Interface Manual
LANIICMS.DLL
TCP/IP – I2C

(Rev1.05)

COPTONIX

Luxemburger Str. 31
D – 13353 Berlin
Phone: +49 – (0)30 – 61 74 12 48
Fax: +49 – (0)30 – 61 74 12 47
www.coptonix.com

Dynamic Link Library DLL

1 General Functions

1.1 InitNetIIC

function initnetiic (pIRQCallBack, pSlvCallBack: Pointer; pcMsgID : PChar) : DWORD Stdcall;
On starting the host application this function must be called if callback functions are needed. The arguments ***pIRQCallBack*** and ***pSlvCallBack*** are the addresses of callback functions.

TIRQCallBack = Procedure; stdcall;
This function is called when an interrupt has been detected.

TSlvCallBack = Procedure(const data : Pointer, wSize : Word); stdcall;
This function is called when the converter is in SLAVE_MODE and it receives data from a master.
data is a pointer to the data were received, and ***wSize*** is the number of bytes were received.

If callback functions are not needed, then just pass ***nil*** to the function.
e.g. iniusbiic(***nil***,***nil***,'My_IIC_Msg_ID');

This function defines also a new window message that is guaranted to be unique throughout the system. The argument ***pcMsgID*** (as PChar e.g. "My_LANI2C_App" should be unique) . If more than one application use the same ***pcMsgID***, so they will share the same window message ID. If the function successes, the return value is a message identifier in the range 0xC000 through 0xFFFF. If the function fails, the return value is zero. The return value must be saved in a global valid variable in order to use it later in processing messages. For the registration of the new window message the window API function "RegisterWindowMessage" is used.

1.2 Connect

function connect (shost : PChar; iport : integer; dwTimeOut : DWORD) : DWORD Stdcall;
Use this function to connect to the *LAN I2C Converter MS* using the IP Address ***shost*** (e.g. "192.168.1.100") and the port number ***iport*** (e.g. 5000). ***dwTimeOut*** is time out value in millisecond. If the function fails the return value is non zero.

Default Network settings:

GAR (*Gateway IP Address Register*):	192.168.1.1
SUBR (*Subnet Mask Register*):	255.255.255.0
IP (*IP Address Register*):	192.168.1.100
SHAR (*Hardware Address register -MAC-*):	00-00-00-00-00-00
TCP/IP Port:	5000
UDP Port:	3000

1.3 Disconnect

function disconnect : DWORD Stdcall;
Use this function to disconnect from *LAN I2C Converter MS*. If the function fails the return value is non zero.

1.4 Connected

function connected : DWORD Stdcall;
The function returns zero if connection is established.

1.5 SetMode

function setmode(var ucMode, ucSlvAddr : Byte;
dwTimeOut : DWORD) : DWORD; stdcall;
Selects one of three modes: MASTER_MODE, SLAVE_MODE or PROG_MODE.

ucMode = 0 -> will not effect any changes. The function returns current mode.
ucMode = 1 -> MASTER_MODE
ucMode = 2 -> SLAVE_MODE
ucMode = 3 -> PROG_MODE

ucSlvAddr is the own slave address of the converter (valid only in SLAVE_MODE).
If the LSB is 1, then the General Call is active. ***dwTimeOut*** is time out value in millisecond.
If the function fails the return value is non zero.

1.6 GetNetConfig

function getnetconfig(var pNetConfig; wSize : WORD;
dwTimeOut : DWORD) : DWORD; stdcall;
reads current network settings from the *LAN I2C Converter MS*. ***pNetConfig*** is a pointer that receives the data read from the device. This buffer must remain valid for the duration of the write operation. ***dwTimeOut*** is time out value in millisecond. If the function fails the return value is non zero.

```
TNetConfig = record
    SHAR       : array[0..5] of byte;   //e.g. [0x00,0x00,0x00, 0x00,0x00,0x00]
    DHCP       : Byte;                  //Bit0: ucActive; Bit1: ucUseTimeOut
    RESERVED   : Byte;                  //reserved byte 0x00
    GAR        : DWORD;                 //e.g. 0xC0A80101 = 192.168.1.1
    SUBR       : DWORD;                 //e.g. 0xFFFFFF00 = 255.255.255.0
    IP         : DWORD;                 //e.g. 0xC0A80164 = 192.168.1.100
    TCPPORT    : WORD;                  //e.g. 5000
    UDPPORT    : WORD;                  //e.g. 3000
End;
PTNetConfig = ^TNetConfig;
```

