

**TMG Technologie  
Management Gruppe  
Technologie und Engineering GmbH**

Zur Gießerei 10  
D - 76227 Karlsruhe

Telefon +49 (0)721-82806 0  
Telefax +49 (0)721-82806 10  
E-Mail [info@tmgte.de](mailto:info@tmgte.de)  
Internet [www.tmgte.de](http://www.tmgte.de)

# **IOLink Master 2.0**

## **UDP DLL Interface Description**

Version 2.12      August 2014

**Geschäftsführer**  
Dipl.-Ing. Klaus-Peter Willems  
Dipl.-Inform. Dirk Brauner

**Bankverbindung**  
Baden-Württembergische  
Bank AG – Karlsruhe  
BLZ: 600 501 01  
Konto: 7495503160

**Register / ID**  
HRB-Nr. 10 4483 Mannheim  
USt.-ID DE 143 605 231

## Index

<b>1. MODULE DOCUMENTATION .....</b>	<b>5</b>
1.1 GLOBAL DEFINITIONS .....	5
1.1.1 Parameter LONG in all function interfaces .....	5
1.1.2 Sensor Status Bit definitions .....	5
1.1.3 Return codes which are used in the library functions. ....	5
1.1.4 Detailed Description.....	6
1.1.5 Macro Definition Documentation .....	6
1.2 INTERFACE MANAGEMENT.....	8
1.2.1 Data Structures.....	8
1.2.2 Functions .....	8
1.2.3 Detailed Description.....	8
1.2.4 Function Documentation .....	8
1.3 PORT CONFIGURATION.....	11
1.3.1 Data Structures.....	11
1.3.2 Functions .....	11
1.3.3 PortModus port modi which are used for TargetMode in SetPortConfig .....	11
1.3.4 Port Commands. Commands which are used to switch the actual state of a port via the function SM_SetCommand. ....	11
1.3.5 Port Mode details for SIO output mode. ....	11
1.3.6 Port Mode details for SIO input mode. ....	11
1.3.7 Validation Mode, used in SetPortConfig.....	12
1.3.8 Commands which are used in the DSConfigure parameter in SetPortConfig. ....	12
1.3.9 Baud rates. Speed of the connection if it's established.....	12
1.3.10 SensorStateDefinitions for TInfo .....	12
1.3.11 Detailed Description.....	12
1.3.12 Macro Definition Documentation .....	12
1.3.13 Function Documentation .....	15
1.4 PROCESS DATA HANDLING .....	19
1.4.1 Functions .....	19
1.4.2 Detailed Description.....	19
1.4.3 Function Documentation .....	19
1.5 ISDU HANDLING .....	22
1.5.1 Data Structures.....	22
1.5.2 Functions .....	22
1.5.3 Detailed Description.....	22

1.5.4	Function Documentation .....	22
1.6	EVENT HANDLING .....	23
1.6.1	Data Structures .....	23
1.6.2	Functions .....	23
1.6.3	Event definitions .....	23
1.6.4	Detailed Description.....	24
1.6.5	Macro Definition Documentation .....	24
1.6.6	Function Documentation .....	27
1.7	DATA STORAGE .....	28
1.7.1	Functions .....	28
1.7.2	Commands which are used in IOL_DS_Command. ....	28
1.7.3	Detailed Description.....	28
1.7.4	Macro Definition Documentation .....	28
1.7.5	Function Documentation .....	29
<b>2.</b>	<b>DATA STRUCTURE DOCUMENTATION.....</b>	<b>30</b>
2.1	TDEVICEIDENTIFICATION STRUCT REFERENCE .....	30
2.1.1	Data Fields .....	30
2.1.2	Detailed Description.....	31
2.1.3	Field Documentation .....	31
2.2	TDLLINFO STRUCT REFERENCE.....	31
2.2.1	Data Fields .....	31
2.2.2	Detailed Description.....	31
2.2.3	Field Documentation .....	31
2.3	TEVENT STRUCT REFERENCE .....	32
2.3.1	Data Fields .....	32
2.3.2	Detailed Description.....	32
2.3.3	Field Documentation .....	32
2.4	TINFO STRUCT REFERENCE .....	33
2.4.1	Data Fields .....	33
2.4.2	Detailed Description.....	33
2.4.3	Field Documentation .....	33
2.5	TINFOEX STRUCT REFERENCE .....	34
2.5.1	Data Fields .....	34
2.5.2	Detailed Description.....	34
2.5.3	Field Documentation .....	34
2.6	TMASTERINFO STRUCT REFERENCE.....	35

2.6.1	Data Fields .....	35
2.6.2	Detailed Description .....	35
2.6.3	Field Documentation .....	35
2.7	TPARAMETER STRUCT REFERENCE .....	36
2.7.1	Data Fields .....	36
2.7.2	Detailed Description .....	36
2.7.3	Field Documentation .....	36
2.8	TPORTCONFIGURATION STRUCT REFERENCE .....	38
2.8.1	Data Fields .....	38
2.8.2	Detailed Description .....	38
2.8.3	Field Documentation .....	38
<b>3.</b>	<b>INDEX .....</b>	<b>40</b>

# 1. Module Documentation

## 1.1 Global Definitions

### 1.1.1 Parameter LONG in all function interfaces

If a Parameter is defined as LONG in the API, the size of the parameter depends on the used DLL. If the DLL which is used is the 32-bit DLL, the size of the variable is 32 bit. If the 64bit-DLL is used, the size of the parameter is 64-bit.

### 1.1.2 Sensor Status Bit definitions

Some of the functions return a sensor status, which contains some status bits. The following definitions define the different informations which are shown by the status

- #define [MASK\\_SENSORSTATE](#) ((BYTE) (0x13))
- #define [BIT\\_CONNECTED](#) ((BYTE) (0x01))
- #define [BIT\\_PREOPERATE](#) ((BYTE) (0x02))
- #define [BIT\\_WRONGSENSOR](#) ((BYTE) (0x10))
- #define [BIT\\_EVENTAVAILABLE](#) ((BYTE) (0x04))
- #define [BIT\\_PDVALID](#) ((BYTE) (0x08))
- #define [BIT\\_SENSORSTATEKNOWN](#) ((BYTE) (0x80))

### 1.1.3 Return codes which are used in the library functions.

These return codes define the reaction of the library functions. This doesn't include the error codes which are returned by the IO-Link devices during ISDU access Codes less than zero are reported from the DLL. The commands have not been transmitted to the IO-Link master if these codes occur. Codes from 1 to 100 are reported from the IO-Link master. They occur if a service which has been received from the DLL cannot be executed due to some reason. all other codes are coming from the IO-Link device as defined in the standard

- #define [RETURN\\_FIRMWARE\\_NOT\\_COMPATIBLE](#) -16
- #define [RETURN\\_FUNCTION\\_NOT\\_IMPLEMENTED](#) -13
- #define [RETURN\\_STATE\\_CONFLICT](#) -12
- #define [RETURN\\_WRONG\\_COMMAND](#) -11
- #define [RETURN\\_WRONG\\_PARAMETER](#) -10
- #define [RETURN\\_WRONG\\_DEVICE](#) -9
- #define [RETURN\\_NO\\_EVENT](#) -8
- #define [RETURN\\_UNKNOWN\\_HANDLE](#) -7
- #define [RETURN\\_UART\\_TIMEOUT](#) -6
- #define [RETURN\\_CONNECTION\\_LOST](#) -5
- #define [RETURN\\_OUT\\_OF\\_MEMORY](#) -4
- #define [RETURN\\_DEVICE\\_ERROR](#) -3
- #define [RETURN\\_DEVICE\\_NOT\\_AVAILABLE](#) -2
- #define [RETURN\\_INTERNAL\\_ERROR](#) -1
- #define [RETURN\\_OK](#) 0
- #define [RESULT\\_STATE\\_CONFLICT](#) 1
- #define [RESULT\\_NOT\\_SUPPORTED](#) 2
- #define [RESULT\\_SERVICE\\_PENDING](#) 3
- #define [RESULT\\_WRONG\\_PARAMETER\\_STACK](#) 4

## 1.1.4 Detailed Description

These common definitions are used for several functions in the interface.

---

## 1.1.5 Macro Definition Documentation

### **#define MASK\_SENSORSTATE ((BYTE) (0x13))**

1= Sensor Found, 0 = Sensor Lost, 2 = Sensor in Preoperate, 0x10 = wrong sensor connected, validation failed

### **#define BIT\_CONNECTED ((BYTE) (0x01))**

0x01 Sensor is connected and in state OPERATE

### **#define BIT\_PREOPERATE ((BYTE) (0x02))**

0x02 Sensor is connected and in state PREOPERATE

### **#define BIT\_WRONGSENSOR ((BYTE) (0x10))**

0x03 Sensor is connected, but the validation failed, and a WRONG\_SENSOR event has been received

### **#define BIT\_EVENTAVAILABLE ((BYTE) (0x04))**

1 means that there are Events to be read, 0 if there is no event

### **#define BIT\_PDVALID ((BYTE) (0x08))**

1 means Process datas are valid, 0 if not

### **#define BIT\_SENSORSTATEKNOWN ((BYTE) (0x80))**

1 means State of Sensor is known, 0 if not. (at start of set mode)

### **#define RETURN\_FIRMWARE\_NOT\_COMPATIBLE -16**

the firmware needs a firmware update because some of the functions are not implemented

