SYBERA

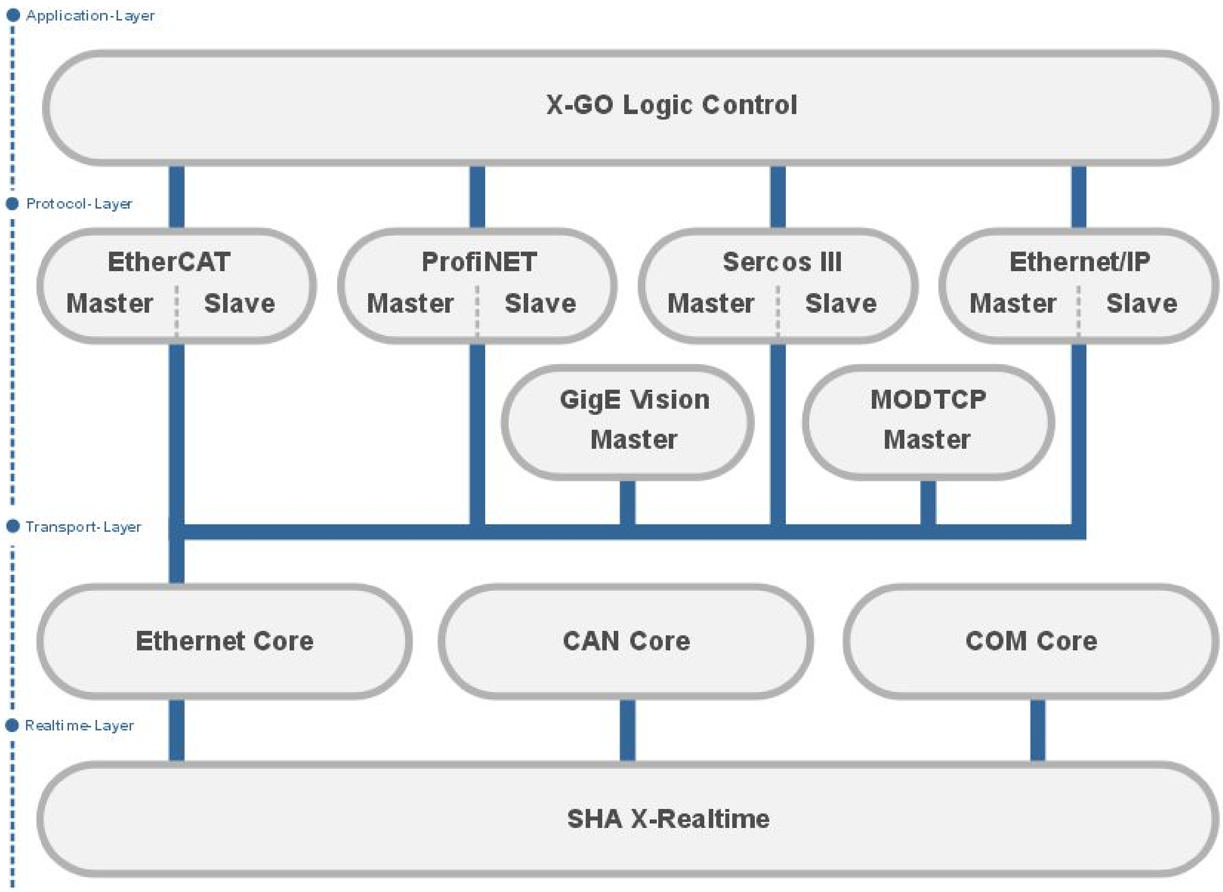
TM

ProfiNET Device Library Documentation

Date: Oct.,4.2023

1 Introduction

The idea of further interface abstruction of the SHA X-Realtime for several communication channels and bus systems, like serial communication, CANBUS, Ethernet (TCP/IP), is realized by the SYBERA AddOn Software Libraries, so called RealtimeCores. All RealtimeCores are based on the SHA X-Realtime system. The RealtimeCores are intended to fullfill Realtime-Level-1, which means collecting and buffering data in realtime without loss of data, as well as Realtime-Level-2, which means functional operation at realtime. Thus the RealtimeCores usually require simple passive harware. One of the great benefits is the adjustable scheduling time of incoming and outgoing data.



With the ProfiNET Device Stack for Windows, a complete system can be simulated in real time with a PC. The stack offers the option of reading in existing GSDML files from real devices and configuring them for the simulation. The simulated devices behave like in the real world. In combination with the "X - Realtime Engine" from SYBERA, the real - time simulation behaves like the real system. The device simulation is implemented directly from the PC with standard Ethernet adapters. The physical connection to the PLC is made using commercially available INTEL or REALTEK PCI (e) adapters. A corresponding PCMCIA or ExpressCard (PDF) adapter is also possible. No further hardware is necessary and there is no need for separate ProfiNET hardware. The basis of the programming library is the ProfiNET device stack with "X - Realtime" technology. The software runs under Windows and enables the simulation of several ProfiNET devices simultaneously in real time. Depending on the PC hardware and application, telegram update times of up to 250 µsec can be achieved.

1.1 Product Features

- Multi-Device Management
- Simulation of whole plants
- Realtime Simulation
- Update Cycles upto 250 usec
- ProfiNET Service Interface
- Alarm Handling
- Error Management
- Sequence Log
- GSDML Device Configuration
- State Management

1.2 Supported Platforms

- Visual C++ (from Version 8)
- CVI LabWindows

1.3 Supported OS

- Windows 7 - 11 (64 Bit)

1.4 Reference Devices

- HMS Anybus-S Module (T_ID_DAP)
- HMS Anybus-S Module (T_ID_ABS_PIR)
- HMS Anybus-S Module (T_ID_ABS_PRT)
- Phoenix ILB 24 DI16 DIO16 – TX2
- Phoenix FL IL 24 BK-PN-PAC
- Deutschmann Unigate CL

2 ProfinetIO Library Installation

For installation following steps are required:

Preparation

1. Provide a PC with INTEL or REALTEK Ethernet adapter and Windows operating system with administrator rights

Installation

2. Install SHA realtime system (separate software package)
3. Install ETH transport library (separate software package)
4. Run the program SYSETUP64 of the ProfiNET library
   (make sure the directory path has no space characters)

   On Installation the PEC information (PID, SERNUM and KEYCODE) must be entered. The KEYCODE for the evaluation version is: 00001111-22223333
5. Optional: Check license with SYLICENCECHECK64.EXE

Operation

6. Run PNIOVERIFY64.EXE to build a device configuration list
7. Build the program with the library interface
8. Run the program

Note: After finishing installation, you must reboot your PC before starting the compiler !!!.

Note: For proper operation, make shure within the BIOS the *INTEL Speedstep Technologie*, the *INTEL TurboBoost Technologie* as well as the *INTEL C-STATE Technologie* is turned off.

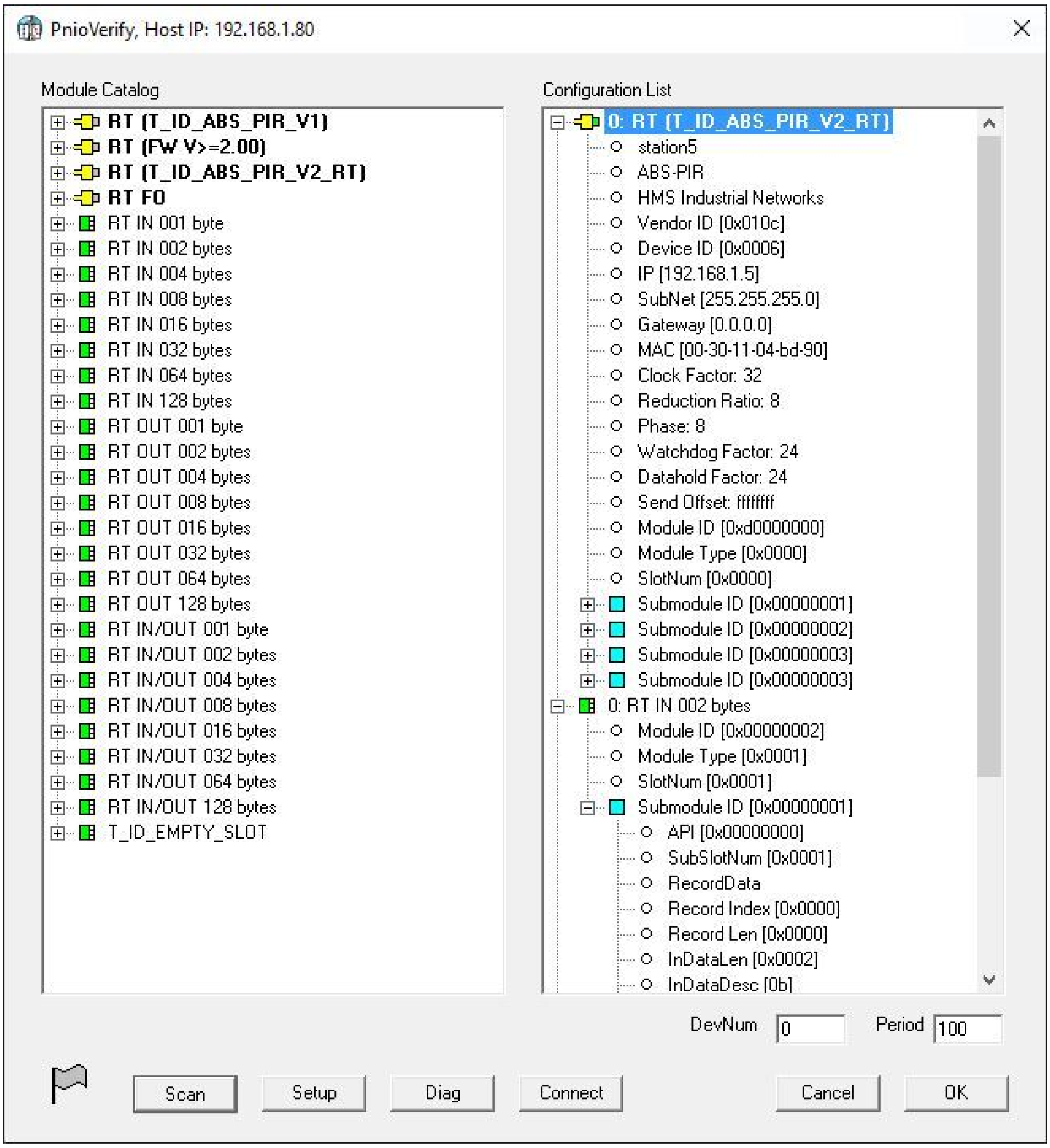
Enhanced SpeedStep — SpeedStep also modulates the CPU clock speed and voltage according to load, but it is invoked via another mechanism. The operating system must be aware of SpeedStep, as must the system BIOS, and then the OS can request frequency changes via ACPI. SpeedStep is more granular than C1E halt, because it offers multiple rungs up and down the ladder between the maximum and minimum CPU multiplier and voltage levels.

