTemp 0x0024`
- `#define HorizontalFov 0x0026`
- `#define VerticalFov 0x0027`
- `#define TriggerDelay 0x002B`
- `#define BootloaderStatus 0x002C`
- `#define TemperatureCompensationGradient 0x002D`
- `#define ApplicationVersion 0x002E`
- `#define DistCalibGradient 0x002F`
- `#define DistCalibOffset 0x0030`
- `#define CmdExec 0x0033`
- `#define CmdExecResult 0x0034`
- `#define FactoryMacAddr2 0x0035`
- `#define FactoryMacAddr1 0x0036`
- `#define FactoryMacAddr0 0x0037`
- `#define FactoryYear 0x0038`
- `#define FactoryMonthDay 0x0039`
- `#define FactoryHourMinute 0x003A`
- `#define FactoryTimezone 0x003B`
- `#define DEPTH_AMP_DATA 0x0000`
- `#define XYZ_COORDS_DATA 0x0010`
- `#define XYZ_AMP_DATA 0x0020`
- `#define PHASES_0_270_DATA 0x0038`
- `#define PHASES_270_0_DATA 0x0040`

## Typedefs

- `typedef void * T\_SENTIS\_HANDLE`  
*Device handle.*
- `typedef int T\_ERROR\_CODE`  
*Error code.*

### 7.1.1 Detailed Description

#### Author

VoXel Interaction Design [office@voxel.at](mailto:office@voxel.at)

#### Version

1.0.0

### 7.1.2 DESCRIPTION

Sentis-ToF-m100 camera API structure and type definitions

### 7.1.3 Typedef Documentation

#### 7.1.3.1 typedef int T\_ERROR\_CODE

Error code.

Type of the response returned when calling API functions

#### 7.1.3.2 typedef void\* T\_SENTIS\_HANDLE

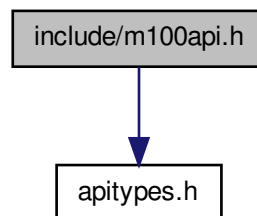
Device handle.

This type contains the necessary data to communicate with the Sentis-ToF-m100 camera after opening a connection. these connection data are created with the STSOpen function and they should be correctly closed with the STSClose function to avoid any memory problems

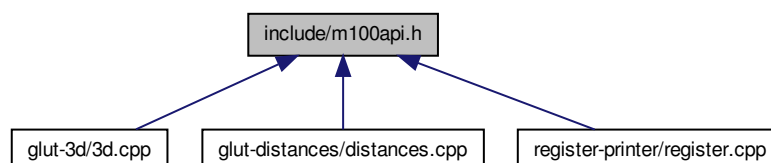
## 7.2 include/m100api.h File Reference

```
#include <apitypes.h>
```

Include dependency graph for m100api.h:



This graph shows which files directly or indirectly include this file:



## Functions

- DLLExport [T\\_SENTIS\\_HANDLE](#) STSopen ( [T\\_SENTIS\\_CONFIG](#) \*pa\_tConf, [T\\_ERROR\\_CODE](#) \*pa\_tErr)  
*Open the UDP and TCP sockets for data and control interfaces. Instance a thread which receives the UDP data and writes them to a queue.*
- DLLExport [T\\_ERROR\\_CODE](#) STSclose ( [T\\_SENTIS\\_HANDLE](#) pa\_tHndl)  
*Deletes the device handle and frees all allocated memory and close any TCP and UDP connections.*
- DLLExport [T\\_ERROR\\_CODE](#) STSgetData ( [T\\_SENTIS\\_HANDLE](#) pa\_tHndl, [T\\_SENTIS\\_DATA\\_HEADER](#) \*pa\_tHeader, char \*pa\_pcBuffer, int \*pa\_nLength, double pa\_nFlags, int pa\_nTimeout)  
*The function is used to read the image data of one frame.*
- DLLExport [T\\_ERROR\\_CODE](#) STSreadRegister ( [T\\_SENTIS\\_HANDLE](#) pa\_tHndl, unsigned short pa\_usAddress, unsigned short \*pa\_usData, int pa\_nFlags, int pa\_nTimeout)  
*Read the content of one register from the camera.*
- DLLExport [T\\_ERROR\\_CODE](#) STSwriteRegister ( [T\\_SENTIS\\_HANDLE](#) pa\_tHndl, unsigned short pa\_usAddress, unsigned short pa\_usData)  
*Write a value to one register of the camera.*