### **#define RETURN\_FUNCTION\_NOT\_IMPLEMENTED -13**

the function is not implemented in the connected IO-Link Master

### **#define RETURN\_STATE\_CONFLICT -12**

the function cannot be used in the actual state of the IO-Link Master

**#define RETURN\_WRONG\_COMMAND -11**

a wrong answer to a command has been received from the IO-Link Master

**#define RETURN\_WRONG\_PARAMETER -10**

one of the function parameters is invalid

**#define RETURN\_WRONG\_DEVICE -9**

the device name was wrong or the device which is connected is not supported

**#define RETURN\_NO\_EVENT -8**

a Read Event was called, but there is no event

**#define RETURN\_UNKNOWN\_HANDLE -7**

the handle of the function is unknown

**#define RETURN\_UART\_TIMEOUT -6**

a timeout has been reached because there as no answer to a command

**#define RETURN\_CONNECTION\_LOST -5**

the master has been unplugged during communication

**#define RETURN\_OUT\_OF\_MEMORY -4**

no more memory available

**#define RETURN\_DEVICE\_ERROR -3**

error in accessing the UDP driver

**#define RETURN\_DEVICE\_NOT\_AVAILABLE -2**

the device is not available at this moment

**#define RETURN\_INTERNAL\_ERROR -1**

internal library error. Please restart the program

**#define RETURN\_OK 0**

sucessful end of the function

**#define RESULT\_STATE\_CONFLICT 1**

the command is not applicable in the actual state

## **#define RESULT\_NOT\_SUPPORTED 2**

the command is not supported on this device

## **#define RESULT\_SERVICE\_PENDING 3**

a Service is pending. A new service must wait for the end of the pending service

## **#define RESULT\_WRONG\_PARAMETER\_STACK 4**

a parameter has been rejected by the IO-Link master

## **1.2 interface management**

### **1.2.1 Data Structures**

- struct [TDeviceIdentification](#)
- struct [TMasterInfo](#)
- struct [TDllInfo](#)

### **1.2.2 Functions**

- LONG \_\_stdcall [IOL\\_Create](#) (char \*Device)
- LONG \_\_stdcall [IOL\\_Destroy](#) (LONG Handle)
- LONG \_\_stdcall [IOL\\_GetUDPDevices](#) ([TDeviceIdentification](#) \*pDeviceList, LONG MaxNumberOfEntries)
- LONG \_\_stdcall [IOL\\_GetMasterInfo](#) (LONG Handle, [TMasterInfo](#) \*pMasterInfo)
- LONG \_\_stdcall [IOL\\_GetDLLInfo](#) ([TDllInfo](#) \*pDllInfo)

---

### **1.2.3 Detailed Description**

These functions are used to manage the access to IO-Link master. There are function to list all connected devices, and to connect or disconnect to a special device.

---

### **1.2.4 Function Documentation**

#### **LONG \_\_stdcall IOL\_Create (char \* *Device*)**

Creates and initializes the communication port and handle.

This function opens the referred COM Port and initializes the internal Datastructures. If the return value is greater than 0 it is the Handle by which the connected Master and its structures are referenced. It shall be used with further calls to functions in this Library.

Parameters:

<i>Device</i>	IP adress of the IO-Link Master , e.g. "192.168.0.110"
---------------	--

Return values:



<i>RETURN_DEVICE_NOT_AVAILABLE</i>	the Device referred by the string parameter "Device" is not available or busy
<i>RETURN_COMM_TIMEOUT</i>	the device did not respond in time
<i>RETURN_OUT_OF_MEMORY</i>	no more Handles can be assigned
<i>RETURN_WRONG_PARAMETER</i>	the device name was wrong
<i>RETURN_FIRMWARE_NOT_COMPATIBLE</i>	the firmware needs a firmware update because some of the functions are not implemented

Returns:

if greater than 0 the returnvalue is a Handle

### **LONG \_\_stdcall IOL\_Destroy (LONG Handle)**

Closes the communication port and discards the Handle.

This function closes the COM Port referred by the Handle. And also frees all the Memory corresponding to the Handle.

Note:

This function has to be called, once the Programm using this DLL is about to terminate. Otherwise, when not unloading the DLL one might risk an OUT\_OF\_MEMORY error.

Parameters:

<i>Handle</i>	Handle to work on/with
---------------	------------------------

Return values:

<i>RETURN_UNKNOWN_HANDLE</i>	Handle is not valid
<i>RETURN_INTERNAL_ERROR</i>	Error that should not occur.
<i>RETURN_OK</i>	Everything worked out alright

### **LONG \_\_stdcall IOL\_GetUDPDevices ([TDeviceIdentification](#) \* pDeviceList, LONG MaxNumberOfEntries)**

Looks for IO-Link devices which are visible on the network.

This function looks for UDP IO-Link masters returns a list of these devices. The information which is achieved from the device manager contains name and product information of the device.

Note:

The memory containing the resulting list must be allocated by the application. The library cannot check if the size is big enough, therefore the application must ensure the size

The method of finding the master uses UDP Broadcasts and the answers from the device. The user has to care about the firewall settings and the routing table if there are more than one network adapter attached to the system.

Parameters:

<i>pDeviceList</i>	pointer to a buffer for the result
<i>MaxNumberOfEntries</i>	max number of entries which can be put in the buffer

Return values:

<i>number</i>	of IO-Link masters which are found
---------------	------------------------------------

**LONG \_\_stdcall IOL\_GetMasterInfo (LONG *Handle*, [TMasterInfo](#) \* *pMasterInfo*)**

Get information from the IO-Link Master.

This function gets version and type information from the IO-Link Master. The module type is contained in the Version string (Standard or Development Version).

Parameters:

<i>Handle</i>	Handle to work on/with
<i>pMasterInfo</i>	Pointer to TMasterinfo structure

Return values:

<i>RETURN_UNKNOWN_HANDLE</i>	Handle is not valid
<i>RETURN_INTERNAL_ERROR</i>	Error that should not occur.
<i>RETURN_OK</i>	Everything worked out alright

**LONG \_\_stdcall IOL\_GetDLLInfo ([TDllInfo](#) \* *pDllInfo*)**

Get information about the DLL.

This function returns the Version information from the DLL

Parameters:

<i>pDllInfo</i>	Pointer to <a href="#">TDllInfo</a> structure
-----------------	---

Return values:

<i>RETURN_OK</i>	Everything worked out alright
------------------	-------------------------------

## 1.3 Port Configuration

### 1.3.1 Data Structures

- struct [TPortConfiguration](#)
- struct [TInfo](#)
- struct [TInfoEx](#)

### 1.3.2 Functions

- LONG \_\_stdcall [IOL\\_SetPortConfig](#) (LONG Handle, DWORD Port, [TPortConfiguration](#) \*pConfig)
- LONG \_\_stdcall [IOL\\_GetMode](#) (LONG Handle, DWORD Port, [TInfo](#) \*pInfo)
- LONG \_\_stdcall [IOL\\_SetCommand](#) (LONG Handle, DWORD Port, DWORD Command)
- LONG \_\_stdcall [IOL\\_GetSensorStatus](#) (LONG Handle, DWORD Port, DWORD \*Status)
- LONG \_\_stdcall [IOL\\_GetModeEx](#) (LONG Handle, DWORD Port, [TInfoEx](#) \*pInfoEx, BOOL OnlyStatus)

### 1.3.3 PortModus port modi which are used for TargetMode in SetPortConfig

- #define [SM\\_MODE\\_RESET](#) 0
- #define [SM\\_MODE\\_IOLINK\\_PREOP](#) 1
- #define [SM\\_MODE\\_SIO\\_INPUT](#) 3
- #define [SM\\_MODE\\_SIO\\_OUTPUT](#) 4
- #define [SM\\_MODE\\_IOLINK\\_PREOP\\_FALLBACK](#) 10
- #define [SM\\_MODE\\_IOLINK\\_OPER\\_FALLBACK](#) 11
- #define [SM\\_MODE\\_IOLINK\\_OPERATE](#) 12
- #define [SM\\_MODE\\_IOLINK\\_FALLBACK](#) 13

### 1.3.4 Port Commands. Commands which are used to switch the actual state of a port via the function SM\_SetCommand.

Note that not all state changes are allowed at any time

- #define [SM\\_COMMAND\\_FALLBACK](#) 5
- #define [SM\\_COMMAND\\_PD\\_OUT\\_VALID](#) 6
- #define [SM\\_COMMAND\\_PD\\_OUT\\_INVALID](#) 7
- #define [SM\\_COMMAND\\_OPERATE](#) 8
- #define [SM\\_COMMAND\\_RESTART](#) 9

### 1.3.5 Port Mode details for SIO output mode.

These values define the mode of a digital output.

- #define [SM\\_MODE\\_SIO\\_PP\\_SWITCH](#) 0x0
- #define [SM\\_MODE\\_SIO\\_HS\\_SWITCH](#) 0x80
- #define [SM\\_MODE\\_SIO\\_LS\\_SWITCH](#) 0x40

### 1.3.6 Port Mode details for SIO input mode.

These values define the mode of a digital input.

- #define [SM\\_MODE\\_NORMAL\\_INPUT](#) 0
- #define [SM\\_MODE\\_DIAGNOSTIC\\_INPUT](#) 1
- #define [SM\\_MODE\\_INVERT\\_INPUT](#) 2

### 1.3.7 Validation Mode, used in SetPortConfig.

These values define the validation mode.

- #define [SM\\_VALIDATION\\_MODE\\_NONE](#) 0
- #define [SM\\_VALIDATION\\_MODE\\_COMPATIBLE](#) 1
- #define [SM\\_VALIDATION\\_MODE\\_IDENTICAL](#) 2

### 1.3.8 Commands which are used in the DSConfigure parameter in SetPortConfig.

These values define the behavior of the parameter server.