C1E enhanced halt state — Introduced in the Pentium 4 500J-series processors, the C1E halt state replaces the old C1 halt state used on the Pentium 4 and most other x86 CPUs. The C1 halt state is invoked when the operating system's idle process issues a HLT command. (Windows does this constantly when not under a full load.). C0 is the operating state. C1 (often known as Halt) is a state where the processor is not executing instructions, but can return to an executing state essentially instantaneously. All ACPI-conformant processors must support this power state. Some processors, such as the Pentium 4, also support an Enhanced C1 state (C1E or Enhanced Halt State) for lower power consumption. C2 (often known as Stop-Clock) is a state where the processor maintains all software-visible state, but may take longer to wake up. This processor state is optional. C3 (often known as Sleep) is a state where the processor does not need to keep its cache coherent, but maintains other state. Some processors have variations on the C3 state (Deep Sleep, Deeper Sleep, etc.) that differ in how long it takes to wake the processor. This processor state is optional.

Intel® Turbo Boost Technology automatically allows processor cores to run faster than the base operating frequency, increasing performance. Under some configurations and workloads, Intel® Turbo Boost technology enables higher performance through the availability of increased core frequency. Intel® Turbo Boost technology automaticallyallows processor cores to run faster than the base operating frequency if the processor is operating below rated power, temperature, and current specification limits. Intel® Turbo Boost technology can be engaged with any number of cores or logical processors enabled and active. This results in increased performance of both multi-threaded and single-threaded workloads.

3 Creating a Configuration

A ProfinetIO fieldbus system consists of several station devices (typically buscoupler devices). A station consists at least of one module (SLOT) and a module consists at least of one submodule (SUBSLOT). For proper operation the ProfinetIO devices needs first to be configured (by Station Name and IP) and a native STATIONLIST for operating the ProfiNET realtime library has to be created. Therefore SYBERA provides a program called PNIOVERIFY.EXE.



Module Catalog:

```
⊞ RT (T_ID_ABS_PIR_V1)
⊞ RT (FW V>=2.00)
⊞ RT (T_ID_ABS_PIR_V2_RT)
⊞ RT FO
⊞ RT IN 001 byte
⊞ RT IN 002 bytes
⊞ RT IN 004 bytes
⊞ RT IN 008 bytes
⊞ RT IN 016 bytes
⊞ RT IN 032 bytes
⊞ RT IN 064 bytes
⊞ RT IN 128 bytes
⊞ RT OUT 001 byte
⊞ RT OUT 002 bytes
⊞ RT OUT 004 bytes
⊞ RT OUT 008 bytes
⊞ RT OUT 016 bytes
⊞ RT OUT 032 bytes
⊞ RT OUT 064 bytes
⊞ RT OUT 128 bytes
⊞ RT IN/OUT 001 byte
⊞ RT IN/OUT 002 bytes
⊞ RT IN/OUT 004 bytes
⊞ RT IN/OUT 008 bytes
⊞ RT IN/OUT 016 bytes
⊞ RT IN/OUT 032 bytes
⊞ RT IN/OUT 064 bytes
⊞ RT IN/OUT 128 bytes
⊞ T_ID_EMPTY_SLOT
```

Configuration List:

```
⊟ 0: RT (T_ID_ABS_PIR_V2_RT)
     ○ station5
     ○ ABS-PIR
     ○ HMS Industrial Networks
     ○ Vendor ID [0x010c]
     ○ Device ID [0x0006]
     ○ IP [192.168.1.5]
     ○ SubNet [255.255.255.0]
     ○ Gateway [0.0.0.0]
     ○ MAC [00-30-11-04-bd-90]
     ○ Clock Factor: 32
     ○ Reduction Ratio: 8
     ○ Phase: 8
     ○ Watchdog Factor: 24
     ○ Datahold Factor: 24
     ○ Send Offset: ffffffff
     ○ Module ID [0xd0000000]
     ○ Module Type [0x0000]
     ○ SlotNum [0x0000]
   ⊞ Submodule ID [0x00000001]
   ⊞ Submodule ID [0x00000002]
   ⊞ Submodule ID [0x00000003]
   ⊞ Submodule ID [0x00000003]
⊟ 0: RT IN 002 bytes
     ○ Module ID [0x00000002]
     ○ Module Type [0x0001]
     ○ SlotNum [0x0001]
   ⊟ Submodule ID [0x00000001]
        ○ API [0x00000000]
        ○ SubSlotNum [0x0001]
        ○ RecordData
        ○ Record Index [0x0000]
        ○ Record Len [0x0000]
        ○ InDataLen [0x0002]
        ○ InDataDesc [0b]
```

DevNum 0   Period 100

Note: The ProfiNET device stack is able to simulate a real existing device, as well as generating a vitual device by setting up a GSDML file for it.

Note: Make shure a valid IP address is provided for the network connection.

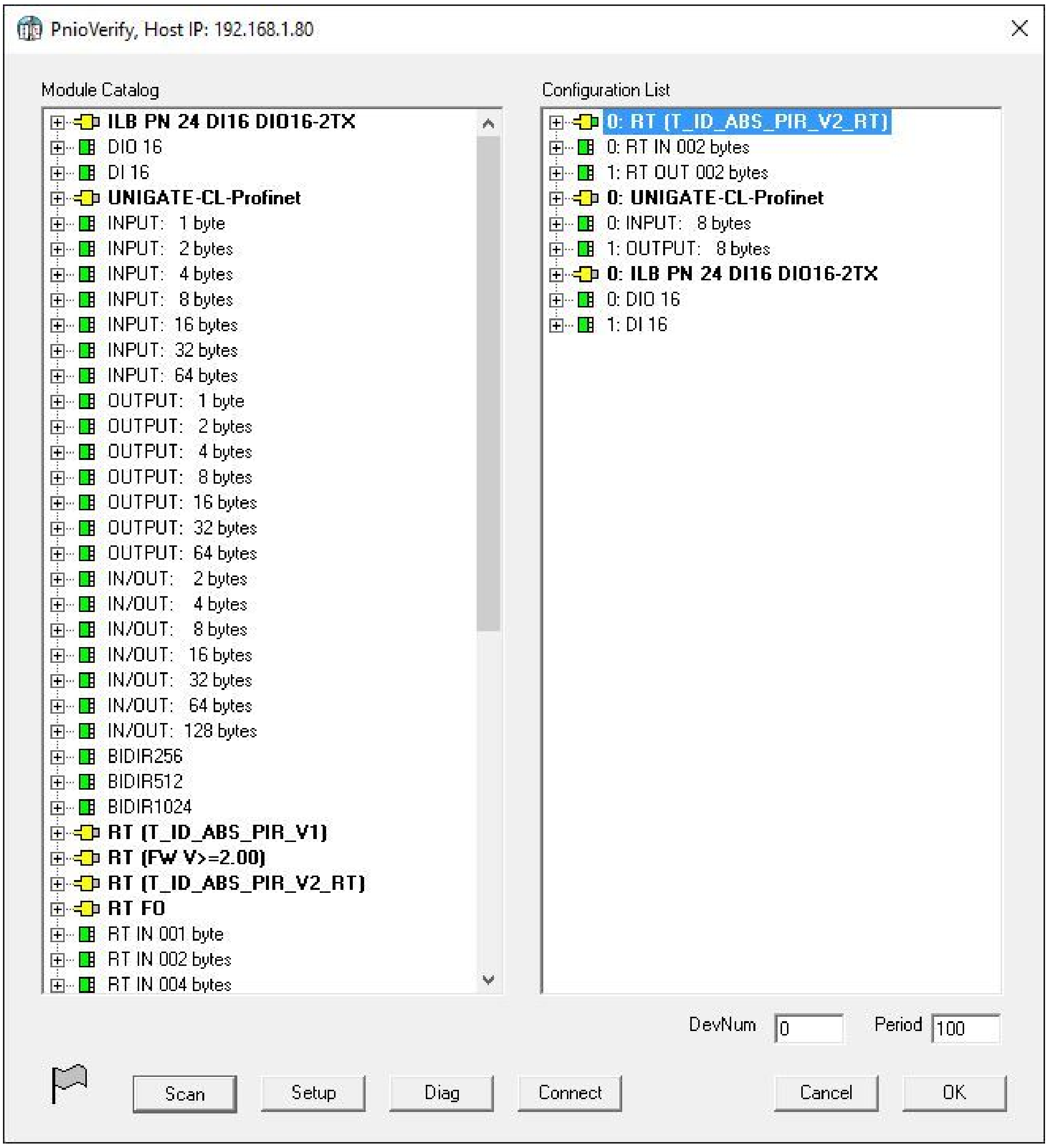
Note: If the application fails to run, check if the lastest Microsoft XML Parser has been installed. If not, install in the directory \APP\MSXML\MSXML6

PNIOVERIFY allows creating a native stationlist by selecting modules from a module catalog (leftside view). The catalog get its entries by the provides GSDML files which must be present in the same directory as PNIOVERIFY. A module is inserted to the station list configuration (rightside view) by a DRAG and DROP operation (just drag a module from the catalog to the station list configuration). There are two types of modules:

🟨	Accesspoint Module	(SLOT 0)
🟩	Functional Module	(SLOT 1 .. n)

3.1 Accesspoint Module

The accesspoint module keeps all information required for connecting to the fieldbus, as station name, IP parameters, MAC address, timing parameters. Therefore first task is to collect information about the ProfinetIO configuration by scanning the bus.



PnioVerify, Host IP: 192.168.1.80

Module Catalog

- **ILB PN 24 DI16 DIO16-2TX**
- DIO 16
- DI 16
- **UNIGATE-CL-Profinet**
- INPUT: 1 byte
- INPUT: 2 bytes
- INPUT: 4 bytes
- INPUT: 8 bytes
- INPUT: 16 bytes
- INPUT: 32 bytes
- INPUT: 64 bytes
- OUTPUT: 1 byte
- OUTPUT: 2 bytes
- OUTPUT: 4 bytes
- OUTPUT: 8 bytes
- OUTPUT: 16 bytes
- OUTPUT: 32 bytes
- OUTPUT: 64 bytes
- IN/OUT: 2 bytes
- IN/OUT: 4 bytes
- IN/OUT: 8 bytes
- IN/OUT: 16 bytes
- IN/OUT: 32 bytes
- IN/OUT: 64 bytes
- IN/OUT: 128 bytes
- BIDIR256
- BIDIR512
- BIDIR1024
- **RT (T_ID_ABS_PIR_V1)**
- **RT (FW V>=2.00)**
- **RT (T_ID_ABS_PIR_V2_RT)**
- **RT FO**
- RT IN 001 byte
- RT IN 002 bytes
- RT IN 004 bytes

