# Ethernet API - Read Me

*Devices:* *iSYS-5020, iSYS-5021, iSYS-5110*

*Revision:* *4*

*Date:* *2018-11-22*

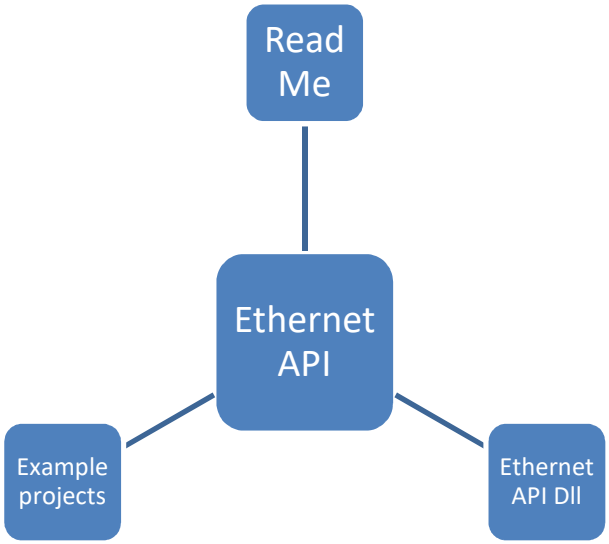
# Table of content

# History

| Document revision | Date | Change log | Author |
|---|---|---|---|
| 1 | 2017-02-15 | Initial Release | SW |
| 2 | 2017-03-28 | Adapted programming example chapter | PG |
| 3 | 2018-05-08 | Added iSYS-5020 | ClB / JW |
| 4 | 2018-11-22 | Added iSYS-5021 | JK |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

## 1. What is the Ethernet API?

The Ethernet API includes the following components:




- Read me
  *Description of the following components "How to use the Ethernet API?"*
- Ethernet API DLL
  *DLL and header file for easy access to system functionality*
- *Example projects for an easy start*

## 2. General Information

- The Ethernet API was built on MS Windows 7 32-bit system with *Microsoft Visual C++ Compiler 10.0 (x86) and MINGW 4.8*
- The Ethernet API was tested with MS Windows 7 32-bit
- The Ethernet API is incompatible with MS Windows XP, Vista and older MS Windows versions
- To guaranty 8 Byte alignment of defined structures the union type is used
- The Ethernet API is designed for synchronal use only (blocking application)
- Do not modify delivered Ethernet API files
- Use the Ethernet API specific data types defined in ethernet_API_basicTypes.h (see 5)

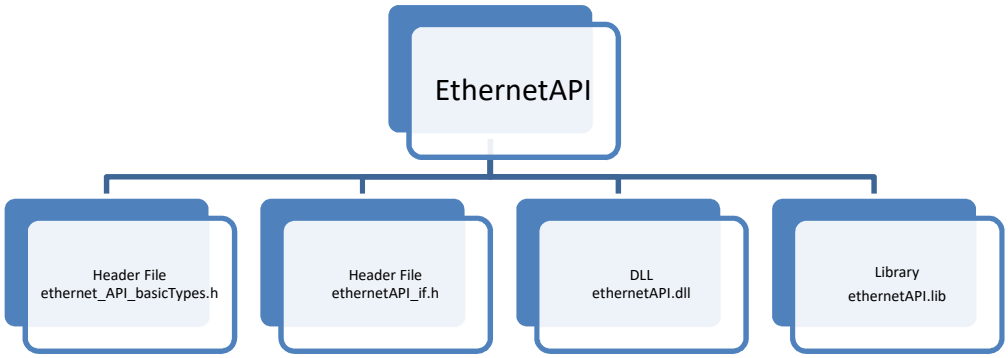
The Ethernet API is designed for usage with iSYS-5020, iSYS-5021 and iSYS-5110.