### 7.2.1 Detailed Description

#### Author

VoXel Interaction Design [office@voxel.at](mailto:office@voxel.at)

#### Version

1.0.0

### 7.2.2 DESCRIPTION

Sentis-ToF-m100 camera API

### 7.2.3 Function Documentation

#### 7.2.3.1 DLLExport [T\\_ERROR\\_CODE](#) STSclose ( [T\\_SENTIS\\_HANDLE](#) pa\_tHndl )

Deletes the device handle and frees all allocated memory and close any TCP and UDP connections.

#### Parameters

in	<a href="#">pa_tHndl</a> ,:	Device handle.
----	-----------------------------	----------------

#### Returns

0 if ok, negative error code otherwise.

#### 7.2.3.2 DLLExport [T\\_ERROR\\_CODE](#) STSgetData ( [T\\_SENTIS\\_HANDLE](#) pa\_tHndl, [T\\_SENTIS\\_DATA\\_HEADER](#) \*pa\_tHeader, char \*pa\_pcBuffer, int \*pa\_nLength, double pa\_nFlags, int pa\_nTimeout )

The function is used to read the image data of one frame.

The meta-information of the image is contained in pa\_tHeader. On each call one frame should be provided. The buffer contains the image data only depending on the selected image format. The buffer length must be indicated in the pa\_nLength parameter and it should be big enough for the set image format in the camera register. If the length

of the buffer is too small, the function will return an error and `pa_nLength` will contain the size needed for the image format set in the camera register.

#### Note

[T\\_SENTIS\\_DATA\\_HEADER](#): The structure should contain all members of the frame header as described in the Sentis-ToF-M100 manual.

#### Parameters

in	<i>pa_tHndl</i>	Device handle.
out	<i>pa_pcBuffer</i>	Buffer to the image data.
in, out	<i>pa_nLength</i>	It should provide the size of <code>pa_pcBuffer</code> . On return " <code>pa_nLength</code> " should contain the real size of the image data. If not enough buffer space has been provided an error code should be reported.
out	<i>pa_tHeader</i>	Pointer to a image header structure.
in	<i>pa_nFlags</i>	Flags to control the behavior of the function.
in	<i>pa_nTimeout</i>	No frame timeout [us] in case of blocking behavior. 0 means default timeout (1000ms).

#### Returns

0 if ok, negative error code otherwise.

#### 7.2.3.3 DLLExport T\_SENTIS\_HANDLE STSopen ( T\_SENTIS\_CONFIG \* pa\_tConf, T\_ERROR\_CODE \* pa\_tErr )

Open the UDP and TCP sockets for data and control interfaces. Instance a thread which receives the UDP data and writes them to a queue.

This function starts the communication with the camera, checks the data connection and keeps it opened if the `HOLD_CONTROL_ALIVE` flag is set. It creates the udp data connection and initializes a queue where the image frames are saved.

To configure these connections and theirs behavior, the function receives the [T\\_SENTIS\\_CONFIG](#) struct where the user should define the connection values.

The user receives an handle `T_SENTIS_HANDLE` which allows to interact with the camera.

The user should check if the connection is established by checking the `T_ERROR_CODE`. Also it is the responsibility of the user to correctly close and delete the connection and the data structures related by using the function `STSclose`.