Configuration List

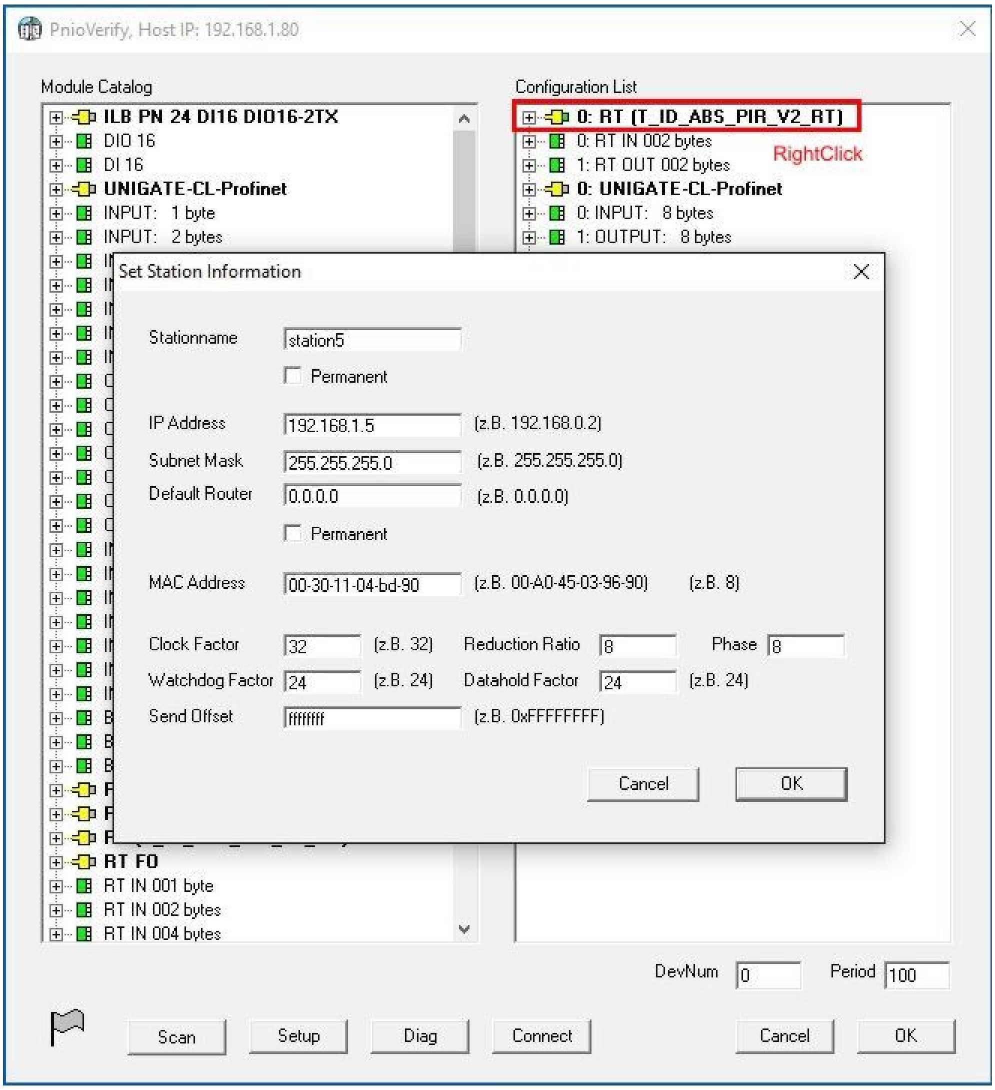
- **0: RT (T_ID_ABS_PIR_V2_RT)**
- 0: RT IN 002 bytes
- 1: RT OUT 002 bytes
- **0: UNIGATE-CL-Profinet**
- 0: INPUT: 8 bytes
- 1: OUTPUT: 8 bytes
- **0: ILB PN 24 DI16 DIO16-2TX**
- 0: DIO 16
- 1: DI 16

DevNum: 0

Period: 100

3.2 Station Settings

The scan gets information about manufacturer name and MAC address. Now individual assignment must set (e.g. IP address, station name, timings). On a right button click at the accesspoint module a dialog appears, which allows setting of station name , IP and timing parameters.



Stationname: station5

□ Permanent

IP Address: 192.168.1.5 (z.B. 192.168.0.2)

Subnet Mask: 255.255.255.0 (z.B. 255.255.255.0)

Default Router: 0.0.0.0 (z.B. 0.0.0.0)

□ Permanent

MAC Address: 00-30-11-04-bd-90 (z.B. 00-A0-45-03-96-90) (z.B. 8)

Clock Factor: 32 (z.B. 32) Reduction Ratio: 8 Phase: 8

Watchdog Factor: 24 (z.B. 24) Datahold Factor: 24 (z.B. 24)

Send Offset: ffffffff (z.B. 0xFFFFFFFF)

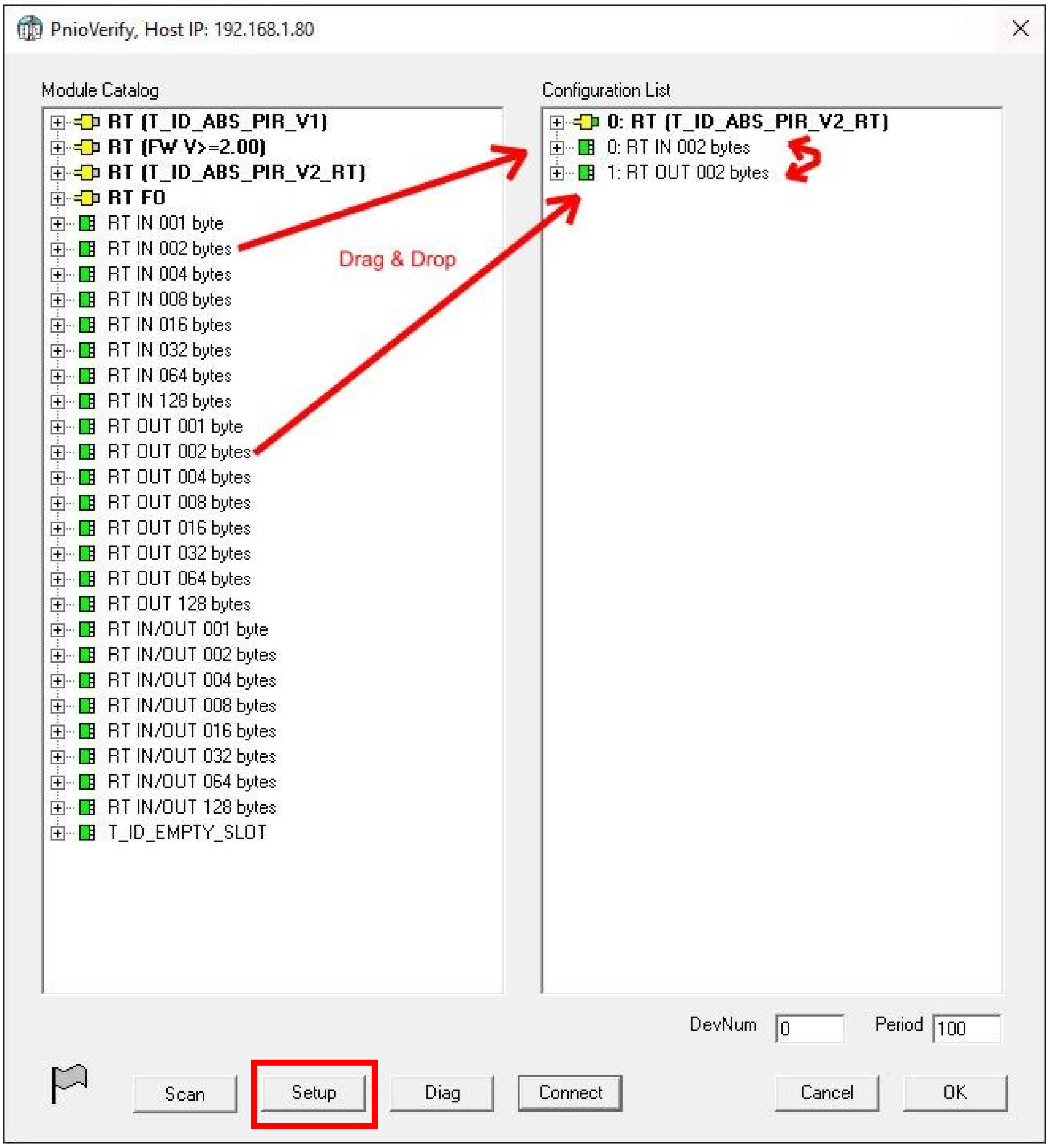
DevNum: 0 Period: 100

<u>Note:</u>

The timing settings of each station are based on the master settings. The Clock Factor and the Reduction Ratio have no meaning.

3.3 Functional Module

Each station typically consists of multiple functional modules (SLOT 1..n). Function Modules have to be inserted from the catalog by DRAG and DROP operations. As well the nmodules may be sorted below the AccessPoint. A station configuration should contain all functional modules (in the order these modules are physically connected). When inserting a new module from the catalog, after dropping, it appears at the end of the configuration list and may be pushed to the correct slot location.



PnioVerify, Host IP: 192.168.1.80

Module Catalog

- RT (T_ID_ABS_PIR_V1)
- RT (FW V>=2.00)
- RT (T_ID_ABS_PIR_V2_RT)
- RT F0
- RT IN 001 byte
- RT IN 002 bytes
- RT IN 004 bytes
- RT IN 008 bytes
- RT IN 016 bytes
- RT IN 032 bytes
- RT IN 064 bytes
- RT IN 128 bytes
- RT OUT 001 byte
- RT OUT 002 bytes
- RT OUT 004 bytes
- RT OUT 008 bytes
- RT OUT 016 bytes
- RT OUT 032 bytes
- RT OUT 064 bytes
- RT OUT 128 bytes
- RT IN/OUT 001 byte
- RT IN/OUT 002 bytes
- RT IN/OUT 004 bytes
- RT IN/OUT 008 bytes
- RT IN/OUT 016 bytes
- RT IN/OUT 032 bytes
- RT IN/OUT 064 bytes
- RT IN/OUT 128 bytes
- T_ID_EMPTY_SLOT

Configuration List

- 0: RT (T_ID_ABS_PIR_V2_RT)
- 0: RT IN 002 bytes
- 1: RT OUT 002 bytes

DevNum: 0

Period: 100

When the settings are done, the station my be initialized by pressing the button [Setup]

The resulting stationlist is stored to a choosen text file (sample):