- #define [DS\\_CFG\\_ENABLED](#) 0x80
- #define [DS\\_CFG\\_UPLOAD\\_ENABLED](#) 0x01

### 1.3.9 Baud rates. Speed of the connection if it's established

- #define [SM\\_BAUD\\_19200](#) 0
- #define [SM\\_BAUD\\_38400](#) 1
- #define [SM\\_BAUD\\_230400](#) 2

### 1.3.10 SensorStateDefinitions for TInfo

The SensorState in [TInfo](#) structure is different from other state definitions. This is due to historical use of this function. it will still work, but the functions IOL\_GetSensorState and IOL\_GetModeEx have some advantages over the function IOL\_GetMode the value is only useful for the IO-Link mode. In other modes the state will show always the value STATE\_DISCONNECTED\_GETMODE

- #define [STATE\\_DISCONNECTED\\_GETMODE](#) 0
- #define [STATE\\_PREOPERATE\\_GETMODE](#) 0x80
- #define [STATE\\_WRONGSENSOR\\_GETMODE](#) 0x40
- #define [STATE\\_OPERATE\\_GETMODE](#) 0xFF

---

### 1.3.11 Detailed Description

These functions are used to set the specific mode of an IO-Link port, and to get information about connected sensors.

---

### 1.3.12 Macro Definition Documentation

**#define SM\_MODE\_RESET 0**

Port is deactivated

**#define SM\_MODE\_IOLINK\_PREOP 1**

Port is in IO-Link mode and stops in Preoperate

**#define SM\_MODE\_SIO\_INPUT 3**

Port is in SIO Input mode

**#define SM\_MODE\_SIO\_OUTPUT 4**

Port is in SIO Output mode

**#define SM\_MODE\_IOLINK\_PREOP\_FALLBACK 10**

io-link to preoperate, fallback allowed

**#define SM\_MODE\_IOLINK\_OPER\_FALLBACK 11**

io-link to operate, fallback allowed

**#define SM\_MODE\_IOLINK\_OPERATE 12**

Io-Link, but go into operate automatically

**#define SM\_MODE\_IOLINK\_FALLBACK 13**

io-link to preoperate, then automatically to fallback

**#define SM\_COMMAND\_FALLBACK 5**

switch Device from IO-Link mode back to SIO

**#define SM\_COMMAND\_PD\_OUT\_VALID 6**

send outputs\_valid to device

**#define SM\_COMMAND\_PD\_OUT\_INVALID 7**

send outputs\_invalid to device

**#define SM\_COMMAND\_OPERATE 8**

switch from preoperate to operate state

**#define SM\_COMMAND\_RESTART 9**

restart the connection

**#define SM\_MODE\_SIO\_PP\_SWITCH 0x0**

Digital output works in Push/Pull mode

**#define SM\_MODE\_SIO\_HS\_SWITCH 0x80**

Digital output works as High Side Switch

**#define SM\_MODE\_SIO\_LS\_SWITCH 0x40**

Digital output works as Low Side Switch

**#define SM\_MODE\_NORMAL\_INPUT 0**

Digital input works as a normal input

**#define SM\_MODE\_DIAGNOSTIC\_INPUT 1**

Digital input works as a diagnostic input

**#define SM\_MODE\_INVERT\_INPUT 2**

Digital input works as a inverted input

**#define SM\_VALIDATION\_MODE\_NONE 0**

no validation, each combination of device and vendor id is allowed

**#define SM\_VALIDATION\_MODE\_COMPATIBLE 1**

device and vendor ID will be checked

**#define SM\_VALIDATION\_MODE\_IDENTICAL 2**

device and vendor ID and the serial number will be checked

**#define DS\_CFG\_ENABLED 0x80**

the data storage is enabled

**#define DS\_CFG\_UPLOAD\_ENABLED 0x01**

the automatical upload is enabled

**#define SM\_BAUD\_19200 0**

speed of the connection is 19200 baud

**#define SM\_BAUD\_38400 1**

speed of the connection is 38400 baud

**#define SM\_BAUD\_230400 2**

speed of the connection is 230400 baud

```
#define STATE_DISCONNECTED_GETMODE 0
```

```
#define STATE_PREOPERATE_GETMODE 0x80
```

```
#define STATE_WRONGSENSOR_GETMODE 0x40
```

```
#define STATE_OPERATE_GETMODE 0xFF
```

---

### 1.3.13 Function Documentation

**LONG \_\_stdcall IOL\_SetPortConfig (LONG *Handle*, DWORD *Port*, [TPortConfiguration](#) \*  
*pConfig*)**

Sets the Mode according to the Parameters.

This function sets the Port on the IO-Link Master Gateway to the desired Mode, specified by the parameters of pConfig.

- TargetMode defines the mode of the port which is used. possible Values are:
  - SM\_MODE\_RESET Port is deactivated
  - SM\_MODE\_IOLINK\_PREOP Port is in IO-Link mode and stops in Preoperate
  - SM\_MODE\_SIO\_INPUT Port is in SIO Input mode
  - SM\_MODE\_SIO\_OUTPUT Port is in SIO Output mode
  - SM\_MODE\_IOLINK\_PREOP\_FALLBACK io-link to preoperate, fallback allowed
  - SM\_MODE\_IOLINK\_OPER\_FALLBACK io-link to operate, fallback allowed
  - SM\_MODE\_IOLINK\_OPERATE Io-Link, but go into operate automatically
  - SM\_MODE\_IOLINK\_FALLBACK io-link to preoperate, then automatically to fallback
- PortModeDetails sets additional information for the port mode. The content depends on the TargetMode:
  - in IO-Link Modes SM\_MODE\_IOLINK\_xxx the value contains the cycle time. The format of the cycle time is defined in the IO-Link specification. A value of 0 means "free running" mode where the maximum of (min cycle time of the device and min cycle time of the master) will be used as the real cycle time.
  - in SM\_MODE\_SIO\_INPUT the value defines the behavior of the input value. Possible values are:
    - SM\_MODE\_NORMAL\_INPUT Digital input works as a normal input
    - SM\_MODE\_DIAGNOSTIC\_INPUT Red if Open, diagnostic input
    - SM\_MODE\_INVERT\_INPUT Digital input works as a inverted input
  - in SM\_MODE\_SIO\_OUTPUT the value defines the physical mode of the output circuit
    - SM\_MODE\_SIO\_PP\_SWITCH Digital output works in Push/Pull mode
    - SM\_MODE\_SIO\_HS\_SWITCH Digital output works as High Side Switch
    - SM\_MODE\_SIO\_LS\_SWITCH Digital output works as Low Side Switch
- CRID defines the Configured revision ID. This Value defines the IO-Link version which will be used to communicate. If the sensor does not support this version, the connection will fail. Possible values are:

- 0x11 The Port will be used in V11 Mode. Devices based on Specification 1.0 will be accessed with V1.0 Frames. Devices based on V1.1 Spec will be accessed with V1.1 Frames.
- 0x10 The Port will run in V10 Mode. Devices based on V11 Specification will be automatically switched to V10 if they are capable to do this.
- DSConfigure configuration of the Data storage. The Values can be combined. Possible values are:
  - DS\_CFG\_ENABLED defines that the data storage is enabled.
  - DS\_CFG\_UPLOAD\_ENABLED defines that the automatically upload is enabled. If not set, the Upload must be done manually.
- InspectionLevel defines the amount of validation which is done at connecting to the device. If one of the validation parameters does not match the parameters in the device, the connection will fail
  - SM\_VALIDATION\_MODE\_NONE there is no validation. Each device can be connected without validating anything. The parameters VendorID, DeviceID and SerialNumber can be left empty
  - SM\_VALIDATION\_MODE\_COMPATIBLE defines the mode where a given device can be exchanged with a device of the same type. the Parameter VendorID and DeviceID must be set and are checked against the parameters of the device. If the device matches these values (this includes compatible devices which can switch the device ID), the connection will be successful. Otherwise it will fail.
  - SM\_VALIDATION\_MODE\_IDENTICAL defines the mode where the exact device will be checked which is connected. All Parameters VendorID, DeviceID and SerialNumber will be checked. Of course the device has to support the Parameter SerialNumber which is not mandatory.
- InputLength defines the input data length of the application. The normal value is 32 so that each device can be connected. If the application doesn't support 32 byte, it can reduce this. If the device needs more data, the connection will fail
- OutputLength defines the output data length of the application. The normal value is 32 so that each device can be connected. If the application doesn't support 32 byte, it can reduce this. If the device needs more data, the connection will fail

Parameters:

<i>Handle</i>	Handle to work on/with
<i>Port</i>	Port number of the used port
<i>pConfig</i>	pointer to the data structure containing the data

Return values:

<i>RETURN_UNK NOWN_HANDLE</i>	Handle is not valid
<i>RETURN_INTERNAL_ERROR</i>	Error that should not occur.
<i>RETURN_OK</i>	Everything worked out alright

**LONG \_\_stdcall IOL\_GetMode (LONG *Handle*, DWORD *Port*, [TInfo](#) \* *plnfo*)**

Gets the current Mode.

This function gets the current state and Mode information of the Port on the IO-Link Master. The result will be stored in the data structure pointed to by the parameter plnfo.