The documentation of Ethernet API functions is structured in:

- common functions

## 3. Get the Ethernet API running in your own application

The Ethernet API consists of the following components:




- ethernet_API_basicTypes.h
  *Contains all basic type definitions used by the Ethernet API*

  *(copy this into your project folder)*
- ethernetAPI_if.h
  *Contains all necessary definitions, type definitions, structures and commands featured by the Ethernet API*

  *(copy this into your project folder)*
- ethernetAPI.dll
  *Library file which contains all functions referred in the header file*

  *(copy this into the folder where your executable is located)*
- ethernetAPI.lib

  *Links against *.dll functions*

  *(copy this into your project folder)*

Each callable function (from Ethernet API) has a return value that inherits information about the status of the function. This return value is of type *ETHResult_t.* If *ETHResult_t* is ETH_ERR_OK the command was successfully completed. Otherwise an error occurred during operation.
Please have a look at the delivered example projects to see how you can use the Ethernet API.

## 4. Programming Example with Ethernet API

In the delivered software package are following examples included.

- *ethernetAPI_iSYS-5110_example_VC10_x86*

  Example project how to get a target list in combination with the ethernet API in Visual Studio 2010 (C++).

- *ethernetAPI_iSYS-5020_example_VC10_x86*

  Example project how to get a target list in combination with the ethernet API in Visual Studio 2010 (C++).

# 5. Used types in Ethernet API

In the Ethernet API specific data types are used. You will find them in the *ethernet_API_basicTypes.h* and ethernetAPI_if.*h* file.

## *5.1. Default data types*

In *ethernet_API_basicTypes.h* are data types defined for the Ethernet API. For full compatibility use these data types as well.

| Data Type | Definition |
|---|---|
| `bool_t` | 1 bit boolean |
| `float32_t` | 32 bit floating point |
| `sint8_t / uint8_t` | 8 bit signed / unsigned integer |
| `sint16_t / uint16_t` | 16 bit signed / unsigned integer |
| `sint32_t / uint32_t` | 32 bit signed / unsigned integer |

## *5.2. Used structures and enumerators*

There are structures and enumerators used in Ethernet API. They are defined in ethernetAPI_if.h and make function call easier for the user. Please use these types as well.

| Note: | To avoid alignment errors some enumerators are also defined as union and the union type is used. |
|---|---|

### 5.2.1. `struct ETHTargetList_t`

The *ETHTargetList_t* inherits all information about a target list calculated by the running system.

```
typedef struct ETHTargetList{
    ETHTargetListError_u ETHTargetListError;
    ETHTarget_t targetList[ETH_MAX_TARGETS];
    uint16_t nrOfTargets;
    uint16_t frameID;
}ETHTargetList_t;
```

- *ETHTargetListError_u ETHTargetListError*

Inherits information about the list status. (see 5.2.3)

- *ETHTarget_t targets[ETH_MAX_TARGETS]*

Inherits the target list (see 5.2.2). Maximum Targets are typical 256.

- *uint16_t nrOfTargets*

Information about the number of detected targets.

- *uint16_t frameID*

Each target list is identified by a frameID and will be incremented each measurement cycle. This parameter can be used to detect lost frames.

### 5.2.2. struct ETHTarget_t

The ETHTarget_t struct contains the parameters for one target.

```
typedef struct ETHTarget{
    float32_t signalStrength;
    float32_t range;
    float32_t velocity;
    float32_t angleAzimuth;
    float32_t reserved1;
    float32_t reserved2;
}ETHTarget_t;
```

- *float32_t signalStrength*

Signal indicator in dB.

- *float32_t range*

Range in m.

- *float32_t velocity*

Radial velocity in m/s.

- *float32_t angleAzimuth*

Angle of the target in °.

- *float32_t reserved1*

Not used

- *float32_t reserved2*

Not used

### 5.2.3. union ETHTargetListError_u

The ETHTargetListError_u inherits the error status for the target list.

```
union ETHTargetListError_u
{
    ETHTargetListError_t ETHTargetListError;
    uint32_t dummy;
};
```

- *ETHTargetListError_t ETHTargetListError*

Error status code for the target list (see 5.2.5)

- *uint32_t dummy*

Dummy value. Not used.