Sample:

```
> Station

[NAME]
station8
[MFG_NAME]
ABS-PIR
[VENDOR_NAME]
HMS Industrial Networks
[STATION_ID]
0c 01 06 00 01 00
[MAC_ADDR]
00 30 11 04 bd 90
[IP_PARAMS]
c0 a8 01 08 ff ff ff 00 00 00 00 00
[TI_PARAMS]
20 00 08 00 08 00 18 00 18 00 ff ff ff ff 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                                                                                                00

>> Module

[NAME]
T_ID_ABS_PIR_V2_RT
[MOD_ID]
00 00 00 d0
[MOD_TYPE]
00 00
[SLOT_NUM]
00 00

>>> Submodule

[SUBMOD_ID]
01 00 00 00
[SUBSLOT_NUM]
01 00
[OBJ_INPUT]
01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                                                               00 00 00 00 01 00 00 00 01 00 00 00

>>> Submodule

[SUBMOD_ID]
02 00 00 00
[SUBSLOT_NUM]
00 80
[OBJ_INPUT]
01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                                                               00 00 00 00 01 00 00 00 01 00 00 00

>>> Submodule

[SUBMOD_ID]
03 00 00 00
[SUBSLOT_NUM]
01 80
[OBJ_INPUT]
01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                                                               00 00 00 00 01 00 00 00 01 00 00 00

>>> Submodule

[SUBMOD_ID]
03 00 00 00
[SUBSLOT_NUM]
02 80
```

```
[OBJ_INPUT]
01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                                                              00 00 00 00 01 00 00 00 01 00 00 00

>> Module

[NAME]
T_ID_RT_IN2
[MOD_ID]
02 00 00 00
[MOD_TYPE]
01 00
[SLOT_NUM]
01 00

>>> Submodule

[SUBMOD_ID]
01 00 00 00
[SUBSLOT_NUM]
01 00
[OBJ_INPUT]
01 0b 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                                                              02 00 00 00 01 00 00 00 01 00 00 00

>> Module

[NAME]
T_ID_RT_OUT2
[MOD_ID]
20 00 00 00
[MOD_TYPE]
01 00
[SLOT_NUM]
02 00

>>> Submodule

[SUBMOD_ID]
01 00 00 00
[SUBSLOT_NUM]
01 00
[OBJ_OUTPUT]
01 0b 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                                                               02 00 00 00 01 00 00 00 01 00 00 00
```

3.4 Data Offsets

The order of the modules below an access point module determines the offset of the payload data within the Ethernet frame. The payload data includes not only the process data itself, but also one status byte per submodule. Here a sample consisting of an access point module, an input module (64 byte data) and an output module (64 byte data):

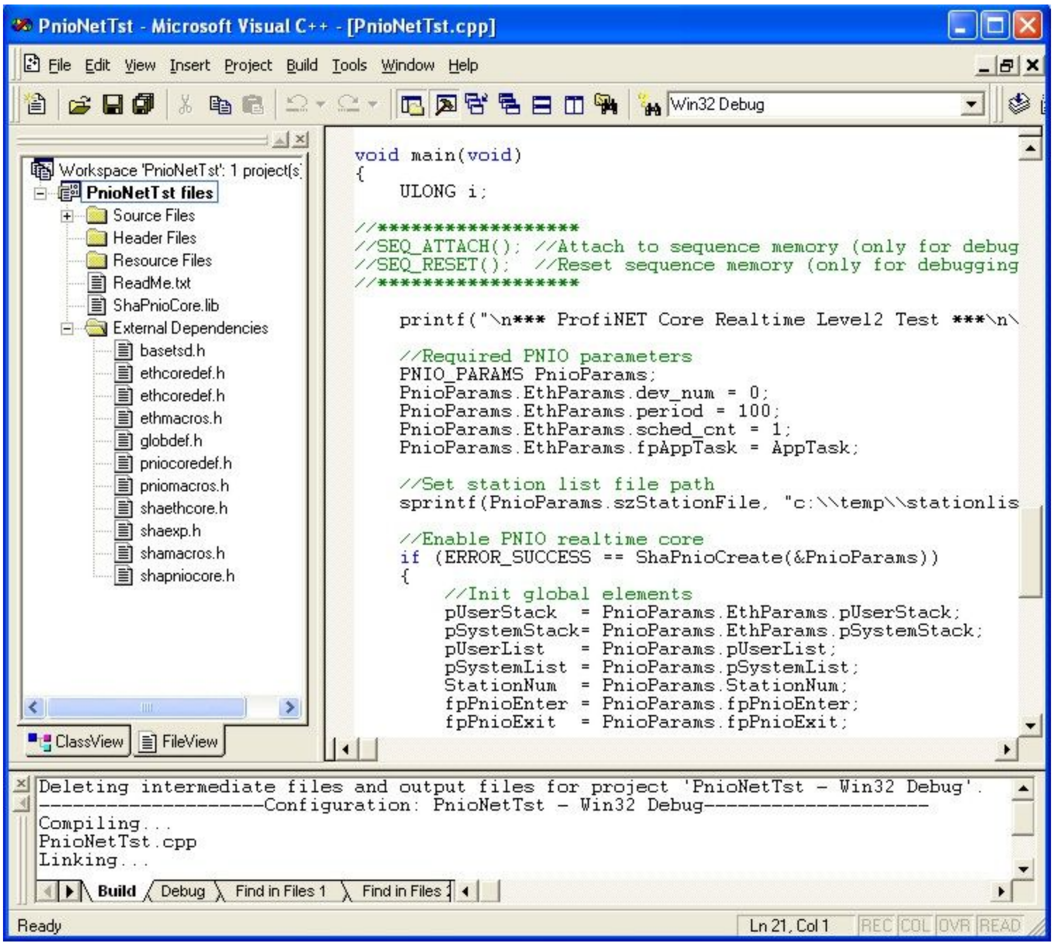
AccessPoint ModuleID : 0x1234	**Controller Output Frame**			**Controller Input Frame**		
		Bytes	Offset		Bytes	Offset
SubModuleID : 1	Status	1	0	Status	1	0
	Data	0		Data	0	
SubModuleID : 1	Status	1	1	Status	1	1
	Data	0		Data	0	
SubModuleID : 1	Status	1	2	Status	1	2
	Data	0		Data	0	
SubModuleID : 1	Status	1	3	Status	1	3
	Data	0		Data	0	
Input 64 Bytes ModuleID : 0x2064						
SubModuleID : 1	Status	1	4	Status	1	4
	Data	0		Data	64	5
Output 64 Bytes ModuleID : 0x1064						
SubModuleID : 1	Status	1	5	Status	1	69
	Data	64	6	Data	0	

4 ProfinetIO Realtime Device Library

The interface functions of the ProfiNET Realtime Device Library are exported by a dynamic link library. Following header files and libraries are required:

SHA*64*PNIODEVICE.DLL	ProfiNET Device DLL (VISUAL C++)
SHA*64*PNIODEVICE.LIB	ProfiNET Device LIB (VISUAL C++)
SHA*64*PNIODEVICE.H	Exported Function Prototypes
PNIO*64*COREDEF.H	ProfiNET Basic Definitions
PNIO*64*MACROS.H	ProfiNET Macro Definitions
PNIO*64*CONTROL.H	ProfiNET Ethernet control Macro
STATIONLIST.PAR	Native Station List (generated by PNIOVERIFY64)
PNTDBG.LOG	Sequence Log (generated at runtime)

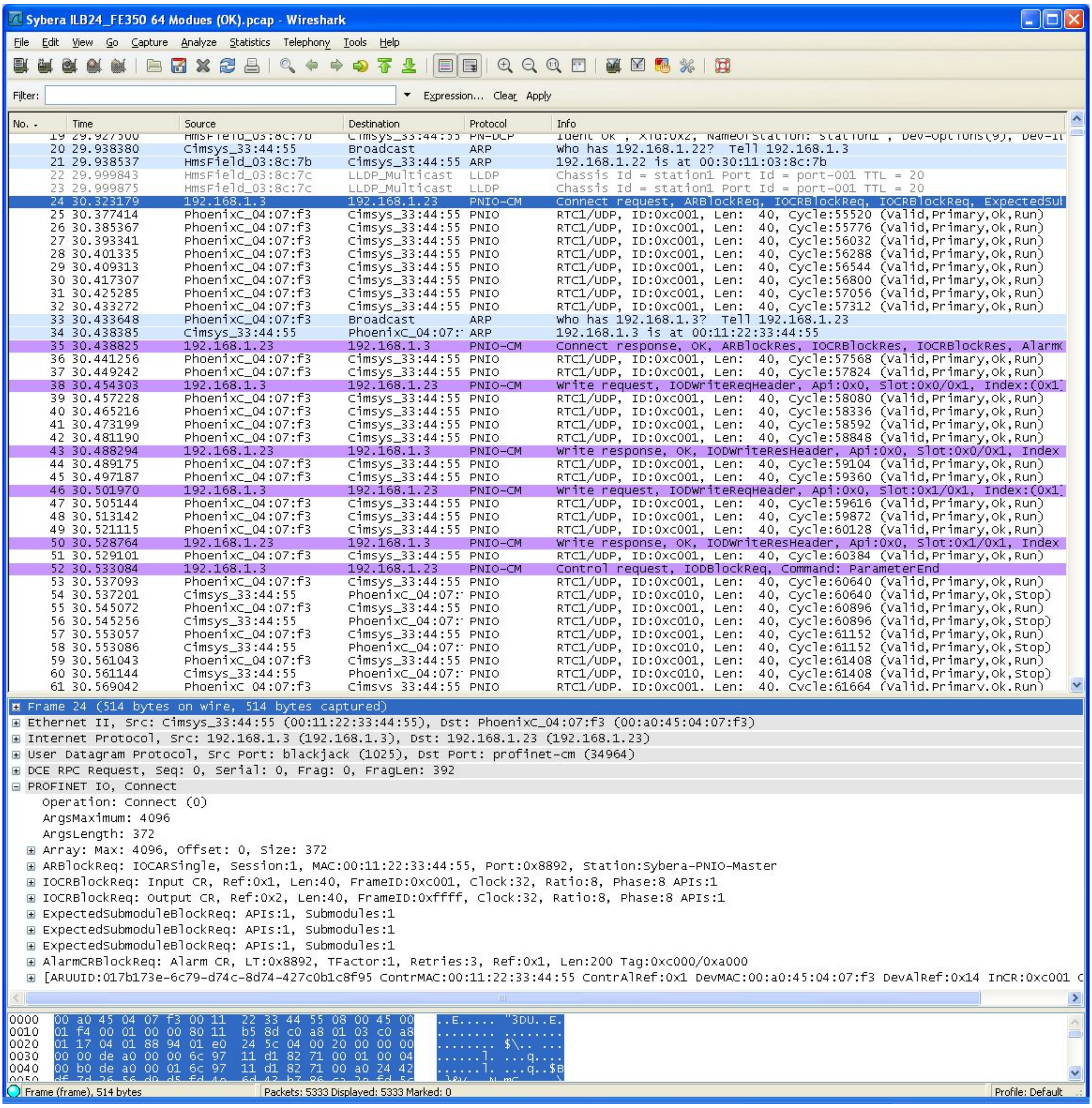
Sample Project



```
void main(void)
{
    ULONG i;

//*******************
//SEQ_ATTACH(); //Attach to sequence memory (only for debug
//SEQ_RESET();  //Reset sequence memory (only for debugging
//*******************

    printf("\n*** ProfiNET Core Realtime Level2 Test ***\n\

    //Required PNIO parameters
    PNIO_PARAMS PnioParams;
    PnioParams.EthParams.dev_num = 0;
    PnioParams.EthParams.period = 100;
    PnioParams.EthParams.sched_cnt = 1;
    PnioParams.EthParams.fpAppTask = AppTask;

    //Set station list file path
    sprintf(PnioParams.szStationFile, "c:\\temp\\stationlis

    //Enable PNIO realtime core
    if (ERROR_SUCCESS == ShaPnioCreate(&PnioParams))
    {
        //Init global elements
        pUserStack  = PnioParams.EthParams.pUserStack;
        pSystemStack= PnioParams.EthParams.pSystemStack;
        pUserList   = PnioParams.pUserList;
        pSystemList = PnioParams.pSystemList;
        StationNum  = PnioParams.StationNum;
        fpPnioEnter = PnioParams.fpPnioEnter;
        fpPnioExit  = PnioParams.fpPnioExit;
```

```
Deleting intermediate files and output files for project 'PnioNetTst - Win32 Debug'.
--------------------Configuration: PnioNetTst - Win32 Debug--------------------
Compiling...
PnioNetTst.cpp
Linking...
```