- COM contains the Device name of the IO-Link Master (such as "COM3")



- DeviceID contains the device ID of the connected device
- VendorID contains the Vendor ID of the connected device
- FunctionID contains the Function ID of the connected device
- ActualMode is the actual running mode of the port. the values are a subset of the values used by SetPortConfig:
  - SM\_MODE\_RESET Port is deactivated
  - SM\_MODE\_IOLINK\_PREOP Port is in IO-Link mode
  - SM\_MODE\_SIO\_INPUT Port is in SIO Input mode
  - SM\_MODE\_SIO\_OUTPUT Port is in SIO Output mode
- SensorState defines the actual state of the sensor:
- MasterCycle defines the actual cycle time which is used in the connection
- CurrentBaudrate defines the actual used baud rate of the connection

Parameters:

<i>Handle</i>	Handle to work on/with
<i>Port</i>	Port number of the used port
<i>pInfo</i>	Pointer to <a href="#">TInfo</a> structure

Return values:

<i>RETURN_UNK NOWN_HANDLE</i>	Handle is not valid
<i>RETURN_INTERRUPT_ERROR</i>	Error that should not occur.
<i>RETURN_OK</i>	Everything worked out alright

Warning:

This function is depreciated and should not be used anymore. Please use IOL\_GetStatus and IOL\_GetModeEx instead because they have advantages.

## **LONG \_\_stdcall IOL\_SetCommand (LONG Handle, DWORD Port, DWORD Command)**

Send Command to the IO-Link Master.

This function sends a command out of a predefined list of commands. These commands are transmitted to the sensor. Possible values for the Command are:

- SM\_COMMAND\_FALLBACK switch Device from IO-Link mode back to SIO
- SM\_COMMAND\_PD\_OUT\_VALID send outputs\_valid to device
- SM\_COMMAND\_PD\_OUT\_INVALID send outputs\_invalid to device
- SM\_COMMAND\_OPERATE switch from preoperate to operate state

Parameters:

<i>Handle</i>	Handle to work on/with
<i>Port</i>	Port number of the used port
<i>Command</i>	Pointer to TMasterinfo structure

Return values:

<i>RETURN_UNK NOWN_HANDLE</i>	Handle is not valid
-----------------------------------	---------------------

<i>LE</i>	
<i>RETURN_INTE RNAL_ERROR</i>	Error that should not occur.
<i>RETURN_OK</i>	Everything worked out alright

### **LONG \_\_stdcall IOL\_GetSensorStatus (LONG *Handle*, DWORD *Port*, DWORD \* *Status*)**

Return the current Sensor Status.

This function will return the current Sensorstatus. It will write the same bits to the Variable Status as it is written in Processdata Exchange.

Parameters:

<i>Handle</i>	Handle to work on/with
<i>Port</i>	Port number of the used port
<i>Status</i>	Status information (Events, Processdata Valid, ...)

Return values:

<i>RETURN_UNK NOWN_HAND LE</i>	Handle is not valid
<i>RETURN_INTE RNAL_ERROR</i>	Error that should not occur.
<i>RETURN_OK</i>	Everything worked out alright

### **LONG \_\_stdcall IOL\_GetModeEx (LONG *Handle*, DWORD *Port*, [TInfoEx](#) \* *pInfoEx*, BOOL *OnlyStatus*)**

Gets the current Mode.

This function gets the current state and Mode information of the Port on the IO-Link Master. The result will be stored in the data structure pointed to by the parameter pInfoEx.

- COM contains the Device name of the IO-Link Master (such as "COM3")
- DirectParameterPage contains the complete DPP1 of the device if OnlyStatus was false
- ActualMode is the actual running mode of the port. the values are a subset of the values used by SetPortConfig:
  - SM\_MODE\_RESET Port is deactivated
  - SM\_MODE\_IOLINK\_PREOP Port is in IO-Link mode
  - SM\_MODE\_SIO\_INPUT Port is in SIO Input mode
  - SM\_MODE\_SIO\_OUTPUT Port is in SIO Output mode
- SensorState defines the actual state of the sensor:
- CurrentBaudrate defines the actual used baud rate of the connection

Parameters:

<i>Handle</i>	Handle to work on/with
---------------	------------------------

<i>Port</i>	Port number of the used port
<i>pInfoEx</i>	Pointer to <a href="#">TInfo</a> structure
<i>OnlyStatus</i>	

Return values:

<i>RETURN_UNK NOWN_HAND LE</i>	Handle is not valid
<i>RETURN_INTE RNAL_ERROR</i>	Error that should not occur.
<i>RETURN_OK</i>	Everything worked out alright

## 1.4 Process Data Handling

### 1.4.1 Functions

- LONG \_\_stdcall [IOL\\_ReadOutputs](#) (LONG Handle, DWORD Port, BYTE \*ProcessData, DWORD \*Length, DWORD \*Status)
- LONG \_\_stdcall [IOL\\_ReadInputs](#) (LONG Handle, DWORD Port, BYTE \*ProcessData, DWORD \*Length, DWORD \*Status)
- LONG \_\_stdcall [IOL\\_WriteOutputs](#) (LONG Handle, DWORD Port, BYTE \*ProcessData, DWORD Length)
- LONG \_\_stdcall [IOL\\_TransferProcessData](#) (LONG Handle, DWORD Port, BYTE \*ProcessDataOut, DWORD LengthOut, BYTE \*ProcessDataIn, DWORD \*LengthIn, DWORD \*Status)

### 1.4.2 Detailed Description

These functions are used to get and set process data. In addition the data login can be activated and deactivated.

### 1.4.3 Function Documentation

**LONG \_\_stdcall IOL\_ReadOutputs (LONG *Handle*, DWORD *Port*, BYTE \* *ProcessData*, DWORD \* *Length*, DWORD \* *Status*)**

Read-back the Output Process Data written.

This function reads-back the Process Data written to the Process-Data- Output-Buffer previously with IOL\_WriteOutputs.

Parameters:

<i>Handle</i>	Handle to work on/with
---------------	------------------------

<i>Port</i>	Port from which to read, 0xFF = ALL Ports
<i>ProcessData</i>	Pointer to write the Process Data to
<i>Length</i>	Length of written Process Data
<i>Status</i>	Status information (Events, Processdata Valid, ...)

Return values:

<i>RETURN_UNK NOWN_HANDLE</i>	Handle is not valid
<i>RETURN_INTERRUPTAL_ERROR</i>	Error that should not occur.
<i>RETURN_OK</i>	Everything worked out alright

**LONG \_\_stdcall IOL\_ReadInputs (LONG *Handle*, DWORD *Port*, BYTE \* *ProcessData*, DWORD \* *Length*, DWORD \* *Status*)**

Read the Input Process Data from the Sensor connected.

This function reads the Process Data from the IO-Link Master, which was received from the Sensor. for specific port numbers, the structure contains the Length, the data, and a valid information for port 0xFF, which means ALL Ports, first byte is the number of entries. Then the above structure follows Length, data, valid

Parameters:

<i>Handle</i>	Handle to work on/with
<i>Port</i>	Port from which to read, 0xFF = ALL Ports
<i>ProcessData</i>	Pointer to write the Process Data to
<i>Length</i>	Length of written Process Data
<i>Status</i>	Status information (Events, Processdata Valid, ...)

Return values:

<i>RETURN_UNK NOWN_HANDLE</i>	Handle is not valid
<i>RETURN_INTERRUPTAL_ERROR</i>	Error that should not occur.
<i>RETURN_OK</i>	Everything worked out alright

**LONG \_\_stdcall IOL\_WriteOutputs (LONG *Handle*, DWORD *Port*, BYTE \* *ProcessData*, DWORD *Length*)**

Write Output Process Data to the IO-Link Master.

This function writes the Process Data referred by ProcessData to the IO-Link Master. The data is then transferred to the connected Sensor.

Parameters:

<i>Handle</i>	Handle to work on/with
<i>Port</i>	Port from which to read, 0xFF = ALL Ports
<i>ProcessData</i>	Pointer to the Process Data to be written
<i>Length</i>	Length of Process Data

Return values:

<i>RETURN_UNK NOWN_HANDLE</i>	Handle is not valid
<i>RETURN_INTERRUPTAL_ERROR</i>	Error that should not occur.
<i>RETURN_OK</i>	Everything worked out alright

**LONG \_\_stdcall IOL\_TransferProcessData (LONG *Handle*, DWORD *Port*, BYTE \* *ProcessDataOut*, DWORD *LengthOut*, BYTE \* *ProcessDataIn*, DWORD \* *LengthIn*, DWORD \* *Status*)**

Transfers Process Data in both directions.

This function transfers Process Data in both directions. It first sends out the Processdata referenced by ProcessDataOut. And then receives the response and writes it's content to ProcessDataIn.

Parameters:

<i>Handle</i>	Handle to work on/with
<i>Port</i>	Port from which to read and write, 0xFF = ALL Ports
<i>ProcessDataOut</i>	Pointer to read the Process Data from
<i>LengthOut</i>	Length of Process Data to be output
<i>ProcessDataIn</i>	Pointer to write the Process Data to
<i>LengthIn</i>	Length of written Process Data
<i>Status</i>	Status information (Events, Processdata Valid, ...)

Return values:

<i>RETURN_UNK NOWN_HANDLE</i>	Handle is not valid
<i>RETURN_INTERRUPTAL_ERROR</i>	Error that should not occur.

<i>RNAL_ERROR</i>	
<i>RETURN_OK</i>	Everything worked out alright

## 1.5 ISDU handling

### 1.5.1 Data Structures

- struct [TParameter](#)

### 1.5.2 Functions

- LONG \_\_stdcall [IOL\\_ReadReq](#) (LONG Handle, DWORD Port, [TParameter](#) \*pParameter)
- LONG \_\_stdcall [IOL\\_WriteReq](#) (LONG Handle, DWORD Port, [TParameter](#) \*pParameter)

---

### 1.5.3 Detailed Description

These functions are used to get and set parameter data via ISDU requests.