#### Parameters

in	<i>pa_tConf</i>	Configuration structure.
out	<i>pa_tErr</i>	0 on success, a negative one byte error code otherwise

#### Returns

Device handle

#### 7.2.3.4 DLLExport T\_ERROR\_CODE STSreadRegister ( T\_SENTIS\_HANDLE pa\_tHndl, unsigned short pa\_usAddress, unsigned short \* pa\_usData, int pa\_nFlags, int pa\_nTimeout )

Read the content of one register from the camera.

**Parameters**

in	<i>pa_tHndl</i>	Device handle.
in	<i>pa_usAddress</i>	16 bits register address.
out	<i>pa_usData</i>	16 bits register content buffer.
in	<i>pa_nFlags</i>	Read behavior. Blocking.
in	<i>pa_nTimeout</i>	Read timeout [us]. 0 means default timeout (1000ms).

**Returns**

0 if ok, negative error code otherwise.

### 7.2.3.5 `DLLExport T_ERROR_CODE STSwriteRegister ( T_SENTIS_HANDLE pa_tHndl, unsigned short pa_usAddress, unsigned short pa_usData )`

Write a value to one register of the camera.

**Note**

Any value could be written to the register. Please be sure you are giving the right values to avoid unexpected behaviors

**Parameters**

in	<i>pa_tHndl</i>	Device handle.
in	<i>pa_usAddress</i>	16 bits register address.
out	<i>pa_usData</i>	16 bits register value to be written.

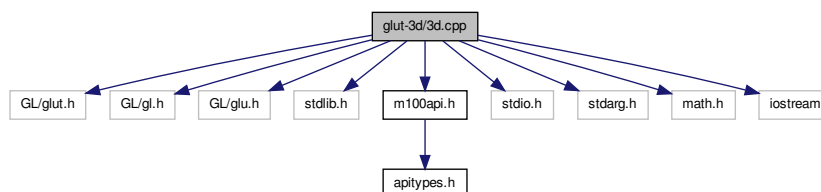
**Returns**

0 if ok, negative error code otherwise.

## 7.3 glut-3d/3d.cpp File Reference

```
#include <GL/glut.h>
#include <GL/gl.h>
#include <GL/glu.h>
#include <stdlib.h>
#include <m100api.h>
#include <stdio.h>
#include <stdarg.h>
#include <math.h>
#include <iostream>
```

Include dependency graph for 3d.cpp:



## Macros

- #define `TICKS_PER_SECOND` 20
- #define `LEN` 8192
- #define `VSNPRINTF` vsnprintf

## Functions

- int `initCamera` (`T_SENTIS_HANDLE` \*handle)  
*Helper method that starts the connection with the camera and set the default image\_data format.*
- int `getFrame` (`T_SENTIS_HANDLE` handle)  
*Get a fame from the camera and save it in the global buffer variable.*
- int `stopCamera` (`T_SENTIS_HANDLE` handle)  
*Close the connections with the camera and set free data structures.*
- void `init` ()  
*Called before main loop to set up the program.*
- void `reshape` (int w, int h)  
*Called every time a window is resized to resize the projection matrix.*
- void `camera` (void)
- void `printv` (va\_list args, const char \*format)  
*Print axis names.*
- void `print` (const char \*format,...)  
*Loop for printing axis names.*
- void `drawAxes` ()  
*draw axes at defined position*
- void `display` (int value)
- void `firstDisplay` ()  
*Called at the start of the program.*
- void `mouseMovement` (int x, int y)  
*Called after mouse clicking. It calculates the movement of the mouse and redisplay the scene.*
- void `mouse` (int button, int state, int x, int y)  
*Callback for mouse clicks.*
- void `keyboard` (unsigned char key, int x, int y)  
*Callback for normal key pressed.*
- void `windowSpecial` (int key, int x, int y)  
*Callback for arrow keys.*
- int `main` (int argc, char \*\*argv)

## Variables

- const int `TIMER_MILLISECONDS` = 100 / `TICKS_PER_SECOND`
- float `xpos` = 0
- float `ypos` = 0
- float `zpos` = 0
- float `xrot` = 0
- float `yrot` = 0
- float `angle` =0.0
- float `lastx`
- float `lasty`
- double `dim` =300.0
- int `windowWidth` =512
- int `windowHeight` =512

