Technologie
Management Gruppe
Technologie und Engineering

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
Internet 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

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:

Device	IP adress of the IO-Link Master , e.g. "192.168.0.110"

Return values:

RETURN_DEVICE_NOT_AVAILABLE	the Device referred by the string parameter "Device" is not available or busy
RETURN_COMM_TIMEOUT	the device did not respond in time
RETURN_OUT_OF_MEMORY	no more Handles can be assigned
RETURN_WRONG_PARAMETER	the device name was wrong
RETURN_FIRMWARE_NOT_COMPATIBLE	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 coresponding 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:

Handle	Handle to work on/with

Return values:

RETURN_UNKNOWN_HANDLE	Handle is not valid
RETURN_INTERNAL_ERROR	Error that should not occur.
RETURN_OK	Everything worked out allright

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 informaion of the device.
Note:

The memory containing the resulting list must be allocated by the applicaation. 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:

pDeviceList	pointer to a buffer for the result
MaxNumberOf Entries	max number of entries which can be put in the buffer

Return values:

number	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:

Handle	Handle to work on/with
pMasterInfo	Pointer to TMasterinfo structure

Return values:

RETURN_UNKNOWN_HANDLE	Handle is not valid
RETURN_INTERNAL_ERROR	Error that should not occur.
RETURN_OK	Everything worked out allright

LONG __stdcall IOL_GetDLLInfo (TDllInfo * *pDllInfo*)

Get information about the DLL.

This function returns the Version information from the DLL
Parameters:

pDllInfo	Pointer to TDllInfo structure

Return values:

RETURN_OK	Everything worked out allright

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 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:

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

Return values:

RETURN_UNKNOWN_HANDLE	Handle is not valid
RETURN_INTERNAL_ERROR	Error that should not occur.
RETURN_OK	Everything worked out allright

LONG __stdcall IOL_GetMode (LONG *Handle*, DWORD *Port*, TInfo * *pInfo*)

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 pInfo.

- 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:

Handle	Handle to work on/with
Port	Port number of the used port
pInfo	Pointer to TInfo structure

Return values:

RETURN_UNKNOWN_HANDLE	Handle is not valid
RETURN_INTERNAL_ERROR	Error that should not occur.
RETURN_OK	Everything worked out allright

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:

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

Return values:

RETURN_UNKNOWN_HAND	Handle is not valid

LE	
RETURN_INTERNAL_ERROR	Error that should not occur.
RETURN_OK	Everything worked out allright

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:

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

Return values:

RETURN_UNKNOWN_HANDLE	Handle is not valid
RETURN_INTERNAL_ERROR	Error that should not occur.
RETURN_OK	Everything worked out allright

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:

Handle	Handle to work on/with

Port	Port number of the used port
pInfoEx	Pointer to TInfo structure
OnlyStatus	

Return values:

RETURN_UNKNOWN_HANDLE	Handle is not valid
RETURN_INTERNAL_ERROR	Error that should not occur.
RETURN_OK	Everything worked out allright

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 loggin 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:

Handle	Handle to work on/with

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

Return values:

RETURN_UNKNOWN_HANDLE	Handle is not valid
RETURN_INTERNAL_ERROR	Error that should not occur.
RETURN_OK	Everything worked out allright

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:

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

Return values:

RETURN_UNKNOWN_HANDLE	Handle is not valid
RETURN_INTERNAL_ERROR	Error that should not occur.
RETURN_OK	Everything worked out allright

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 refered by ProcessData to the IO-Link Master. The data is then transferred to the connected Sensor.
Parameters:

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

Return values:

RETURN_UNKNOWN_HANDLE	Handle is not valid
RETURN_INTERNAL_ERROR	Error that should not occur.
RETURN_OK	Everything worked out allright

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:

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

Return values:

RETURN_UNKNOWN_HANDLE	Handle is not valid
RETURN_INTE	Error that should not occur.

RNAL_ERROR	
RETURN_OK	Everything worked out allright

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:

Handle	Handle to work on/with
Port	Port number of the used port
pParameter	Pointer to TParameter struct

Return values:

RETURN_UNKNOWN_HANDLE	Handle is not valid
RETURN_INTERNAL_ERROR	Error that should not occur.
RETURN_OK	Everything worked out allright

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:

Handle	Handle to work on/with
Port	Port number of the used port
pParameter	Pointer to TParameter struct

Return values:

RETURN_UNKNOWN_HANDLE	Handle is not valid
RETURN_INTERNAL_ERROR	Error that should not occur.
RETURN_OK	Everything worked out allright

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 occuring 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:

Handle	Handle to work on/with
pEvent	Pointer to a TEvent struct

Status	Status information (Events, Processdata Valid, ...)

Return values:

RETURN_UNKNOWN_HANDLE	Handle is not valid
RETURN_INTERNAL_ERROR	Error that should not occur.
RETURN_OK	Everything worked out allright

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:

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

Return values:

RETURN_UNKNOWN_HANDLE	Handle is not valid
RETURN_INTERNAL_ERROR	Error that should not occur.
RETURN_OK	Everything worked out allright

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:

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

Return values:

RETURN_UNK	Handle is not valid

NOWN_HANDLE	
RETURN_INTERNAL_ERROR	Error that should not occur.
RETURN_OK	Everything worked out allright

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:

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

Return values:

RETURN_UNKNOWN_HANDLE	Handle is not valid
RETURN_INTERNAL_ERROR	Error that should not occur.
RETURN_OK	Everything worked out allright

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 occured 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 occured

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 contans the information about a connected sensor and the stat e 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