1.7 MemWR8Bit

function memwr8bit(wAddr : WORD; ucVal : Byte;
dwTimeOut : DWORD) : DWORD; stdcall;
writes ***ucVal*** (1 BYTE / 8Bit value) into the onboard memory at address ***wAddr***. ***dwTimeOut*** is time out value in millisecond. If the function fails the return value is non zero.

1.8 MemWR16Bit

function memwr16bit(wAddr : WORD; wVal : WORD;
dwTimeOut : DWORD) : DWORD; stdcall;

writes ***wVal*** (1 WORD / 16Bit value) into the onboard memory at address ***wAddr***. ***dwTimeOut*** is time out value in millisecond. If the function fails the return value is non zero.

1.9 MemWR32Bit

function memwr32bit(wAddr : WORD; dwVal : DWORD;
dwTimeOut : DWORD) : DWORD; stdcall;

writes ***dwVal*** (1 DWORD / 32Bit value) into the onboard memory at address ***wAddr***. ***dwTimeOut*** is time out value in millisecond. If the function fails the return value is non zero.

1.10 MemWRBlock

function memwrblock(wAddr : WORD; const pmemdata; wSize : WORD;
dwTimeOut : DWORD) : DWORD; stdcall;

writes data into the onbaord memory. ***pmemdata*** is a pointer to the buffer containing the data to be written to the memory. This buffer must remain valid for the duration of the write operation. ***wSize*** is the number of bytes to be written to the memory. ***dwTimeOut*** is time out value in millisecond. If the function fails the return value is non zero.

1.11 MemRD8Bit

function memrd8bit(wAddr : WORD; var ucVal : Byte;
dwTimeOut : DWORD) : DWORD; stdcall;

reads 1 BYTE (8Bit) from memory at address ***wAddr***. ***ucVal*** returns the BYTE read. ***dwTimeOut*** is time out value in millisecond. If the function fails the return value is non zero.

1.12 MemRD16Bit

function memrd16bit(wAddr : WORD; var wVal : WORD;
dwTimeOut : DWORD) : DWORD; stdcall;

reads 1 WORD (16Bit) from memory at address ***wAddr***. ***wVal*** returns the WORD read. ***dwTimeOut*** is time out value in millisecond. If the function fails the return value is non zero.

1.13 MemRD32Bit

function memrd32bit(wAddr : WORD; var dwVal : DWORD;
dwTimeOut : DWORD) : DWORD; stdcall;

reads 1 DWORD (32Bit) from memory at address ***wAddr***. ***dwVal*** returns the DWORD read. ***dwTimeOut*** is time out value in millisecond. If the function fails the return value is non zero.

1.14 MemRDBlock

function memrdblock(wAddr : WORD; var pmemdata; var wSize : WORD;
dwTimeOut : DWORD) : DWORD; stdcall;

reads number of bytes (***wSize***) of data from memory starting at address ***wAddr***. ***pi2cdata*** is a pointer that receives the data read from a memory. This buffer must remain valid for the duration of the write operation. ***wSize*** is the number of bytes to be read from the memory. ***wSize*** returns also the number of bytes read from the memory. ***dwTimeOut*** is time out value in millisecond. If the function fails the return value is non zero.

1.15 SaveSettings

function savesettings(dwTimeOut : DWORD) : DWORD; stdcall;

By calling the function "savesettings", the settings (operation mode, slave address and SCL frequency) are stored into onboard memory. ***dwTimeOut*** is time out value in millisecond. If the function fails the return value is non zero.

1.16 ResetSettings

function resetsettings(dwTimeOut : DWORD) : DWORD; stdcall;

loads factory (default) settings:
MASTER_MODE
SLAVE ADDRESS = 0x00
SCL FREQUENCY = 100 kHz. (wSCLL = 0x012C and wSCLH = 0x012C)
dwTimeOut is time out value in millisecond. If the function fails the return value is non zero.