- int **moving** = 0
- int **toggleAxes** = 1
- [T\\_SENTIS\\_HANDLE](#) **handler**
- [T\\_ERROR\\_CODE](#) **error**

### 7.3.1 Detailed Description

#### Version

1.0.0

### 7.3.2 Macro Definition Documentation

#### 7.3.2.1 #define TICKS\_PER\_SECOND 20

### 7.3.3 DESCRIPTION

Example application for the sentis-ToF-m100 camera API

This application shows an OpenGL view which displays 3D images based on 3D coordinates. It uses the XYZ+I data format. The camera is supposed to be at 1,5m from the ground. The windows shows 300cm\*300cm\*300cm. The Y and Z axis show +150cm. The X axis points 250cm.

The amplitude values are used to set the color (gray) intensity of the image.

### 7.3.4 Function Documentation

#### 7.3.4.1 void drawAxes ( )

draw axes at defined position

We have defined the height and width of 300cm. the y and z axis show the 150cm positive. The camera should be at 1,5m over the ground in order to get a centered image. The depth of the X axis is set to 250cm.

#### 7.3.4.2 int getFrame ( [T\\_SENTIS\\_HANDLE](#) *handle* )

Get a frame from the camera and save it in the global buffer variable.

#### Parameters

in	<a href="#">T_SENTIS_HANDLE</a>	Camera handler.
----	---------------------------------	-----------------

#### 7.3.4.3 int initCamera ( [T\\_SENTIS\\_HANDLE](#) \* *handle* )

Helper method that starts the connection with the camera and set the default image\_data format.

#### Parameters

in, out	<a href="#">T_SENTIS_HANDLE</a>	* Camera handler.
---------	---------------------------------	-------------------



## 7.3.4.4 int stopCamera ( T\_SENTIS\_HANDLE handle )

Close the connections with the camera and set free data structures.

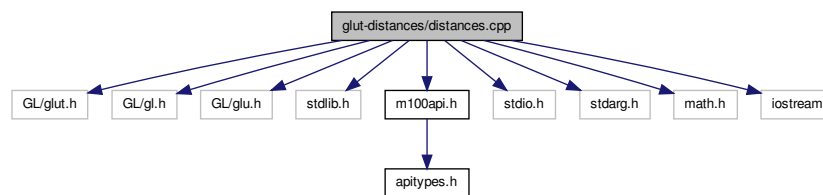
## Parameters

in	T_SENTIS_HANDLE	Camera handler.
----	-----------------	-----------------

## 7.4 glut-distances/distances.cpp File Reference

```
#include <GL/glut.h>
#include <GL/gl.h>
#include <GL/glu.h>
#include <stdlib.h>
#include <m100api.h>
#include <stdio.h>
#include <stdarg.h>
#include <math.h>
#include <iostream>
```

Include dependency graph for distances.cpp:



## Macros

- #define **TICKS\_PER\_SECOND** 20
- #define **LEN** 8192
- #define **VSNPRINTF** vsnprintf

## Functions

- int **initCamera** (T\_SENTIS\_HANDLE \*handle)  
*Helper method that starts the connection with the camera and set default image\_data format.*
- int **getFrame** (T\_SENTIS\_HANDLE handle)  
*Get a frame from the camera and save it in the global buffer variable.*
- int **stopCamera** (T\_SENTIS\_HANDLE handle)  
*Close the connections with the camera and set free data structures.*
- void **init** ()  
*Called before main loop to set up the program.*
- void **reshape** (int w, int h)  
*Called every time a window is resized to resize the projection matrix.*
- void **camera** (void)
- void **printv** (va\_list args, const char \*format)