---

### 1.5.4 Function Documentation

**LONG \_\_stdcall IOL\_ReadReq (LONG *Handle*, DWORD *Port*, [TParameter](#) \* *pParameter*)**

Request read on SPDU from the Sensor.

This function sends a Read Request to the IO-Link Master, which passes it to the Device connected. The pParameter struct is used to set the Index and Subindex, that is requested via the SPDU-Channel.

Parameters:

<i>Handle</i>	Handle to work on/with
<i>Port</i>	Port number of the used port
<i>pParameter</i>	Pointer to <a href="#">TParameter</a> struct

Return values:

<i>RETURN_UNK NOWN_HAND LE</i>	Handle is not valid
<i>RETURN_INTE RNAL_ERROR</i>	Error that should not occur.
<i>RETURN_OK</i>	Everything worked out alright

**LONG \_\_stdcall IOL\_WriteReq (LONG *Handle*, DWORD *Port*, [TParameter](#) \* *pParameter*)**

Request to write on SPDU to the Sensor.

This function sends a Write Request to the IO-Link Master, which passes it to the Device connected. The pParameter struct is used to set the Index and Subindex, that is requested to be written via the SPDU-Channel. The pParameter struct also contains the Data that will be written.

Parameters:

<i>Handle</i>	Handle to work on/with
<i>Port</i>	Port number of the used port
<i>pParameter</i>	Pointer to <a href="#">TParameter</a> struct

Return values:

<i>RETURN_UNK NOWN_HAND LE</i>	Handle is not valid
<i>RETURN_INTE RNAL_ERROR</i>	Error that should not occur.
<i>RETURN_OK</i>	Everything worked out alright

## 1.6 Event handling

### 1.6.1 Data Structures

- struct [TEvent](#)

### 1.6.2 Functions

- LONG \_\_stdcall [IOL\\_ReadEvent](#) (LONG *Handle*, [TEvent](#) \**pEvent*, DWORD \**Status*)

### 1.6.3 Event definitions

These values define the content of the event buffer

- #define [EVNT\\_INST\\_UNKNOWN](#) 0
- #define [EVNT\\_INST\\_PHL](#) 1
- #define [EVNT\\_INST\\_DL](#) 2
- #define [EVNT\\_INST\\_AL](#) 3
- #define [EVNT\\_INST\\_APPL](#) 4
- #define [EVNT\\_TYPE\\_ERROR](#) 0x30
- #define [EVNT\\_TYPE\\_WARNING](#) 0x20
- #define [EVNT\\_TYPE\\_MESSAGE](#) 0x10
- #define [EVNT\\_MODE\\_SINGLE](#) 0x40
- #define [EVNT\\_MODE\\_COMING](#) 0xC0
- #define [EVNT\\_MODE\\_GOING](#) 0x80
- #define [EVNT\\_CODE\\_M\\_PDU\\_CHECK](#) 2
- #define [EVNT\\_CODE\\_S\\_DEVICELOST](#) 16

- #define [EVNT\\_CODE\\_S\\_WRONGSENSOR](#) 26
- #define [EVNT\\_CODE\\_S\\_RETRY](#) 27
- #define [EVNT\\_CODE\\_P\\_SHORT](#) 30
- #define [EVNT\\_CODE\\_P\\_SENSOR](#) 31
- #define [EVNT\\_CODE\\_P\\_ACTOR](#) 32
- #define [EVNT\\_CODE\\_P\\_POWER](#) 33
- #define [EVNT\\_CODE\\_P\\_RESET](#) 34
- #define [EVNT\\_CODE\\_S\\_FALLBACK](#) 35
- #define [EVNT\\_CODE\\_M\\_PREOPERATE](#) 36
- #define [EVNT\\_CODE\\_DSREADY\\_NOACTION](#) 40
- #define [DS\\_FAULT\\_IDENT](#) 41
- #define [DS\\_FAULT\\_SIZE](#) 42
- #define [DS\\_FAULT\\_UPLOAD](#) 43
- #define [DS\\_FAULT\\_DOWNLOAD](#) 44
- #define [DS\\_FAULT\\_DEVICE\\_LOCKED](#) 47
- #define [EVNT\\_CODE\\_DSREADY\\_DOWNLOAD](#) 50
- #define [EVNT\\_CODE\\_DSREADY\\_UPLOAD](#) 51
- #define [EVNT\\_CODE\\_S\\_WRONG\\_PDINLENGTH](#) 64
- #define [EVNT\\_CODE\\_S\\_WRONG\\_PDOUTLENGTH](#) 65
- #define [EVNT\\_CODE\\_S\\_WRONG\\_REVISION](#) 66
- #define [EVNT\\_CODE\\_S\\_WRONG\\_COMP\\_VENDORID](#) 67
- #define [EVNT\\_CODE\\_S\\_WRONG\\_COMP\\_DEVICEID](#) 68
- #define [EVNT\\_CODE\\_S\\_WRONG\\_COMP10\\_VENDORID](#) 69
- #define [EVNT\\_CODE\\_S\\_WRONG\\_COMP10\\_DEVICEID](#) 70
- #define [EVNT\\_CODE\\_S\\_WRONG\\_SERNUM](#) 71
- #define [EVNT\\_CODE\\_S\\_WRONG\\_CYCLE](#) 72

---

## 1.6.4 Detailed Description

These functions are used to handle the device events.

---

## 1.6.5 Macro Definition Documentation

**#define EVNT\_INST\_UNKNOWN 0**

instance is unknown

**#define EVNT\_INST\_PHL 1**

instance physical layer

**#define EVNT\_INST\_DL 2**

instance data layer

**#define EVNT\_INST\_AL 3**

instance Application Layer



**#define EVNT\_INST\_APPL 4**

instance Application

**#define EVNT\_TYPE\_ERROR 0x30**

event shows an error

**#define EVNT\_TYPE\_WARNING 0x20**

event shows a warning

**#define EVNT\_TYPE\_MESSAGE 0x10**

event shows a Message

**#define EVNT\_MODE\_SINGLE 0x40**

event shows a single message or warning

**#define EVNT\_MODE\_COMING 0xC0**

event shows that an error has appeared

**#define EVNT\_MODE\_GOING 0x80**

event shows that an error has disappeared

**#define EVNT\_CODE\_M\_PDU\_CHECK 2**

a frame with a CRC error has been received

**#define EVNT\_CODE\_S\_DEVICELOST 16**

Device has been disconnected: coming: line break going: device is in operate

**#define EVNT\_CODE\_S\_WRONGSENSOR 26**

a wrong sensor has been detected. Unspecific error. The normal case is code 64-72

**#define EVNT\_CODE\_S\_RETRY 27**

Retries have been detected

**#define EVNT\_CODE\_P\_SHORT 30**

a short circuit has been detected on the C/Q line

**#define EVNT\_CODE\_P\_SENSOR 31**

there is an error in the Sensor supply

**#define EVNT\_CODE\_P\_ACTOR 32**

there is an error in the Actor supply

**#define EVNT\_CODE\_P\_POWER 33**

there is an error in the Power Supply of the IO-Link master

**#define EVNT\_CODE\_P\_RESET 34**

an event is send if a port has been resetted

**#define EVNT\_CODE\_S\_FALLBACK 35**

fallback has been done successful, device is back in SIO state

**#define EVNT\_CODE\_M\_PREOPERATE 36**

device has reached the preoperate state

**#define EVNT\_CODE\_DSREADY\_NOACTION 40**

data storage come to the end, but there os no action, because the CRC was correct

**#define DS\_FAULT\_IDENT 41**

the sensor doesn't match the content in the data storage

**#define DS\_FAULT\_SIZE 42**

the sensor parameters doesn't fit in the memory of the data storage

**#define DS\_FAULT\_UPLOAD 43**

error in uploading the data storage

**#define DS\_FAULT\_DOWNLOAD 44**

error in downloading the data storage

**#define DS\_FAULT\_DEVICE\_LOCKED 47**

error in data storage function because the device is locked

**#define EVNT\_CODE\_DSREADY\_DOWNLOAD 50**

the parameter download has come to the end

**#define EVNT\_CODE\_DSREADY\_UPLOAD 51**

the parameter upload has come to the end

**#define EVNT\_CODE\_S\_WRONG\_PDINLENGTH 64**

process data input length don't match

**#define EVNT\_CODE\_S\_WRONG\_PDOUTLENGTH 65**

process data output length don't match

**#define EVNT\_CODE\_S\_WRONG\_REVISION 66**

device revision doesn't match

**#define EVNT\_CODE\_S\_WRONG\_COMP\_VENDORID 67**

vendor id is wrong V1.1 sensor

**#define EVNT\_CODE\_S\_WRONG\_COMP\_DEVICEID 68**

device id is wrong V1.1 sensor

**#define EVNT\_CODE\_S\_WRONG\_COMP10\_VENDORID 69**

vendor id is wrong V1.0 sensor

**#define EVNT\_CODE\_S\_WRONG\_COMP10\_DEVICEID 70**

device id is wrong V1.0 sensor

**#define EVNT\_CODE\_S\_WRONG\_SERNUM 71**

serial number is wrong

**#define EVNT\_CODE\_S\_WRONG\_CYCLE 72**

cycle time not matching

---

## 1.6.6 Function Documentation

**LONG \_\_stdcall IOL\_ReadEvent (LONG *Handle*, [TEvent](#) \* *pEvent*, DWORD \* *Status*)**

Get the last event out of the Event Buffer.