### 5.2.4. enum ETHResult_t

Enum for Function Results

| ETHResult_t | Description |
|---|---|
| ETH_ERR_OK | No error. Function was executed successful |
| ETH_ERR_HANDLE_NOT_INITIALISED | System is not connected. Unknown Handle ID. Please use ETH_InitSystem Function |
| ETH_ERR_SYSTEM_ALREADY_INITIALISED | System is already connected |
| ETH_ERR_SYSTEM_NOT_INITIALISED | System is not initialized |
| ETH_ERR_CREATE_HANDLE | Failed to create new handle ID |
| ERR_NULL_POINTER | Pointer does not refer to a valid object. |
| ERR_FUNCTION_DEPRECATED | Function deprecated, do not use this function anymore |
| ETH_ERR_PORT_ALREADY_INITIALISED | UDP Port initialized with another system |
| ETH_ERR_PORT_IN_USE | UDP Port in use with another system |
| ETH_ERR_CONNECTION_CLOSED | Connection closed |
| ETH_ERR_CONNECTION_RESET | Connection reset |
| ETH_ERR_COMMUNICATION_TIMEOUT | Connection timeout |
| ETH_ERR_COMMUNICATION_ERROR | Communication error |
| ETH_ERR_CONNECTION_LOST | Connection list |
| ETH_ERR_TARGET_NOT_ENOUGH_DATA_AVAILABLE | Not enough data for a complete target list |
| ETH_ERR_TARGET_DATA_CORRUPTED | Target list data is corrupted |
| ETH_ERR_TARGET_DATA_SIZE | Target list size does not match |
| ETH_ERR_RAW_NOT_ENOUGH_DATA_AVAILABLE | Not used |
| ETH_ERR_RAW_DATA_CORRUPTED | Not used |
| ETH_ERR_RAW_DATA_SIZE | Not used |
| ETH_ERR_MUTEX_ERROR | Mutex error in Thread |

### 5.2.5. enum ETHTargetListError_t

*Enum for ETH Target list errors.*

| ETHTargetListError_t | Description |
|---|---|
| ETH_TARGET_LIST_OK | No error. Target list is valid |
| ETH_TARGET_LIST_FULL | Target list full |
| ETH_TARGET_LIST_REFRESHED | Target list refreshed |
| ETH_TARGET_LIST_ALREADY_REQUESTED | Target list already requested |
| ETH_TARGET_LIST_NOT_ACTIVE | Acquisition not started, could not read target list |
| ETH_TARGET_LIST_DATA_CORRUPTED | Target list data corrupted |

# 6. API Function Description

This chapter describes all available functions including calling examples. It is structured into two sections. There are functions that could be used for all systems and functions that are only valid for special systems.

## *6.1. Common Functions*

### 6.1.1. ETH_initSystem (…)

To open the UDP connection to a connected iSYS-5x10 use ETH_initSystem (…).

**Input:**

```
ETHHandle_t *pHandle, uint8_t ipPart1, uint8_t ipPart2, uint8_t
ipPart3, uint8_t ipPart4, uint32_t udpPortNumber
```

For each connection to a system, the user has to create an empty ETHHandle_t pHandle that is a unique identifier to communicate with this unique system. This function will initiate the connection and initialize pHandle.

### 6.1.2. ETH_exitSystem (…)

To close (disconnect) an existing connection use ETH_exitSystem(…).

**Input:**

```
ETHHandle_t pHandle
```

### 6.1.3. ETH_getApiVersion (…)

To get the Ethernet API version use ETH_getApiVersion(…).

**Input:**

```
float32_t *version
```

### 6.1.4. ETH_getTargetList (…)

To get the target list use ETH_getTargetList (…).

**Input:**

```
iSYSHandle_t pHandle, ETHTargetList_t *pTargetList
```

**InnoSenT GmbH**
Am Roedertor 30
97499 Donnersdorf
GERMANY

Tel.: +49 95289518-0
E-Mail: info@innosent.de
www.innosent.de