- *Print axis names.*
- void `print` (const char \*format,...)
  - Loop for print axis names.*
- void `drawAxes` ()
  - draw axes at defined position*
- void `display` (int value)
  - Called at the start of the program, after a `glutPostRedisplay()` and during idle to display a frame.*
- void `firstDisplay` ()
  - Called at the start of the program.*
- void `mouseMovement` (int x, int y)
  - Called after mouse click. It calculates the movement of the mouse and redisplay the scene.*
- void `mouse` (int button, int state, int x, int y)
  - Callback for mouse clicks.*
- void `keyboard` (unsigned char key, int x, int y)
  - Callback for normal key pressed.*
- void `windowSpecial` (int key, int x, int y)
  - Callback for arrow keys.*
- int `main` (int argc, char \*\*argv)

## Variables

- const int `TIMER_MILLISECONDS` = 100 / `TICKS_PER_SECOND`
- float `xpos` = 0
- float `ypos` = 0
- float `zpos` = 0
- float `xrot` = 0
- float `yrot` = 0
- float `angle` = 0.0
- float `lastx`
- float `lasty`
- double `dim` = 300.0
- int `windowWidth` = 512
- int `windowHeight` = 512
- int `moving` = 0
- int `toggleAxes` = 1
- `T_SENTIS_HANDLE` handler
- `T_ERROR_CODE` error

### 7.4.1 Detailed Description

#### Version

1.0.0

### 7.4.2 Macro Definition Documentation

#### 7.4.2.1 `#define TICKS_PER_SECOND 20`

### 7.4.3 DESCRIPTION

Example application for the sentis-ToF-m100 camera API

This application shows an OpenGL view which displays 3D images based in distance values. It uses the distances + amplitudes data format. The view displays a grid of 160cmX120cm and the X axis shows the depth ~300cm.

The amplitude values are used to set the color (gray) intensity of the image.

### 7.4.4 Function Documentation

#### 7.4.4.1 void drawAxes ( )

draw axes at defined position

We have a grid of 160cm\*120cm. The depth of the X axis is set to 250cm

#### 7.4.4.2 int getFrame ( T\_SENTIS\_HANDLE *handle* )

Get a frame from the camera and save it in the global buffer variable.

##### Parameters

in	T_SENTIS_HANDLE	Camera handler.
----	-----------------	-----------------

#### 7.4.4.3 int initCamera ( T\_SENTIS\_HANDLE \* *handle* )

Helper method that starts the connection with the camera and set default image\_data format.

##### Parameters

in, out	T_SENTIS_HANDLE	* Camera handler.
---------	-----------------	-------------------

#### 7.4.4.4 int stopCamera ( T\_SENTIS\_HANDLE *handle* )

Close the connections with the camera and set free data structures.