Sample Startup Protocoll:


Sybera ILB24_FE350 64 Modues (OK).pcap - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter:
Expression... Clear Apply

```
No. .  Time        Source             Destination        Protocol  Info
   19 29.927500   HmsField_03:8c:7b  Cimsys_33:44:55    PN-DCP    Ident OK , Xid:0x2, NameOfStation: station1 , Dev-Options(9), Dev-I
   20 29.938380   Cimsys_33:44:55    Broadcast          ARP       Who has 192.168.1.22?  Tell 192.168.1.3
   21 29.938537   HmsField_03:8c:7b  Cimsys_33:44:55    ARP       192.168.1.22 is at 00:30:11:03:8c:7b
   22 29.999843   HmsField_03:8c:7c  LLDP_Multicast     LLDP      Chassis Id = station1 Port Id = port-001 TTL = 20
   23 29.999875   HmsField_03:8c:7c  LLDP_Multicast     LLDP      Chassis Id = station1 Port Id = port-001 TTL = 20
   24 30.323179   192.168.1.3        192.168.1.23       PNIO-CM   Connect request, ARBlockReq, IOCRBlockReq, IOCRBlockReq, ExpectedSul
   25 30.377414   PhoenixC_04:07:f3  Cimsys_33:44:55    PNIO      RTC1/UDP, ID:0xc001, Len:  40, Cycle:55520 (Valid,Primary,Ok,Run)
   26 30.385367   PhoenixC_04:07:f3  Cimsys_33:44:55    PNIO      RTC1/UDP, ID:0xc001, Len:  40, Cycle:55776 (Valid,Primary,Ok,Run)
   27 30.393341   PhoenixC_04:07:f3  Cimsys_33:44:55    PNIO      RTC1/UDP, ID:0xc001, Len:  40, Cycle:56032 (Valid,Primary,Ok,Run)
   28 30.401335   PhoenixC_04:07:f3  Cimsys_33:44:55    PNIO      RTC1/UDP, ID:0xc001, Len:  40, Cycle:56288 (Valid,Primary,Ok,Run)
   29 30.409313   PhoenixC_04:07:f3  Cimsys_33:44:55    PNIO      RTC1/UDP, ID:0xc001, Len:  40, Cycle:56544 (Valid,Primary,Ok,Run)
   30 30.417307   PhoenixC_04:07:f3  Cimsys_33:44:55    PNIO      RTC1/UDP, ID:0xc001, Len:  40, Cycle:56800 (Valid,Primary,Ok,Run)
   31 30.425285   PhoenixC_04:07:f3  Cimsys_33:44:55    PNIO      RTC1/UDP, ID:0xc001, Len:  40, Cycle:57056 (Valid,Primary,Ok,Run)
   32 30.433272   PhoenixC_04:07:f3  Cimsys_33:44:55    PNIO      RTC1/UDP, ID:0xc001, Len:  40, Cycle:57312 (Valid,Primary,Ok,Run)
   33 30.433648   PhoenixC_04:07:f3  Broadcast          ARP       Who has 192.168.1.3?  Tell 192.168.1.23
   34 30.438385   Cimsys_33:44:55    PhoenixC_04:07:    ARP       192.168.1.3 is at 00:11:22:33:44:55
   35 30.438825   192.168.1.23       192.168.1.3        PNIO-CM   Connect response, OK, ARBlockRes, IOCRBlockRes, IOCRBlockRes, AlarmC
   36 30.441256   PhoenixC_04:07:f3  Cimsys_33:44:55    PNIO      RTC1/UDP, ID:0xc001, Len:  40, Cycle:57568 (Valid,Primary,Ok,Run)
   37 30.449242   PhoenixC_04:07:f3  Cimsys_33:44:55    PNIO      RTC1/UDP, ID:0xc001, Len:  40, Cycle:57824 (Valid,Primary,Ok,Run)
   38 30.454303   192.168.1.3        192.168.1.23       PNIO-CM   Write request, IODWriteReqHeader, Api:0x0, Slot:0x0/0x1, Index:(0x1
   39 30.457228   PhoenixC_04:07:f3  Cimsys_33:44:55    PNIO      RTC1/UDP, ID:0xc001, Len:  40, Cycle:58080 (Valid,Primary,Ok,Run)
   40 30.465216   PhoenixC_04:07:f3  Cimsys_33:44:55    PNIO      RTC1/UDP, ID:0xc001, Len:  40, Cycle:58336 (Valid,Primary,Ok,Run)
   41 30.473199   PhoenixC_04:07:f3  Cimsys_33:44:55    PNIO      RTC1/UDP, ID:0xc001, Len:  40, Cycle:58592 (Valid,Primary,Ok,Run)
   42 30.481190   PhoenixC_04:07:f3  Cimsys_33:44:55    PNIO      RTC1/UDP, ID:0xc001, Len:  40, Cycle:58848 (Valid,Primary,Ok,Run)
   43 30.488294   192.168.1.23       192.168.1.3        PNIO-CM   Write response, OK, IODWriteResHeader, Api:0x0, Slot:0x0/0x1, Index
   44 30.489175   PhoenixC_04:07:f3  Cimsys_33:44:55    PNIO      RTC1/UDP, ID:0xc001, Len:  40, Cycle:59104 (Valid,Primary,Ok,Run)
   45 30.497187   PhoenixC_04:07:f3  Cimsys_33:44:55    PNIO      RTC1/UDP, ID:0xc001, Len:  40, Cycle:59360 (Valid,Primary,Ok,Run)
   46 30.501970   192.168.1.3        192.168.1.23       PNIO-CM   Write request, IODWriteReqHeader, Api:0x0, Slot:0x1/0x1, Index:(0x1
   47 30.505144   PhoenixC_04:07:f3  Cimsys_33:44:55    PNIO      RTC1/UDP, ID:0xc001, Len:  40, Cycle:59616 (Valid,Primary,Ok,Run)
   48 30.513142   PhoenixC_04:07:f3  Cimsys_33:44:55    PNIO      RTC1/UDP, ID:0xc001, Len:  40, Cycle:59872 (Valid,Primary,Ok,Run)
   49 30.521115   PhoenixC_04:07:f3  Cimsys_33:44:55    PNIO      RTC1/UDP, ID:0xc001, Len:  40, Cycle:60128 (Valid,Primary,Ok,Run)
   50 30.528764   192.168.1.23       192.168.1.3        PNIO-CM   Write response, OK, IODWriteResHeader, Api:0x0, Slot:0x1/0x1, Index
   51 30.529101   PhoenixC_04:07:f3  Cimsys_33:44:55    PNIO      RTC1/UDP, ID:0xc001, Len:  40, Cycle:60384 (Valid,Primary,Ok,Run)
   52 30.533084   192.168.1.3        192.168.1.23       PNIO-CM   Control request, IODBlockReq, Command: ParameterEnd
   53 30.537093   PhoenixC_04:07:f3  Cimsys_33:44:55    PNIO      RTC1/UDP, ID:0xc001, Len:  40, Cycle:60640 (Valid,Primary,Ok,Run)
   54 30.537201   Cimsys_33:44:55    PhoenixC_04:07:    PNIO      RTC1/UDP, ID:0xc010, Len:  40, Cycle:60640 (Valid,Primary,Ok,Stop)
   55 30.545072   PhoenixC_04:07:f3  Cimsys_33:44:55    PNIO      RTC1/UDP, ID:0xc001, Len:  40, Cycle:60896 (Valid,Primary,Ok,Run)
   56 30.545256   Cimsys_33:44:55    PhoenixC_04:07:    PNIO      RTC1/UDP, ID:0xc010, Len:  40, Cycle:60896 (Valid,Primary,Ok,Stop)
   57 30.553057   PhoenixC_04:07:f3  Cimsys_33:44:55    PNIO      RTC1/UDP, ID:0xc001, Len:  40, Cycle:61152 (Valid,Primary,Ok,Run)
   58 30.553086   Cimsys_33:44:55    PhoenixC_04:07:    PNIO      RTC1/UDP, ID:0xc010, Len:  40, Cycle:61152 (Valid,Primary,Ok,Stop)
   59 30.561043   PhoenixC_04:07:f3  Cimsys_33:44:55    PNIO      RTC1/UDP, ID:0xc001, Len:  40, Cycle:61408 (Valid,Primary,Ok,Run)
   60 30.561144   Cimsys_33:44:55    PhoenixC_04:07:    PNIO      RTC1/UDP, ID:0xc010, Len:  40, Cycle:61408 (Valid,Primary,Ok,Stop)
   61 30.569042   PhoenixC_04:07:f3  Cimsys_33:44:55    PNIO      RTC1/UDP, ID:0xc001, Len:  40, Cycle:61664 (Valid,Primary,Ok,Run)
```

```
⊞ Frame 24 (514 bytes on wire, 514 bytes captured)
⊞ Ethernet II, Src: Cimsys_33:44:55 (00:11:22:33:44:55), Dst: PhoenixC_04:07:f3 (00:a0:45:04:07:f3)
⊞ Internet Protocol, Src: 192.168.1.3 (192.168.1.3), Dst: 192.168.1.23 (192.168.1.23)
⊞ User Datagram Protocol, Src Port: blackjack (1025), Dst Port: profinet-cm (34964)
⊞ DCE RPC Request, Seq: 0, Serial: 0, Frag: 0, FragLen: 392
⊟ PROFINET IO, Connect
    Operation: Connect (0)
    ArgsMaximum: 4096
    ArgsLength: 372
  ⊞ Array: Max: 4096, Offset: 0, Size: 372
  ⊞ ARBlockReq: IOCARSingle, Session:1, MAC:00:11:22:33:44:55, Port:0x8892, Station:Sybera-PNIO-Master
  ⊞ IOCRBlockReq: Input CR, Ref:0x1, Len:40, FrameID:0xc001, Clock:32, Ratio:8, Phase:8 APIs:1
  ⊞ IOCRBlockReq: Output CR, Ref:0x2, Len:40, FrameID:0xffff, Clock:32, Ratio:8, Phase:8 APIs:1
  ⊞ ExpectedSubmoduleBlockReq: APIs:1, Submodules:1
  ⊞ ExpectedSubmoduleBlockReq: APIs:1, Submodules:1
  ⊞ ExpectedSubmoduleBlockReq: APIs:1, Submodules:1
  ⊞ AlarmCRBlockReq: Alarm CR, LT:0x8892, TFactor:1, Retries:3, Ref:0x1, Len:200 Tag:0xc000/0xa000
  ⊞ [ARUUID:017b173e-6c79-d74c-8d74-427c0b1c8f95 ContrMAC:00:11:22:33:44:55 ContrAlRef:0x1 DevMAC:00:a0:45:04:07:f3 DevAlRef:0x14 InCR:0xc001 C
```

```
0000  00 a0 45 04 07 f3 00 11  22 33 44 55 08 00 45 00   ..E..... "3DU..E.
0010  01 f4 00 01 00 00 80 11  b5 8d c0 a8 01 03 c0 a8   ........ ........
0020  01 17 04 01 88 94 01 e0  24 5c 04 00 20 00 00 00   ........ $\.. ...
0030  00 00 de a0 00 00 6c 97  11 d1 82 71 00 01 00 04   ......l. ...q....
0040  00 b0 de a0 00 01 6c 97  11 d1 82 71 00 a0 24 42   ......l. ...q..$B
0050  df 7d 26 56 d0 d5 fd 4e  6d 43 b7 86 c2 2e fd 5c
```

Frame (frame), 514 bytes | Packets: 5333 Displayed: 5333 Marked: 0 | Profile: Default

4.1 Header File PNIO*64*COREDEF.H

The header file PNIODEVICEDEF.H declares all required structures when handling ProfinetIO interface functions or handling the Core Realtime Stack directly (Realtime Level2).