This function gets the most next Event out of the internal FIFO Buffer. The DLL stores occurring events in an internal FIFO Buffer with enough Space for 10 Events. If this function doesn't get called after 10 Events the last Event will be overridden.

Parameters:

<i>Handle</i>	Handle to work on/with
<i>pEvent</i>	Pointer to a <a href="#">TEvent</a> struct

<i>Status</i>	Status information (Events, Processdata Valid, ...)
---------------	---

Return values:

<i>RETURN_UNK NOWN_HAND LE</i>	Handle is not valid
<i>RETURN_INTE RNAL_ERROR</i>	Error that should not occur.
<i>RETURN_OK</i>	Everything worked out alright

## 1.7 Data Storage

### 1.7.1 Functions

- LONG \_\_stdcall [IOL\\_DS\\_Command](#) (LONG Handle, DWORD Port, DWORD DSCommand)
- LONG \_\_stdcall [IOL\\_DS\\_ContentGet](#) (LONG Handle, DWORD Port, BYTE \*pDSContentData, DWORD \*pDSContentLength)
- LONG \_\_stdcall [IOL\\_DS\\_ContentSet](#) (LONG Handle, DWORD Port, BYTE \*pDSContentData, DWORD DSContentLength)

### 1.7.2 Commands which are used in IOL\_DS\_Command.

These commands are used to activate data storage commands.

- #define [DS\\_CMD\\_UPLOAD](#) 0x01
- #define [DS\\_CMD\\_DOWNLOAD](#) 0x02
- #define [DS\\_CMD\\_CLEAR](#) 0x03

---

### 1.7.3 Detailed Description

These functions are used to handle the data storage commands.

---

### 1.7.4 Macro Definition Documentation

**#define DS\_CMD\_UPLOAD 0x01**

upload parameter-set

**#define DS\_CMD\_DOWNLOAD 0x02**

download current parameter-set

**#define DS\_CMD\_CLEAR 0x03**

clear stored parameter set

## 1.7.5 Function Documentation

### **LONG \_\_stdcall IOL\_DS\_Command (LONG *Handle*, DWORD *Port*, DWORD *DSCommand*)**

sends a data storage command

This function sends a data storage command to the data storage for a given port. The command is set in the parameter *DSCommand* and can contain the following values:

DS\_CMD\_UPLOAD starts an upload from the device DS\_CMD\_DOWNLOAD starts a download to the device DS\_CMD\_CLEAR clears the content of the data storage  
Parameters:

<i>Handle</i>	Handle to work on/with
<i>Port</i>	Port number of the used port
<i>DSCommand</i>	Command which shall be sent to the data storage

Return values:

<i>RETURN_UNKNOWN_HANDLE</i>	Handle is not valid
<i>RETURN_INTERNAL_ERROR</i>	Error that should not occur.
<i>RETURN_OK</i>	Everything worked out alright

### **LONG \_\_stdcall IOL\_DS\_ContentGet (LONG *Handle*, DWORD *Port*, BYTE \* *pDSContentData*, DWORD \* *pDSContentLength*)**

Reads out the content of the data storage.

This function reads the data storage buffer of the IO-Link master for a given port.  
Parameters:

<i>Handle</i>	Handle to work on/with
<i>Port</i>	Port number of the used port
<i>pDSContentData</i>	pointer to a buffer for the content of the data storage
<i>pDSContentLength</i>	pointer to the length. Must be initialized with the size of the buffer

Return values:

<i>RETURN_UNKN</i>	Handle is not valid
--------------------	---------------------

<i>NOWN_HANDLE</i>	
<i>RETURN_INTERNAL_ERROR</i>	Error that should not occur.
<i>RETURN_OK</i>	Everything worked out alright

**LONG \_\_stdcall IOL\_DS\_ContentSet (LONG *Handle*, DWORD *Port*, BYTE \* *pDSContentData*, DWORD *DSContentLength*)**

Writes the content of a data storage to the IO-Link master.

This function writes a buffer to the data storage of the IO-Link master.

Parameters:

<i>Handle</i>	Handle to work on/with
<i>Port</i>	Port number of the used port
<i>pDSContentData</i>	pointer to a buffer for the content of the data storage
<i>DSContentLength</i>	length of the buffer which shall be written

Return values:

<i>RETURN_UNKNOWN_HANDLE</i>	Handle is not valid
<i>RETURN_INTERNAL_ERROR</i>	Error that should not occur.
<i>RETURN_OK</i>	Everything worked out alright

---

## 2. Data Structure Documentation

### 2.1 TDeviceIdentification Struct Reference

#### 2.1.1 Data Fields

- char [Name](#) [20]
- char [NetworkName](#) [400]

## 2.1.2 Detailed Description

[TDeviceIdentification](#) contains the information about an IO-Link master.

---

## 2.1.3 Field Documentation

### **char TDeviceIdentification::Name[20]**

contains the device name (ip address) of the IO-Link master

### **char TDeviceIdentification::NetworkName[400]**

resolved DNS name if possible, IP adress otherwise

---

## 2.2 TDllInfo Struct Reference

### 2.2.1 Data Fields

- char [Build](#) [20]
  - char [Datum](#) [20]
  - char [Version](#) [20]
- 

### 2.2.2 Detailed Description

[TDllInfo](#) contains the DLL version information

---

### 2.2.3 Field Documentation

#### **char TDllInfo::Build[20]**

Build revision of the DLL

#### **char TDllInfo::Datum[20]**

build date of the DLL

#### **char TDllInfo::Version[20]**

major revision of the DLL

---

## 2.3 TEvent Struct Reference

### 2.3.1 Data Fields

- WORD [Number](#)
- WORD [Port](#)
- WORD [EventCode](#)
- BYTE [Instance](#)
- BYTE [Mode](#)
- BYTE [Type](#)
- BYTE [PDValid](#)
- BYTE [LocalGenerated](#)

---

### 2.3.2 Detailed Description

[TEvent](#) contains the data of an occurred event

---

### 2.3.3 Field Documentation

#### **WORD TEvent::Number**

number of the event, is incremented by the DLL

#### **WORD TEvent::Port**

port on which the event occurred

#### **WORD TEvent::EventCode**

event code

#### **BYTE TEvent::Instance**

instance of the event

#### **BYTE TEvent::Mode**

event mode

#### **BYTE TEvent::Type**

event type

#### **BYTE TEvent::PDValid**

event mode



## BYTE TEvent::LocalGenerated

TRUE if the event was generated by the IO-Link master

---

## 2.4 TInfo Struct Reference

### 2.4.1 Data Fields

- char [COM](#) [20]
- BYTE [DeviceID](#) [3]
- BYTE [VendorID](#) [2]
- BYTE [FunctionID](#) [2]
- BYTE [ActualMode](#)
- BYTE [SensorState](#)
- BYTE [MasterCycle](#)
- BYTE [CurrentBaudrate](#)

---

### 2.4.2 Detailed Description

[TInfo](#) contains the information about a connected sensor and the state of a port

---

### 2.4.3 Field Documentation

#### char TInfo::COM[20]

device interface name

#### BYTE TInfo::DeviceID[3]

Device ID

#### BYTE TInfo::VendorID[2]

Vendor ID

#### BYTE TInfo::FunctionID[2]

Function ID

#### BYTE TInfo::ActualMode

Actual Mode of the Port, Deactivated, IO-Link or SIO

### **BYTE TInfo::SensorState**

state of the sensor see

See Also:

SensorStateDefinitions

### **BYTE TInfo::MasterCycle**

used cycle time if sensor is connected

### **BYTE TInfo::CurrentBaudrate**

current baud rate

---

## **2.5 TInfoEx Struct Reference**

### **2.5.1 Data Fields**

- char [COM](#) [20]
- BYTE [DirectParameterPage](#) [16]
- BYTE [ActualMode](#)
- BYTE [SensorStatus](#)
- BYTE [CurrentBaudrate](#)

---

### **2.5.2 Detailed Description**

[TInfoEx](#) contains the extended information about a connected sensor

---

### **2.5.3 Field Documentation**

#### **char TInfoEx::COM[20]**

device interface name

#### **BYTE TInfoEx::DirectParameterPage[16]**

information from direct parameter page (Index 0)

#### **BYTE TInfoEx::ActualMode**

actual master port state

**BYTE TInfoEx::SensorStatus**

actual connection state of the sensor

**BYTE TInfoEx::CurrentBaudrate**

actual baud rate

---

## 2.6 TMasterInfo Struct Reference

### 2.6.1 Data Fields

- char [Version](#) [13]
- BYTE [Major](#)
- BYTE [Minor](#)
- BYTE [Build](#)
- BYTE [MajorRevisionIOLStack](#)
- BYTE [MinorRevisionIOLStack](#)
- BYTE [BuildRevisionIOLStack](#)

---

### 2.6.2 Detailed Description

[TMasterInfo](#) contains revision information from the connected master

---

### 2.6.3 Field Documentation

**char TMasterInfo::Version[13]**

string which was build from the following parameters

**BYTE TMasterInfo::Major**

major firmware revision

**BYTE TMasterInfo::Minor**

minor firmware revision

**BYTE TMasterInfo::Build**

build revision of the firmware

**BYTE TMasterInfo::MajorRevisionIOLStack**

major revision of the IO-Link stack used by the master

**BYTE TMasterInfo::MinorRevisionIOLStack**

minor revision of the IO-Link stack used by the master

**BYTE TMasterInfo::BuildRevisionIOLStack**

build revision of the IO-Link stack used by the master

---

## 2.7 TParameter Struct Reference

### 2.7.1 Data Fields

- BYTE [Result](#) [256]
- WORD [Index](#)
- BYTE [SubIndex](#)
- BYTE [Length](#)
- BYTE [ErrorCode](#)
- BYTE [AdditionalCode](#)