1.17 GetErrorString

function geterrorstring(dwErr : DWORD) : PCHAR; stdcall;

To obtain an error string for error codes, use the function *GetErrorString*. For a complete list of error codes provided by the DLL, see section *5.1 Error codes*. ***dwErr*** is the return value of all functions mentioned in this manual. The return value is the error string.

2 PROG_MODE Functions

2.1 SetGAR

function setgar (ucVal1, ucVal2, ucVal3, ucVal4: Byte; dwTimeOut : DWORD) : DWORD Stdcall;
sets new value for GAR (Gateway IP Address Register).
e.g. 192.168.1.1
ucVal1 = 192
ucVal2 = 168
ucVal3 = 1
ucVal4 = 1
dwTimeOut is time out value in millisecond. If the function fails the return value is non zero.

2.2 SetSUBR

function setsubr (ucVal1, ucVal2, ucVal3, ucVal4: Byte; dwTimeOut : DWORD) : DWORD Stdcall;
sets new value for SUBR (Subnet Mask Register).
e.g. 255.255.255. 0
ucVal1 = 255
ucVal2 = 255
ucVal3 = 255
ucVal4 = 0
dwTimeOut is time out value in millisecond. If the function fails the return value is non zero.

2.3 SetSHAR

function setshar (ucVal1, ucVal2, ucVal3, ucVal4, ucVal5, ucVal6: Byte; dwTimeOut : DWORD) : DWORD Stdcall;
sets new value for SHAR (Hardware Address Register -MAC-).
e.g. 00-50-C2-00-00-00
ucVal1 = 0x00
ucVal2 = 0x50
ucVal3 = 0xC2
ucVal4 = 0x00
ucVal5 = 0x00
ucVal6 = 0x00
dwTimeOut is time out value in millisecond. If the function fails the return value is non zero.

2.4 SetIP

function setip (ucVal1, ucVal2, ucVal3, ucVal4: Byte; dwTimeOut : DWORD) : DWORD Stdcall;
sets new value for IP Address (IP Address Register).
e.g. 192.168.1.100
ucVal1 = 192
ucVal2 = 168
ucVal3 = 1
ucVal4 = 100
dwTimeOut is time out value in millisecond. If the function fails the return value is non zero.

2.5 SetTCPPort

function settcpport (wPort: Word; dwTimeOut : DWORD) : DWORD Stdcall;
sets new value for TCP/IP listening port. ***wPort*** is the port number (e.g. 5000).
dwTimeOut is time out value in millisecond. If the function fails the return value is non zero.

2.6 SetUDPPort

function setudpport (wPort: Word; dwTimeOut : DWORD) : DWORD Stdcall;
sets new value for UDP port. ***wPort*** is the port number (e.g. 3000).
dwTimeOut is time out value in millisecond. If the function fails the return value is non zero.

2.7 LoadFactorySettings

function loadfactorysettings (dwTimeOut : DWORD) : DWORD Stdcall;
loads factory (default) settings. On next PowerOn **or** after calling the function *ReInit* (*s. 2.8*) the converter uses new settings. ***dwTimeOut*** is time out value in millisecond. If the function fails the return value is non zero.

Default Network settings:

GAR (*Gateway IP Address Register*):	192.168.1.1
SUBR (*Subnet Mask Register*):	255.255.255.0
IP (*IP Address Register*):	192.168.1.100
SHAR (*Hardware Address register -MAC-*):	00-00-00-00-00-00
TCP/IP Port:	5000
UDP Port:	3000

2.8 ReInit

function reinit (dwTimeOut : DWORD) : DWORD Stdcall;
New Network settings becomes valid after PowerOn or after calling the function "*ReInit*". The Function "*ReInit*" reinitialize the converter. ***dwTimeOut*** is time out value in millisecond. If the function fails the return value is non zero.