4.1.1 Structure PNIO_PARAMS

This structure is required by the core interface functions, and contains all required and optional input and output data members.

```
typedef struct _PNIO_PARAMS
{
      //Input parameters
      char              szStationFile[MAX_PATH_SIZE]; //Station list file name

      //Output parameters
      ULONG             ErrCnts;          //Error Counters
      FP_PNIO_ENTER     fpPnioEnter;      //Function Pointer to PnioEnter()
      FP_PNIO_EXIT      fpPnioExit;       //Function Pointer to PnioExit()
      ULONG             core_dll_ver;     //Core DLL version
      ULONG             core_drv_ver;     //Core driver version

      //Input - Output parameters
      ETH_PARAMS        EthParams;        //Ethernet Core Parameters

      //Realtime level2 parameters
      ULONG             StationNum;       //Station Number
      PSTATION_INFO     pSystemList;      //PSTATION_INFO structure for realtime
                                          //application task
      PSTATION_INFO     pUserList;        //PSTATION_INFO structure for windows
                                          //application task
} PNIO_PARAMS, *PPNIO_PARAMS;
```

Note:

The structure ETH_PARAMS is part of the Ethernet Core Library and described in the the documentation of this core libraray. Thus the Ethernet Core library must be installed first. The required elements of the structure ETH_PARAMS must be used in the same way as using the Ethernet realtime core.

4.1.2 Structure STATION_INFO

This structure keeps all information of each ProfinetIO modul and may be required for further interface functions.

```
typedef struct _STATION_INFO
{
      STATION_HDR       Hdr;                        //Station Header
      ULONG             State;                      //Station State
      ULONG             Event;                      //Station Event
      ULONG             Error;                      //Station Error (PNIO Status)
      ULONG             ModNum;                     //Module Number
      MODULE_INFO       ModList[MAX_MODULE_NUM];          //Module List
      FRAME_INFO        FrameInfo[MAX_OBJ_DIR];           //Frame information
      ALARM_INFO        AlarmInfo;                        //Alarm information
      USHORT            SessionKey;                       //Session Key  (AR-Block)
      GUID              SessionUuid;                      //Session UUID (AR-Block)
      GUID              InitiatorUuid;                    //Initiator UUID (AR-Block)
      ULONG             Reserved[5];                      //Reserved

} STATION_INFO, *PSTATION_INFO;
```

Note:

The most elements of the structure STATION_INFO will be automatically filled with the provided Stationlist information. The elements InputFrameData and OutputFrameData keep the payload data of the station.

Sample:

```
PUCHAR pInData;
PUCHAR pOutData;

//Get input  data from Slot3 (IB IL 24 DI 4-ME)
//Set output data to   Slot2 (IB IL 24 DO 4-ME)
PNIO_GET_INPUT_DATAPTR (pStation, 3, 0, &pInData);
PNIO_GET_OUTPUT_DATAPTR(pStation, 2, 0, &pOutData);
```

4.2 Header File PNIO*64*MACROS.H

This header file defines all macros required for handling the realtime Task

//Macro to check PNIO frame ID

```
#define PNIO_CHECK_FRAMEID(__pFrame, __pID, __pbVlan)
```

//Macro to set PNIO frame ID

```
#define PNIO_SET_FRAMEID(__pFrame, __id, __bVlan)
```

//Macro to get PNIO input data pointer

```
#define PNIO_GET_INPUT_DATAPTR(__pStation, __ModIndex, __SubModIndex, __ppData)
```

//Macro to get PNIO output data pointer

```
#define PNIO_GET_OUTPUT_DATAPTR(__pStation, __ModIndex, __SubModIndex, __ppData)
```

//Macro to compare station ID

```
#define PNIO_CHECK_STATION_ID(__pStation, __VendorID, __DeviceID, __NodeID)
```

//Macro to compare station MAC

```
#define PNIO_CHECK_STATION_MAC(__pStation, __pMacAddr)
```

//Use inline function to compare station Name

__inline BOOLEAN __PnioCheckStationName(PSTATION_INFO pStation, char* szName)

<u>Sample:</u>

```
//Check for station 1 (FL IL 24 BK-PN-PAC)
if (__PnioCheckStationName(pStation, "station1"))
{
 //Get input  data from Slot3 (IB IL 24 DI 4-ME)
 //Set output data to   Slot2 (IB IL 24 DO 4-ME)
 PNIO_GET_INPUT_DATAPTR (pStation, 3, 0, &pInData);
 PNIO_GET_OUTPUT_DATAPTR(pStation, 2, 0, &pOutData);
 if ((pInData) &&
     (pOutData))
 {
       //Set input data to output data
       *pOutData = *pInData;
 }
}
```

4.3 Debug Log File

The ProfiNET Device library provides a buildin log sytem which produces a debug log file called *PNTDBG.LOG*. This file contains all nessecary information of the library sequence.

Sample:

```
PNIODEVICE -> ShaPnioCreate

PNIODEVICE -> GetHostAddress

PNIODEVICE -> DrvCreate

PNIODEVICE -> DriverOpen

PNIODEVICE -> CreatePnioThread

PNIODEVICE -> CreateStationList
C:\XGO\StationList.par

PNIODEVICE -> LoadStationList
C:\XGO\StationList.par

PNIODEVICE -> PnioSetFromStationFile
C:\XGO\StationList.par

PNIODEVICE -> PnioSetStationName
station2
Permanent Flag: 1

PNIODEVICE -> DcpCmd
ServiceType: 0x00000000
ServiceID: 0x00000005

PNIODEVICE -> WaitForDcp
ServiceType: 0x00000001
ServiceID: 0x00000005
*** Frame Received ***

PNIODEVICE -> PnioSetStationIP IP:192.168.1.23
Permanent Flag: 0

PNIODEVICE -> ArpCmd IP:192.168.1.23
OpCode: 0x00000001

PNIODEVICE -> WaitForArp IP:192.168.1.23
OpCode: 0x00000002
*** Frame Received ***

PNIODEVICE -> PnioSetStationName
station1
Permanent Flag: 1

PNIODEVICE -> DcpCmd
```

```
ServiceType: 0x00000000
ServiceID: 0x00000005

PNIODEVICE -> WaitForDcp
ServiceType: 0x00000001
ServiceID: 0x00000005
*** Frame Received ***

PNIODEVICE -> PnioSetStationIP IP:192.168.1.22
Permanent Flag: 0

PNIODEVICE -> ArpCmd IP:192.168.1.22
OpCode: 0x00000001

PNIODEVICE -> WaitForArp IP:192.168.1.22
OpCode: 0x00000002
*** Frame Received ***

PNIODEVICE -> ShaPnioGetVersion

PNIODEVICE -> DrvGetVersion

PNIODEVICE -> PnioEnableStation
station2

PNIODEVICE -> PnioServiceConnect
Output IOCS  , Mod:0, SubMod:0, Offs:0  , Len:1
Output IOCS  , Mod:1, SubMod:0, Offs:1  , Len:1
Output IOCS  , Mod:2, SubMod:0, Offs:2  , Len:1
Input  IOCS  , Mod:1, SubMod:0, Offs:0  , Len:1
Input  IODATA, Mod:0, SubMod:0, Offs:1  , Len:0
Input  IOPS  , Mod:0, SubMod:0, Offs:1  , Len:1
Input  IODATA, Mod:1, SubMod:0, Offs:2  , Len:2
Input  IOPS  , Mod:1, SubMod:0, Offs:4  , Len:1
Input  IODATA, Mod:2, SubMod:0, Offs:5  , Len:2
Input  IOPS  , Mod:2, SubMod:0, Offs:7  , Len:1
Output IODATA, Mod:1, SubMod:0, Offs:3  , Len:2
Output IOPS  , Mod:1, SubMod:0, Offs:5  , Len:1
ModID:0x00000000, SubModID:0x00000001, Prop:0x0000
ModID:0x00000001, SubModID:0x00000001, Prop:0x000b
ModID:0x00000002, SubModID:0x00000001, Prop:0x0001

PNIODEVICE -> RpcCmd IP:192.168.1.23
PacketType: 0x00000000

PNIODEVICE -> WaitForRcp IP:192.168.1.3
PacketType: 0x00000002
*** Frame Received ***

PNIODEVICE -> PnioServiceWrite
SlotNum:    0
SubSlotNum: 1
RecordIndex: 1
RecordLen:   2

…
```

5 ProfinetIO Library Interface

The header file SHAPNIODEVICE.H defines all required prototypes and parameters of the Ethernet Core Library. The header file is based on the files RAWCOREDEF.H and ETHCOREDEF.H. In the following all function prototypes will be discussed by samples. Since all platforms have their own syntax and dependencies, therefore the topics for the different platforms are marked as follow:

VC : Visual C++, Borland C++ Builder and CVI Lab Windows

5.1 Basic Functions

5.1.1 Sha*64*PnioCreate

This function initializes the ProfinetIO module states. On success the returning value is ERROR_SUCCESS, otherwise the returning value corresponds to that with GetLastError().

VC
```
ULONG Sha64PnioCreate (PPNIO_PARAMS);
```

5.1.2 Sha*64*PnioDestroy

This function closes the ProfinetIO communication.

VC
```
ULONG Sha64PnioDestroy(PPNIO_PARAMS);
```

5.1.3 Sha*64*PnioGetVersion

This function retrieves the version information of the ProfinetIO Device Library, the Ethernet Core Library, the Ethernet Core Driver, the SHA Dll, the SHA Library and the SHA Driver.