---

### 2.7.2 Detailed Description

[TParameter](#) contains the information which are used for ISDU read and write

---

### 2.7.3 Field Documentation

**BYTE TParameter::Result[256]**

buffer for data bytes (read and write)

**WORD TParameter::Index**

index of the variable to be read or written

**BYTE TParameter::SubIndex**

subindex of the variable to be read or written

**BYTE TParameter::Length**

length of the parameter data

**BYTE TParameter::ErrorCode**

error code for the result of the service

**BYTE TParameter::AdditionalCode**

additional error code of the result

---

## 2.8 TPortConfiguration Struct Reference

### 2.8.1 Data Fields

- BYTE [PortModeDetails](#)
- BYTE [TargetMode](#)
- BYTE [CRID](#)
- BYTE [DSConfigure](#)
- BYTE [Synchronisation](#)
- BYTE [FunctionID](#) [2]
- BYTE [InspectionLevel](#)
- BYTE [VendorID](#) [2]
- BYTE [DeviceID](#) [3]
- BYTE [SerialNumber](#) [16]
- BYTE [InputLength](#)
- BYTE [OutputLength](#)

---

### 2.8.2 Detailed Description

[TPortConfiguration](#) contains the port configuration information

---

### 2.8.3 Field Documentation

#### **BYTE TPortConfiguration::PortModeDetails**

additional info for the port

#### **BYTE TPortConfiguration::TargetMode**

Mode in which the port shall be run

#### **BYTE TPortConfiguration::CRID**

configured revision ID

#### **BYTE TPortConfiguration::DSConfigure**

Data Storage configuration

#### **BYTE TPortConfiguration::Synchronisation**

Synchronisation, not used

#### **BYTE TPortConfiguration::FunctionID[2]**

Function ID, not used

**BYTE TPortConfiguration::InspectionLevel**

NO\_CHECK, TYPE\_COMP, IDENTICAL

**BYTE TPortConfiguration::VendorID[2]**

validation: Vendor ID of the device

**BYTE TPortConfiguration::DeviceID[3]**

validation: Device ID of the device

**BYTE TPortConfiguration::SerialNumber[16]**

NULL-terminated string with the serial number

**BYTE TPortConfiguration::InputLength**

configured input length

**BYTE TPortConfiguration::OutputLength**

configured Output length

### 3. Index

ActualMode  
    TInfo 33  
    TInfoEx 34  
AdditionalCode  
    TParameter 37  
BIT\_CONNECTED  
    Global Definitions 6  
BIT\_EVENTAVAILABLE  
    Global Definitions 6  
BIT\_PDVALID  
    Global Definitions 6  
BIT\_PREOPERATE  
    Global Definitions 6  
BIT\_SENSORSTATEKNOWN  
    Global Definitions 6  
BIT\_WRONGSENSOR  
    Global Definitions 6  
Build  
    TDIInfo 31  
    TMasterInfo 35  
BuildRevisionIOLStack  
    TMasterInfo 36  
COM  
    TInfo 33  
    TInfoEx 34  
CRID  
    TPortConfiguration 38  
CurrentBaudrate  
    TInfo 34  
    TInfoEx 35  
Data Storage 28  
    DS\_CMD\_CLEAR 28  
    DS\_CMD\_DOWNLOAD 28  
    DS\_CMD\_UPLOAD 28  
    IOL\_DS\_Command 29  
    IOL\_DS\_ContentGet 29  
    IOL\_DS\_ContentSet 30  
Datum  
    TDIInfo 31  
DeviceID  
    TInfo 33  
    TPortConfiguration 39  
DirectParameterPage  
    TInfoEx 34  
DS\_CFG\_ENABLED  
    Port Configuration 14  
DS\_CFG\_UPLOAD\_ENABLED  
    Port Configuration 14  
DS\_CMD\_CLEAR  
    Data Storage 28  
DS\_CMD\_DOWNLOAD  
    Data Storage 28  
DS\_CMD\_UPLOAD  
    Data Storage 28  
DS\_FAULT\_DEVICE\_LOCKED  
    Event handling 26  
DS\_FAULT\_DOWNLOAD  
    Event handling 26  
DS\_FAULT\_IDENT  
    Event handling 26  
DS\_FAULT\_SIZE  
    Event handling 26  
DS\_FAULT\_UPLOAD  
    Event handling 26  
DSConfigure  
    TPortConfiguration 38  
ErrorCode  
    TParameter 37  
Event handling 23  
    DS\_FAULT\_DEVICE\_LOCKED 26  
    DS\_FAULT\_DOWNLOAD 26  
    DS\_FAULT\_IDENT 26  
    DS\_FAULT\_SIZE 26  
    DS\_FAULT\_UPLOAD 26  
    EVNT\_CODE\_DSREADY\_DOWNLOAD  
        26  
    EVNT\_CODE\_DSREADY\_NOACTION  
        26  
    EVNT\_CODE\_DSREADY\_UPLOAD 26  
    EVNT\_CODE\_M\_PDU\_CHECK 25  
    EVNT\_CODE\_M\_PREOPERATE 26  
    EVNT\_CODE\_P\_ACTOR 26  
    EVNT\_CODE\_P\_POWER 26  
    EVNT\_CODE\_P\_RESET 26  
    EVNT\_CODE\_P\_SENSOR 25  
    EVNT\_CODE\_P\_SHORT 25  
    EVNT\_CODE\_S\_DEVICELOST 25  
    EVNT\_CODE\_S\_FALLBACK 26  
    EVNT\_CODE\_S\_RETRY 25  
    EVNT\_CODE\_S\_WRONG\_COMP\_DE  
        VICEID 27  
    EVNT\_CODE\_S\_WRONG\_COMP\_VE  
        NDORID 27  
    EVNT\_CODE\_S\_WRONG\_COMP10\_D  
        EVICEID 27  
    EVNT\_CODE\_S\_WRONG\_COMP10\_V  
        ENDORID 27  
    EVNT\_CODE\_S\_WRONG\_CYCLE 27  
    EVNT\_CODE\_S\_WRONG\_PDINLENG  
        TH 27



EVNT_CODE_S_WRONG_PDOUTLEN	EVNT_CODE_S_WRONG_COMP10_DE
GTH 27	VICEID
EVNT_CODE_S_WRONG_REVISION	Event handling 27
27	EVNT_CODE_S_WRONG_COMP10_VE
EVNT_CODE_S_WRONG_SERNUM	NDORID
27	Event handling 27
EVNT_CODE_S_WRONGSENSOR 25	EVNT_CODE_S_WRONG_CYCLE
EVNT_INST_AL 24	Event handling 27
EVNT_INST_APPL 25	EVNT_CODE_S_WRONG_PDINLENGTH
EVNT_INST_DL 24	Event handling 27
EVNT_INST_PHL 24	EVNT_CODE_S_WRONG_PDOUTLENG
EVNT_INST_UNKNOWN 24	TH
EVNT_MODE_COMING 25	Event handling 27
EVNT_MODE_GOING 25	EVNT_CODE_S_WRONG_REVISION
EVNT_MODE_SINGLE 25	Event handling 27
EVNT_TYPE_ERROR 25	EVNT_CODE_S_WRONG_SERNUM
EVNT_TYPE_MESSAGE 25	Event handling 27
EVNT_TYPE_WARNING 25	EVNT_CODE_S_WRONGSENSOR
IOL_ReadEvent 27	Event handling 25
EventCode	EVNT_INST_AL
TEvent 32	Event handling 24
EVNT_CODE_DSREADY_DOWNLOAD	EVNT_INST_APPL
Event handling 26	Event handling 25
EVNT_CODE_DSREADY_NOACTION	EVNT_INST_DL
Event handling 26	Event handling 24
EVNT_CODE_DSREADY_UPLOAD	EVNT_INST_PHL
Event handling 26	Event handling 24
EVNT_CODE_M_PDU_CHECK	EVNT_INST_UNKNOWN
Event handling 25	Event handling 24
EVNT_CODE_M_PREOPERATE	EVNT_MODE_COMING
Event handling 26	Event handling 25
EVNT_CODE_P_ACTOR	EVNT_MODE_GOING
Event handling 26	Event handling 25
EVNT_CODE_P_POWER	EVNT_MODE_SINGLE
Event handling 26	Event handling 25
EVNT_CODE_P_RESET	EVNT_TYPE_ERROR
Event handling 26	Event handling 25
EVNT_CODE_P_SENSOR	EVNT_TYPE_MESSAGE
Event handling 25	Event handling 25
EVNT_CODE_P_SHORT	EVNT_TYPE_WARNING
Event handling 25	Event handling 25
EVNT_CODE_S_DEVICELOST	FunctionID
Event handling 25	TInfo 33
EVNT_CODE_S_FALLBACK	TPortConfiguration 38
Event handling 26	Global Definitions 5
EVNT_CODE_S_RETRY	BIT_CONNECTED 6
Event handling 25	BIT_EVENTAVAILABLE 6
EVNT_CODE_S_WRONG_COMP_DEVI	BIT_PDVALID 6
CEID	BIT_PREOPERATE 6
Event handling 27	BIT_SENSORSTATEKNOWN 6
EVNT_CODE_S_WRONG_COMP_VEND	BIT_WRONGSENSOR 6
ORID	MASK_SENSORSTATE 6
Event handling 27	RESULT_NOT_SUPPORTED 8
	RESULT_SERVICE_PENDING 8