2.9 MemProtect

function memprotect(var ucProtect : Byte; dwTimeOut : DWORD) : DWORD; stdcall;
The *memprotect* function is useful for protecting the contents of the on board memory (only user memory > 0xFF) from inadvertent write operations. Write operations are disabled to the memory when *ucProtect* is set to 0x01. When *ucProtect* is set to 0x02 write operations are allowed.

ucProtect = 0 -> will not effect any changes. The function would only return current settings.
ucProtect = 1 -> Write Operations disabled
ucProtect = 2 -> Write Operations enabled

dwTimeOut is time out value in millisecond.
If the function fails the return value is non zero.

2.10 SetDHCP

function setdhcp(ucActive, ucUseTimeOut : Byte;
dwTimeOut : DWORD) : DWORD; stdcall;

SetDHCP activates/ deactivates automatic IP and network settings.

ucActive = 0 -> Read IP address and network settings from EEPROM.
ucActive = 1 -> Obtain an IP address and network settings automatically from DHCP server.

ucUseTimeOut = 0 -> Infinite wait for response from DHCP server.
ucUseTimeOut = 1 -> TimeOut is used. If a DHCP server is not available, then default IP address and network settings are used.

dwTimeOut is time out value in millisecond.
If the function fails the return value is non zero.

3 MASTER_MODE Functions

3.1 WriteI2C

function writei2c(ucSlvAddr : Byte; const pi2cdata; wSize : WORD;
var wStatus : WORD; dwTimeOut : DWORD) : DWORD; stdcall;

writes data to a I2C device. ***ucSlvAddr*** is the slave address of a I2C device. ***pi2cdata*** is a pointer to the buffer containing the data to be written to a device. This buffer must remain valid for the duration of the write operation.
wSize is the number of bytes to be written to the device. ***wStatus*** returns the state of the I2C bus. ***dwTimeOut*** is time out value in millisecond. If the function fails the return value is non zero.

3.2 ReadI2C

function readi2c(ucSlvAddr : Byte; var pi2cdata; var wSize, wStatus : WORD;
dwTimeOut : DWORD) : DWORD; stdcall;

reads data from a I2C device. ***ucSlvAddr*** is the slave address of a I2C device. ***pi2cdata*** is a pointer that receives the data read from a device. This buffer must remain valid for the duration of the write operation.
wSize is the number of bytes to be read from the device. ***wSize*** returns also the number of bytes read from the device. ***wStatus*** returns the state of the I2C bus. ***dwTimeOut*** is time out value in millisecond. If the function fails the return value is non zero.

3.3 WRDI2C

function wrdi2c(ucSlvAddr : Byte; const pwrdata; wwrSize : WORD;
var prddata; var wrdSize, wStatus : WORD;
dwTimeOut : DWORD) : DWORD; stdcall;

writes and reads data from a I2C device. ***ucSlvAddr*** is the slave address of a I2C device. ***pwrdata*** is a pointer to the buffer containing the data to be written to a device. This buffer must remain valid for the duration of the write operation. ***wwrSize*** is the number of bytes to be written to the device. ***prddata*** is a pointer that receives the data read from a device. This buffer must remain valid for the duration of the write operation. ***wrdSize*** is the number of bytes to be read from the device. ***wrdSize*** returns also the number bytes read from the device. ***wStatus*** returns the state of the I2C bus. ***dwTimeOut*** is time out value in millisecond. If the function fails the return value is non zero.

3.4 ChkSlvAddr

function chkslvaddr(ucslvaddr : Byte; var ucConnected : byte;
dwTimeOut : DWORD) : DWORD; stdcall;

checks if the slave address ***ucSlvAddr*** is connected to the I2C bus. If ***ucConnected*** returns 0, then the device is not connected. If ***ucConnected*** returns 1, then the device is connected to the I2C bus. ***dwTimeOut*** is time out value in millisecond. If the function fails the return value is non zero.

3.5 ScanI2C

function scani2c(var pscandata; var ucSize : byte;
dwTimeOut : DWORD) : DWORD; stdcall;

scans all I2C device currently connected to the bus. ***pscandata*** is a pointer that receives the list of slave addresses (devices) were detected. This buffer must remain valid for the duration of the scan operation. ***ucSize*** returns the number of devices were detected. ***dwTimeOut*** is time out value in millisecond. If the function fails the return value is non zero.