VC
```
ULONG Sha64PnioGetVersion(PPNIO_PARAMS);
```

Sample:

```
#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include "c:\eth\ShaEthCore.h"
#include "c:\pnt\ShaPnioCore.h"
#include "c:\sha\shaexp.h"


PETH_STACK         pUserStack = NULL;
PETH_STACK         pSystemStack = NULL;
PSTATION_INFO      pUserList = 0;    //PSTATION_INFO structure for
                                     //windows application
PSTATION_INFO      pSystemList = 0;  //PSTATION_INFO structure for
                                     //RT application
ULONG              StationNum = 0;
FP_PNIO_ENTER      fpPnioEnter = NULL;     //Function pointer to PnioEnter
FP_PNIO_EXIT       fpPnioExit = NULL;      //Function pointer to PnioExit
PUCHAR             pInData;
PUCHAR             pOutData;

//*****************************************************************
//*** !!! Check if compiler setting /GZ was removed !!!         ***
//*****************************************************************

void static AppTask(void)
{
 //Check if system memory is present
 if ((!pSystemStack) ||
       (!pSystemList))
       return;

 //Call PNIO enter function
 PSTATION_INFO    pStation    =    fpPnioEnter(pSystemStack,    pSystemList,
StationNum);
 if (pStation)
 {
       //Get input  data from Slot3 (IB IL 24 DI 4-ME)
       //Set output data to   Slot2 (IB IL 24 DO 4-ME)
       PNIO_GET_INPUT_DATAPTR (pStation, 3, 0, &pInData);
       PNIO_GET_OUTPUT_DATAPTR(pStation, 2, 0, &pOutData);
       if ((pInData) &&
           (pOutData))
       {
             //Set input data to output data
             *pOutData = *pInData;
       }
 }

 //Call PNIO exit function
 fpPnioExit(pStation);
}


void main(void)
{
 ULONG i;

 printf("\n*** ProfiNET Core Test ***\n\n");

 //Required PNIO parameters
 PNIO_PARAMS PnioParams;
 PnioParams.EthParams.dev_num = 0;
 PnioParams.EthParams.period = 100;
 PnioParams.EthParams.sched_cnt = 1;
 PnioParams.EthParams.fpAppTask = AppTask;

 //Set station list file path
 sprintf(PnioParams.szStationFile, "c:\\temp\\stationlist.par");

 //Enable PNIO realtime core
 if (ERROR_SUCCESS == ShaPnioCreate(&PnioParams))
 {
       //Init global elements
       pUserStack  = PnioParams.EthParams.pUserStack;
       pSystemStack= PnioParams.EthParams.pSystemStack;
       pUserList   = PnioParams.pUserList;
       pSystemList = PnioParams.pSystemList;
       StationNum  = PnioParams.StationNum;
       fpPnioEnter = PnioParams.fpPnioEnter;
       fpPnioExit  = PnioParams.fpPnioExit;

       //Get version information
       ShaPnioGetVersion(&PnioParams);
       printf("PNIODEVICE-DLL: %.2f\nPNIODEVICE-DRV: %.2f\n",
             PnioParams.core_dll_ver / (double)100,
             PnioParams.core_drv_ver / (double)100);
       printf("ETHCORE-DLL : %.2f\nETHCORE-DRV : %.2f\n",
             PnioParams.EthParams.core_dll_ver / (double)100,
             PnioParams.EthParams.core_drv_ver / (double)100);
       printf("SHA-LIB     : %.2f\nSHA-DRV     : %.2f\n",
             PnioParams.EthParams.sha_lib_ver / (double)100,
             PnioParams.EthParams.sha_drv_ver / (double)100);
       printf("\n");

       //Set station names and IPs from file
       PnioSetFromStationFile(PnioParams.szStationFile);

       //Enable stations
       for (i=0; i<PnioParams.StationNum; i++)
             PnioEnableStation(&pUserList[i]);

       //Wait for key pressed
       printf("\nPress any key ...\n");
       while (!kbhit()) { Sleep(100); }

       //Disable stations
       for (i=0; i<PnioParams.StationNum; i++)
             PnioDisableStation(&pUserList[i]);

       //Destroy PNIO realtime core
       ShaPnioDestroy(&PnioParams);
 }
}
```

5.2 Command functions

The low level interface provides all function to handle ProfinetIO Commands

5.2.1 Enable station

VC `ULONG Pnio64EnableStation(PSTATION_INFO pStation);`

5.2.2 Disable station

VC `ULONG Pnio64DisableStation(PSTATION_INFO pStation);`

6 Realtime Operation

After enabling the ProfiNET system (*ShaPnioCreate*) with a corresponding station list, the realtime tasks become active. The application realtime task is decorated by Realtime Wrapper functions:

```
//Call PNIO enter function (Return: pointer to current station)
PSTATION_INFO pStation = __fpPnioEnter(
                              __pSystemStack,   //In: Ethernet Stack Pointer
                              __pSystemList,    //In: Station List
                              __StationNum);    //In: Number of stations
```

```
typedef PSTATION_INFO   (__cdecl *FP_PNIO_ENTER)
                                    (PETH_STACK, PSTATION_INFO, ULONG);
typedef VOID            (__cdecl *FP_PNIO_EXIT) (PSTATION_INFO);
```

These wrapper functions are used to manage the realtime ProfiNET protocol management, like ethernet frame update, error handling, stack management,… The ProfiNET Library Realtime System itself is managed by syncronized states:

```
//Define PNIO Wrapper States
enum _PNIO_STATES
{
      PNIO_STATE_INIT = 0,
      PNIO_STATE_CONNECTED,
      PNIO_STATE_ACTIVE,
      PNIO_STATE_WRITTEN,
      PNIO_STATE_CONTROLLED,
      PNIO_STATE_RUNNING,
      PNIO_STATE_ERROR
};
```

Get station state

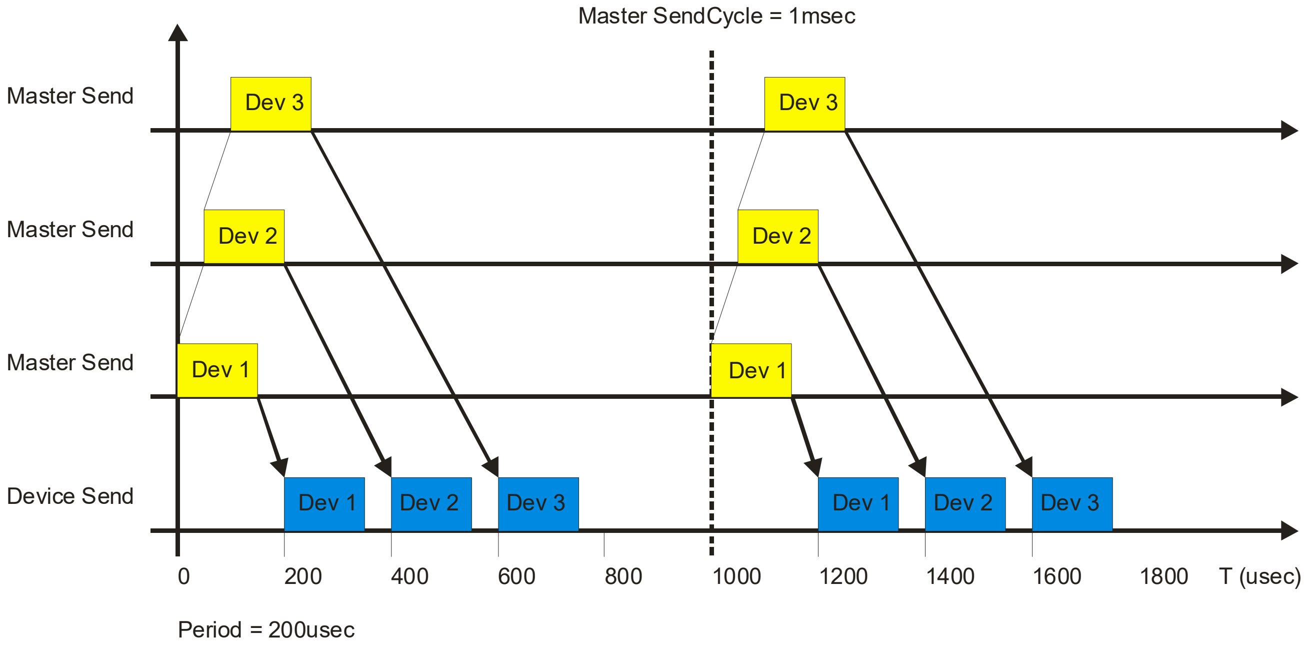
```
if (pStation->State == PNIO_STATE_RUNNING)
{
      …
};
```

Sample

```
void static AppTask(void)
{
      PUCHAR pInData;
      PUCHAR pOutData;
      BOOLEAN bResult;

      //Check if system memory is present
      if ((!__pSystemStack) ||
          (!__pSystemList))
          return;

      //Call PNIO enter function
      PSTATION_INFO pStation = __fpPnioEnter(
                                    __pSystemStack,
                                    __pSystemList,
                                    __StationNum);
      if (pStation)
      {
            //Check station name
            if (__PnioCheckStationName(pStation, "station1"))
            {
                  //Anybus PRT Modul on development board
                  //Get input  pointer from Slot1, Offset0 (2 Byte Input)
                  //Set output pointer to   Slot2, Offset0 (2 Byte Output)
                  PNIO_GET_INPUT_DATAPTR (pStation, 1, 0, &pInData);
                  PNIO_GET_OUTPUT_DATAPTR(pStation, 2, 0, &pOutData);

                  if ((pInData) &&
                      (pOutData))
                  {
                        //Set outputs as inputs
                        pOutData[0] = pInData[0];
                        pOutData[1] = pInData[1];
                  }

                  //Increase update counter
                  __UpdateCnt++;
            }
      }

      //Call PNIO exit function
      __fpPnioExit(pStation);
}
```

6.1 PLL Send Mode

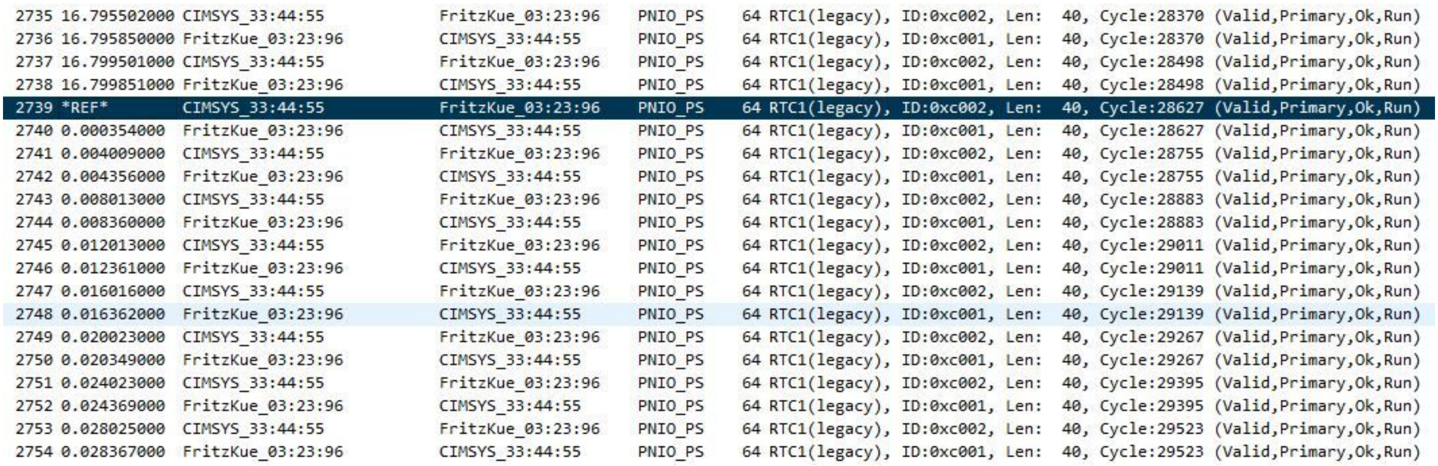
With the PLL send mode, a station is bound to the send timing of the master. The device will send its frame, when receiving a master frame.