RESULT_STATE_CONFLICT	7	Port Configuration	18
RESULT_WRONG_PARAMETER_STACK	8	IOL_GetUDPDevices	9
RETURN_CONNECTION_LOST	7	interface management	9
RETURN_DEVICE_ERROR	7	IOL_ReadEvent	27
RETURN_DEVICE_NOT_AVAILABLE	7	Event handling	27
RETURN_FIRMWARE_NOT_COMPATIBLE	6	IOL_ReadInputs	20
RETURN_FUNCTION_NOT_IMPLEMENTED	6	Process Data Handling	20
RETURN_INTERNAL_ERROR	7	IOL_ReadOutputs	19
RETURN_NO_EVENT	7	Process Data Handling	19
RETURN_OK	7	IOL_ReadReq	22
RETURN_OUT_OF_MEMORY	7	ISDU handling	22
RETURN_STATE_CONFLICT	6	IOL_SetCommand	17
RETURN_UART_TIMEOUT	7	Port Configuration	17
RETURN_UNKNOWN_HANDLE	7	IOL_SetPortConfig	15
RETURN_WRONG_COMMAND	7	Port Configuration	15
RETURN_WRONG_DEVICE	7	IOL_TransferProcessData	21
RETURN_WRONG_PARAMETER	7	Process Data Handling	21
Index		IOL_WriteOutputs	20
TPParameter	36	Process Data Handling	20
InputLength		IOL_WriteReq	23
TPortConfiguration	39	ISDU handling	23
InspectionLevel		ISDU handling	22
TPortConfiguration	39	IOL_ReadReq	22
Instance		IOL_WriteReq	23
TEvent	32	Length	
interface management	8	TPParameter	36
IOL_Create	8	LocalGenerated	
IOL_Destroy	9	TEvent	33
IOL_GetDLLInfo	10	Major	
IOL_GetMasterInfo	10	TMasterInfo	35
IOL_GetUDPDevices	9	MajorRevisionIOLStack	
IOL_Create		TMasterInfo	36
interface management	8	MASK_SENSORSTATE	
IOL_Destroy		Global Definitions	6
interface management	9	MasterCycle	
IOL_DS_Command		TInfo	34
Data Storage	29	Minor	
IOL_DS_ContentGet		TMasterInfo	35
Data Storage	29	MinorRevisionIOLStack	
IOL_DS_ContentSet		TMasterInfo	36
Data Storage	30	Mode	
IOL_GetDLLInfo		TEvent	32
interface management	10	Name	
IOL_GetMasterInfo		TDeviceIdentification	31
interface management	10	NetworkName	
IOL_GetMode		TDeviceIdentification	31
Port Configuration	16	Number	
IOL_GetModeEx		TEvent	32
Port Configuration	18	OutputLength	
IOL_GetSensorStatus		TPortConfiguration	39
		PDValid	
		TEvent	32
		Port	
		TEvent	32

Port Configuration	11	Global Definitions	8
DS_CFG_ENABLED	14	RESULT_SERVICE_PENDING	
DS_CFG_UPLOAD_ENABLED	14	Global Definitions	8
IOL_GetMode	16	RESULT_STATE_CONFLICT	
IOL_GetModeEx	18	Global Definitions	7
IOL_GetSensorStatus	18	RESULT_WRONG_PARAMETER_STACK	
IOL_SetCommand	17	Global Definitions	8
IOL_SetPortConfig	15	RETURN_CONNECTION_LOST	
SM_BAUD_19200	14	Global Definitions	7
SM_BAUD_230400	14	RETURN_DEVICE_ERROR	
SM_BAUD_38400	14	Global Definitions	7
SM_COMMAND_FALLBACK	13	RETURN_DEVICE_NOT_AVAILABLE	
SM_COMMAND_OPERATE	13	Global Definitions	7
SM_COMMAND_PD_OUT_INVALID	13	RETURN_FIRMWARE_NOT_COMPATIBLE	
SM_COMMAND_PD_OUT_VALID	13	Global Definitions	6
SM_COMMAND_RESTART	13	RETURN_FUNCTION_NOT_IMPLEMENTED	
SM_MODE_DIAGNOSTIC_INPUT	14	Global Definitions	6
SM_MODE_INVERT_INPUT	14	RETURN_INTERNAL_ERROR	
SM_MODE_IOLINK_FALLBACK	13	Global Definitions	7
SM_MODE_IOLINK_OPER_FALLBACK	13	RETURN_NO_EVENT	
SM_MODE_IOLINK_OPERATE	13	Global Definitions	7
SM_MODE_IOLINK_PREOP	12	RETURN_OK	
SM_MODE_IOLINK_PREOP_FALLBACK	13	Global Definitions	7
SM_MODE_NORMAL_INPUT	14	RETURN_OUT_OF_MEMORY	
SM_MODE_RESET	12	Global Definitions	7
SM_MODE_SIO_HS_SWITCH	13	RETURN_STATE_CONFLICT	
SM_MODE_SIO_INPUT	13	Global Definitions	6
SM_MODE_SIO_LS_SWITCH	14	RETURN_UART_TIMEOUT	
SM_MODE_SIO_OUTPUT	13	Global Definitions	7
SM_MODE_SIO_PP_SWITCH	13	RETURN_UNKNOWN_HANDLE	
SM_VALIDATION_MODE_COMPATIBLE	14	Global Definitions	7
SM_VALIDATION_MODE_IDENTICAL	14	RETURN_WRONG_COMMAND	
SM_VALIDATION_MODE_NONE	14	Global Definitions	7
STATE_DISCONNECTED_GETMODE	15	RETURN_WRONG_DEVICE	
STATE_OPERATE_GETMODE	15	Global Definitions	7
STATE_PREOPERATE_GETMODE	15	RETURN_WRONG_PARAMETER	
STATE_WRONGSENSOR_GETMODE	15	Global Definitions	7
PortModeDetails		SensorState	
TPortConfiguration	38	TInfo	34
Process Data Handling	19	SensorStatus	
IOL_ReadInputs	20	TInfoEx	35
IOL_ReadOutputs	19	SerialNumber	
IOL_TransferProcessData	21	TPortConfiguration	39
IOL_WriteOutputs	20	SM_BAUD_19200	
Result		Port Configuration	14
TParameter	36	SM_BAUD_230400	
RESULT_NOT_SUPPORTED		Port Configuration	14
		SM_BAUD_38400	
		Port Configuration	14
		SM_COMMAND_FALLBACK	
		Port Configuration	13

SM_COMMAND_OPERATE	TargetMode
Port Configuration 13	TPortConfiguration 38
SM_COMMAND_PD_OUT_INVALID	TDeviceIdentification 30
Port Configuration 13	Name 31
SM_COMMAND_PD_OUT_VALID	NetworkName 31
Port Configuration 13	TDllInfo 31
SM_COMMAND_RESTART	Build 31
Port Configuration 13	Datum 31
SM_MODE_DIAGNOSTIC_INPUT	Version 31
Port Configuration 14	TEvent 32
SM_MODE_INVERT_INPUT	EventCode 32
Port Configuration 14	Instance 32
SM_MODE_IOLINK_FALLBACK	LocalGenerated 33
Port Configuration 13	Mode 32
SM_MODE_IOLINK_OPER_FALLBACK	Number 32
Port Configuration 13	PDValid 32
SM_MODE_IOLINK_OPERATE	Port 32
Port Configuration 13	Type 32
SM_MODE_IOLINK_PREOP	TInfo 33
Port Configuration 12	ActualMode 33
SM_MODE_IOLINK_PREOP_FALLBACK	COM 33
Port Configuration 13	CurrentBaudrate 34
SM_MODE_NORMAL_INPUT	DeviceID 33
Port Configuration 14	FunctionID 33
SM_MODE_RESET	MasterCycle 34
Port Configuration 12	SensorState 34
SM_MODE_SIO_HS_SWITCH	VendorID 33
Port Configuration 13	TInfoEx 34
SM_MODE_SIO_INPUT	ActualMode 34
Port Configuration 13	COM 34
SM_MODE_SIO_LS_SWITCH	CurrentBaudrate 35
Port Configuration 14	DirectParameterPage 34
SM_MODE_SIO_OUTPUT	SensorStatus 35
Port Configuration 13	TMasterInfo 35
SM_MODE_SIO_PP_SWITCH	Build 35
Port Configuration 13	BuildRevisionIOLStack 36
SM_VALIDATION_MODE_COMPATIBLE	Major 35
Port Configuration 14	MajorRevisionIOLStack 36
SM_VALIDATION_MODE_IDENTICAL	Minor 35
Port Configuration 14	MinorRevisionIOLStack 36
SM_VALIDATION_MODE_NONE	Version 35
Port Configuration 14	TParameter 36
STATE_DISCONNECTED_GETMODE	AdditionalCode 37
Port Configuration 15	ErrorCode 37
STATE_OPERATE_GETMODE	Index 36
Port Configuration 15	Length 36
STATE_PREOPERATE_GETMODE	Result 36
Port Configuration 15	SubIndex 36
STATE_WRONGSENSOR_GETMODE	TPortConfiguration 38
Port Configuration 15	CRID 38
SubIndex	DeviceID 39
TParameter 36	DSConfigure 38
Synchronisation	FunctionID 38
TPortConfiguration 38	InputLength 39

InspectionLevel	39	TEvent	32
OutputLength	39	VendorID	
PortModeDetails	38	TInfo	33
SerialNumber	39	TPortConfiguration	39
Synchronisation	38	Version	
TargetMode	38	TDllInfo	31
VendorID	39	TMasterInfo	35
Type			