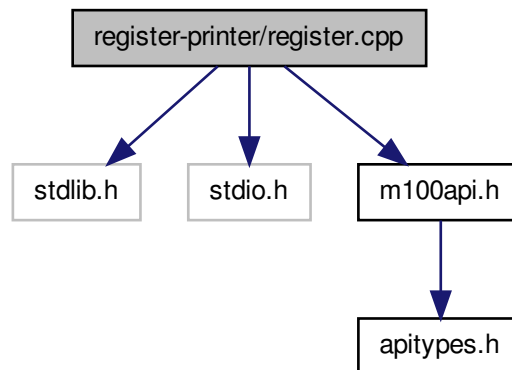
##### Parameters

in	T_SENTIS_HANDLE	Camera handler.
----	-----------------	-----------------

## 7.5 register-printer/register.cpp File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <m100api.h>
```

Include dependency graph for register.cpp:



## Functions

- int `main` (int argc, char \*argv[])

### 7.5.1 Detailed Description

#### Version

1.0.0

### 7.5.2 Function Documentation

#### 7.5.2.1 int main ( int argc, char \* argv[] )

### 7.5.3 DESCRIPTION

Example application for the sentis-ToF-m100 camera API

This application shows how to connect to the camera with the default network values and how to read the registers.

It reads all the status registers of the camera and prints the values in the console.

# Index

3d.cpp  
  drawAxes, [26](#)  
  getFrame, [26](#)  
  initCamera, [26](#)  
  stopCamera, [26](#)  
  TICKS\_PER\_SECOND, [26](#)

apitypes.h  
  T\_ERROR\_CODE, [21](#)  
  T\_SENTIS\_HANDLE, [21](#)

bytesPerPixel  
  T\_SENTIS\_DATA\_HEADER, [16](#)

distances.cpp  
  drawAxes, [29](#)  
  getFrame, [29](#)  
  initCamera, [29](#)  
  stopCamera, [29](#)  
  TICKS\_PER\_SECOND, [28](#)

drawAxes  
  3d.cpp, [26](#)  
  distances.cpp, [29](#)

firmwareVersion  
  T\_SENTIS\_DATA\_HEADER, [16](#)

flags  
  T\_SENTIS\_CONFIG, [15](#)

Flags macros, [11](#)  
  HOLD\_CONTROL\_ALIVE, [11](#)

frameCounter  
  T\_SENTIS\_DATA\_HEADER, [16](#)

getFrame  
  3d.cpp, [26](#)  
  distances.cpp, [29](#)

glut-3d/3d.cpp, [24](#)

glut-distances/distances.cpp, [27](#)

HOLD\_CONTROL\_ALIVE  
  Flags macros, [11](#)

headerCrc16  
  T\_SENTIS\_DATA\_HEADER, [17](#)

headerVersion  
  T\_SENTIS\_DATA\_HEADER, [17](#)

Image data format macros, [13](#)

imageFormat  
  T\_SENTIS\_DATA\_HEADER, [17](#)

imageHeight  
  T\_SENTIS\_DATA\_HEADER, [17](#)

imageWidth  
  T\_SENTIS\_DATA\_HEADER, [17](#)

include/apitypes.h, [19](#)

include/m100api.h, [21](#)

initCamera  
  3d.cpp, [26](#)  
  distances.cpp, [29](#)

ledTemp  
  T\_SENTIS\_DATA\_HEADER, [17](#)

m100api.h  
  STSclose, [22](#)  
  STSgetData, [22](#)  
  STSoopen, [23](#)  
  STSreadRegister, [23](#)  
  STSwriteRegister, [24](#)

main  
  register.cpp, [30](#)

mainTemp  
  T\_SENTIS\_DATA\_HEADER, [17](#)

nofChannels  
  T\_SENTIS\_DATA\_HEADER, [17](#)

register-printer/register.cpp, [29](#)

register.cpp  
  main, [30](#)

Registers macros, [12](#)

reserved1  
  T\_SENTIS\_DATA\_HEADER, [17](#)

reserved2  
  T\_SENTIS\_DATA\_HEADER, [17](#)

reserved3  
  T\_SENTIS\_DATA\_HEADER, [17](#)

STSclose  
  m100api.h, [22](#)

STSgetData  
  m100api.h, [22](#)

STSoopen  
  m100api.h, [23](#)

STSreadRegister  
  m100api.h, [23](#)

STSwriteRegister  
  m100api.h, [24](#)

stopCamera  
  3d.cpp, [26](#)  
  distances.cpp, [29](#)

T\_ERROR\_CODE

- apitypes.h, [21](#)
- T\_SENTIS\_CONFIG, [15](#)
  - flags, [15](#)
  - tcp\_ip, [15](#)
  - tcp\_port, [15](#)
  - udp\_ip, [15](#)
  - udp\_port, [16](#)
- T\_SENTIS\_HANDLE
  - apitypes.h, [21](#)
- TICKS\_PER\_SECOND
  - 3d.cpp, [26](#)
  - distances.cpp, [28](#)
- tcp\_ip
  - T\_SENTIS\_CONFIG, [15](#)
- tcp\_port
  - T\_SENTIS\_CONFIG, [15](#)
- timestamp
  - T\_SENTIS\_DATA\_HEADER, [17](#)
- udp\_ip
  - T\_SENTIS\_CONFIG, [15](#)
- udp\_port
  - T\_SENTIS\_CONFIG, [16](#)