3.6 SetSCLHiLo

function setsclhilo(wSCLH, wSCLL: WORD; dwTimeOut : DWORD) : DWORD; stdcall;

Use this function to set the I2C clock frequency and the duty cycle. ***wSCLH*** determines the high time and ***wSCLL*** the low time of the I2C clock. ***wSCLH*** and ***wSCLL*** together determine the clock frequency generated by the master. The clock frequency is determined by the following formula:

$$I2C_{frequency} = 60000000 / (wSCLH + wSCLL)$$

The values for ***wSCLH*** and ***wSCLL*** should not necessarily be the same. Software can set different duty cycles on SCL by setting different values ***wSCLH*** and ***wSCLL***. The values must ensure the data rate is in the data rate range of 500 Hz through 1MHz.
dwTimeOut is time out value in millisecond. If the function fails the return value is non zero.

3.7 GetSCLHiLo

function getsclhilo(var wSCLH, wSCLL: WORD;
dwTimeOut : DWORD) : DWORD; stdcall;

reads the current settings for I2C SCL clock frequency (see *SetSCLHiLo* for further information). ***dwTimeOut*** is time out value in millisecond. If the function fails the return value is non zero.

3.8 SetSCL

function setscl(dwfreq : DWORD; dwTimeOut : DWORD) : DWORD; stdcall;

sets a new value for I2C clock frequency. ***dwfreq*** is the new frequency (in [Hz]) in the range of 500Hz through 1MHz. ***dwTimeOut*** is time out value in millisecond. If the function fails the return value is non zero.

3.9 GetSCL

function getscl(var dwfreq : DWORD; dwTimeOut : DWORD) : DWORD; stdcall;

reads the value for I2C clock frequency. ***dwfreq*** is the frequency (in [Hz]) in the range of 500Hz through 1MHz. ***dwTimeOut*** is time out value in millisecond. If the function fails the return value is non zero.

3.10 ResetIRQ

function resetirq(var ucIRQMode : Byte; dwTimeOut : DWORD) : DWORD; stdcall;
The converter has an interrupt input. e.g. if you use an IO-Expander, you could connect it's interrupt output with the interrupt input of the converter. Then your software would recognize all events of the IO-Expander if the state of the IOs has been changed.
Possible values for ***ucIRQMode***:
0: disable interrupt.
1: enable interrupt. Interrupt is falling-edge sensitive.
2: enable interrupt. Interrupt is rising-edge sensitive.
dwTimeOut is time out value in millisecond. If the function fails the return value is non zero.

3.11 GetI2CStatusString

function geti2cstatusstring(wStatus : WORD) : PCHAR; stdcall;
To obtain a status string of the last I2C read/write operation, use the function *GetI2CStatusString*. ***wStatus*** is the value returned when calling the functions *WriteI2C*, *ReadI2C* and *WRDI2C*. The return value is the status string.

4 SLAVE_MODE Functions

4.1 SetI2CSlvAddr

function seti2cslvaddr(var ucslvaddr : Byte; dwTimeOut : DWORD) : DWORD; stdcall;
If the converter operates in SLAVE_MODE, so it is possible to change the slave address any time using this function. ***ucslvaddr*** is the new slave address of the converter. ***dwTimeOut*** is time out value in millisecond. If the function fails the return value is non zero.

4.2 WriteI2CSlaveBuf

function writei2cslvbuf(const pi2cdata; wSize : WORD; ucEvent: Byte;
dwTimeOut : DWORD) : DWORD; stdcall;
writes data to the I2C slave output buffer. ***pi2cdata*** is a pointer to the buffer containing the data to be written to the buffer. This buffer must remain valid for the duration of the write operation. ***wSize*** is the number of bytes to be written to the buffer. If ***ucEvent*** is set to 1, the converter sends an interrupt signal to the master. Thus the converter informs the master that there is data ready to read. ***dwTimeOut*** is time out value in millisecond. If the function fails the return value is non zero.

5. Further definitions

5.1 Error codes

$00000000	RET_OK
$00010001	ERR_UNABLE_TO_CONNECT
$00010002	ERR_UNABLE_TO_DISCONNECT
$00010003	ERR_DISCONNECTED
$00020001	ERR_NET_WRITE
$00020002	ERR_NET_READ_TIMEOUT
$00020003	ERR_NET_READ_WAIT_FAILED
$00020004	ERR_NET_READ_WAIT_ABANDONED
$00020005	ERR_NET_IO_CALLBACK
$00030001	ERR_RD_TH_TIMEOUT
$00030002	ERR_RD_TH_WAIT_FAILED
$00030003	ERR_RD_TH_WAIT_ABANDONED
$00040001	ERR_CREATE_SYNC_EVENT
$00040002	ERR_IO_TO_TH_PULSEEVENT
$00040003	ERR_TH_TO_IO_SETEVENT
$00040004	ERR_TH_TO_IO_RESETEVENT
$00050001	ERR_CREATE_MMF_HANDLE
$00050002	ERR_CREATE_MMF
$00050003	ERR_CREATE_MMF_LOCK_HANDLE
$00050004	ERR_LOCK_MMF
$00050005	ERR_MMF_MAX_APP_NUM_REACHED
$00070001	ERR_WRI2C_INVALID_SIZE
$00070002	ERR_WRI2C_FAILED
$00070003	ERR_WRI2C_FAILED_UNKNOWN
$00070004	ERR_RDI2C_INVALID_SIZE
$00070005	ERR_RDI2C_FAILED
$00070006	ERR_RDI2C_FAILED_UNKNOWN
$00070007	ERR_SCNI2C_FAILED_UNKNOWN
$00070008	ERR_SETSCL_FAILED_UNKNOWN
$00070009	ERR_GETSCL_FAILED_UNKNOWN
$0007000A	ERR_INVALID_SCL_FREQUENCY
$0007000B	ERR_SET_IRQ_FAILED_UNKNOWN
$0007000C	ERR_WRDI2C_FAILED
$0007000D	ERR_WRDI2C_FAILED_UNKNOWN
$00070010	ERR_WRI2C_SLAVE_BUF
$00070011	ERR_SET_SLAVE_ADDR
$00070012	ERR_CHK_SLAVE_ADDR
$00080001	ERR_SETGAR_FAILED
$00080002	ERR_SETGAR_FAILED_UNKNOWN
$00080003	ERR_SETSUBR_FAILED
$00080004	ERR_SETSUBR_FAILED_UNKNOWN
$00080005	ERR_SETSHAR_FAILED

$00080006	ERR_SETSHAR_FAILED_UNKNOWN
$00080007	ERR_SETIP_FAILED
$00080008	ERR_SETIP_FAILED_UNKNOWN
$00080009	ERR_SETTCPPORT_FAILED
$0008000A	ERR_SETTCPPORT_FAILED_UNKNOWN
$0008000B	ERR_SETUDPPORT_FAILED
$0008000C	ERR_SETUDPPORT_FAILED_UNKNOWN
$0008000D	ERR_GETCONFIG_FAILED_UNKNOWN
$0008000E	ERR_REINIT_FAILED_UNKNOWN
$0008000F	ERR_MEM_PROTECT_FAILED
$00080010	ERR_MEM_PROTECT_FAILED_UNKNOWN
$00090001	ERR_NET_MODE
$00090002	ERR_SET_MODE_FAILED_UNKNOWN
$00090003	ERR_MEMWRITE_FAILED
$00090004	RR_MEMWRITE_FAILED_UNKNOWN
$00090005	ERR_MEMREAD_FAILED
$00090006	ERR_MEMREAD_FAILED_UNKNOWN
$00090007	ERR_WRMEM_INVALID_SIZE
$00090008	ERR_RDMEM_INVALID_SIZE
$00090009	ERR_SAVE_SETTINGS_FAILED
$0009000A	ERR_SAVE_SETTINGS_FAILED_UNKNOWN
$0009000B	ERR_RESET_SETTINGS_FAILED
$0009000C	ERR_RESET_SETTINGS_FAILED_UNKNOWN
$000A0001	ERR_UNKNOWN_CMDID
$000A0002	ERR_UNKNOWN_CMD
$000A0003	ERR_MODE_CMD