Registry:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Sybera\PNS\SendMode        0
```

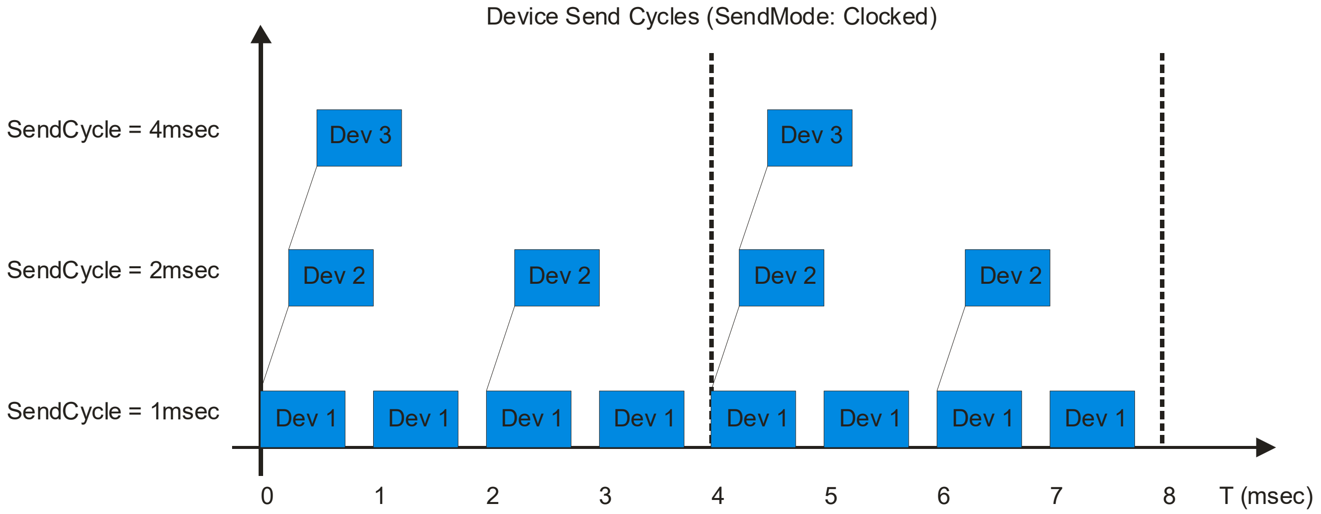
Wireshark:



```
2735 16.795502000 CIMSYS_33:44:55           FritzKue_03:23:96   PNIO_PS    64 RTC1(legacy), ID:0xc002, Len:  40, Cycle:28370 (Valid,Primary,Ok,Run)
2736 16.795850000 FritzKue_03:23:96         CIMSYS_33:44:55     PNIO_PS    64 RTC1(legacy), ID:0xc001, Len:  40, Cycle:28370 (Valid,Primary,Ok,Run)
2737 16.799501000 CIMSYS_33:44:55           FritzKue_03:23:96   PNIO_PS    64 RTC1(legacy), ID:0xc002, Len:  40, Cycle:28498 (Valid,Primary,Ok,Run)
2738 16.799851000 FritzKue_03:23:96         CIMSYS_33:44:55     PNIO_PS    64 RTC1(legacy), ID:0xc001, Len:  40, Cycle:28498 (Valid,Primary,Ok,Run)
2739 *REF*        CIMSYS_33:44:55           FritzKue_03:23:96   PNIO_PS    64 RTC1(legacy), ID:0xc002, Len:  40, Cycle:28627 (Valid,Primary,Ok,Run)
2740 0.000354000  FritzKue_03:23:96         CIMSYS_33:44:55     PNIO_PS    64 RTC1(legacy), ID:0xc001, Len:  40, Cycle:28627 (Valid,Primary,Ok,Run)
2741 0.004009000  CIMSYS_33:44:55           FritzKue_03:23:96   PNIO_PS    64 RTC1(legacy), ID:0xc002, Len:  40, Cycle:28755 (Valid,Primary,Ok,Run)
2742 0.004356000  FritzKue_03:23:96         CIMSYS_33:44:55     PNIO_PS    64 RTC1(legacy), ID:0xc001, Len:  40, Cycle:28755 (Valid,Primary,Ok,Run)
2743 0.008013000  CIMSYS_33:44:55           FritzKue_03:23:96   PNIO_PS    64 RTC1(legacy), ID:0xc002, Len:  40, Cycle:28883 (Valid,Primary,Ok,Run)
2744 0.008360000  FritzKue_03:23:96         CIMSYS_33:44:55     PNIO_PS    64 RTC1(legacy), ID:0xc001, Len:  40, Cycle:28883 (Valid,Primary,Ok,Run)
2745 0.012013000  CIMSYS_33:44:55           FritzKue_03:23:96   PNIO_PS    64 RTC1(legacy), ID:0xc002, Len:  40, Cycle:29011 (Valid,Primary,Ok,Run)
2746 0.012361000  FritzKue_03:23:96         CIMSYS_33:44:55     PNIO_PS    64 RTC1(legacy), ID:0xc001, Len:  40, Cycle:29011 (Valid,Primary,Ok,Run)
2747 0.016016000  CIMSYS_33:44:55           FritzKue_03:23:96   PNIO_PS    64 RTC1(legacy), ID:0xc002, Len:  40, Cycle:29139 (Valid,Primary,Ok,Run)
2748 0.016362000  FritzKue_03:23:96         CIMSYS_33:44:55     PNIO_PS    64 RTC1(legacy), ID:0xc001, Len:  40, Cycle:29139 (Valid,Primary,Ok,Run)
2749 0.020023000  CIMSYS_33:44:55           FritzKue_03:23:96   PNIO_PS    64 RTC1(legacy), ID:0xc002, Len:  40, Cycle:29267 (Valid,Primary,Ok,Run)
2750 0.020349000  FritzKue_03:23:96         CIMSYS_33:44:55     PNIO_PS    64 RTC1(legacy), ID:0xc001, Len:  40, Cycle:29267 (Valid,Primary,Ok,Run)
2751 0.024023000  CIMSYS_33:44:55           FritzKue_03:23:96   PNIO_PS    64 RTC1(legacy), ID:0xc002, Len:  40, Cycle:29395 (Valid,Primary,Ok,Run)
2752 0.024369000  FritzKue_03:23:96         CIMSYS_33:44:55     PNIO_PS    64 RTC1(legacy), ID:0xc001, Len:  40, Cycle:29395 (Valid,Primary,Ok,Run)
2753 0.028025000  CIMSYS_33:44:55           FritzKue_03:23:96   PNIO_PS    64 RTC1(legacy), ID:0xc002, Len:  40, Cycle:29523 (Valid,Primary,Ok,Run)
2754 0.028367000  FritzKue_03:23:96         CIMSYS_33:44:55     PNIO_PS    64 RTC1(legacy), ID:0xc001, Len:  40, Cycle:29523 (Valid,Primary,Ok,Run)
```

6.2 Clocked Send Mode

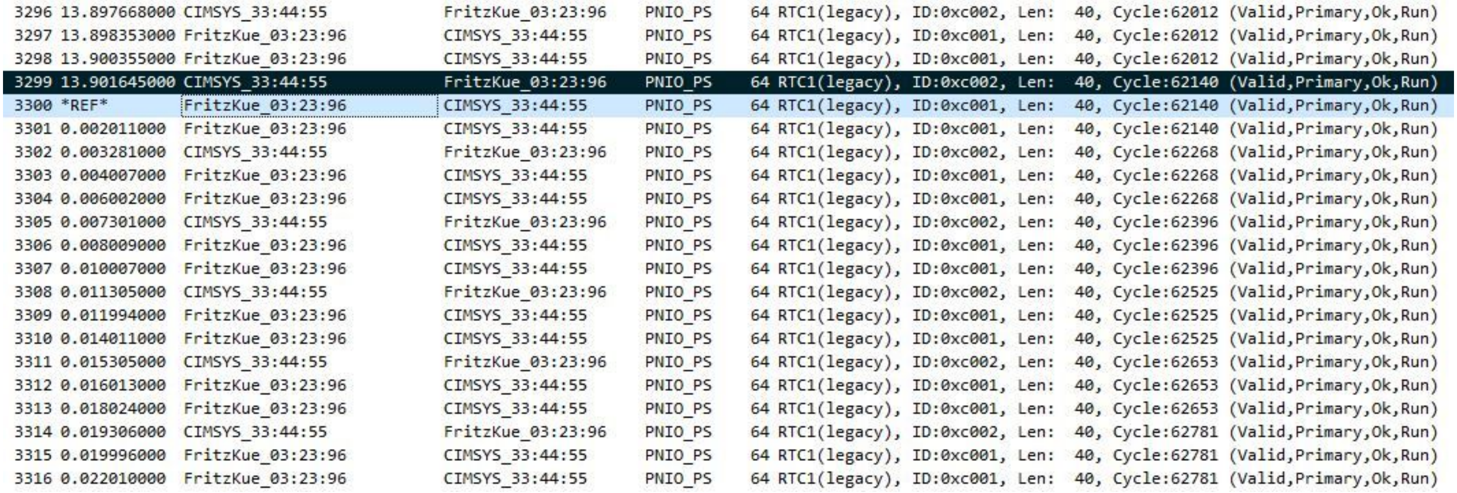
With the clocked send mode, a station is bound to a (master) specified send cycle, independently to the master send cycle itself. This allows a different device send cycle to the master send cycle.



Registry:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Sybera\PNS\SendMode        1
```

Wireshark:



```
3296 13.897668000 CIMSYS_33:44:55       FritzKue_03:23:96   PNIO_PS    64 RTC1(legacy), ID:0xc002, Len:  40, Cycle:62012 (Valid,Primary,Ok,Run)
3297 13.898353000 FritzKue_03:23:96     CIMSYS_33:44:55     PNIO_PS    64 RTC1(legacy), ID:0xc001, Len:  40, Cycle:62012 (Valid,Primary,Ok,Run)
3298 13.900355000 FritzKue_03:23:96     CIMSYS_33:44:55     PNIO_PS    64 RTC1(legacy), ID:0xc001, Len:  40, Cycle:62012 (Valid,Primary,Ok,Run)
3299 13.901645000 CIMSYS_33:44:55       FritzKue_03:23:96   PNIO_PS    64 RTC1(legacy), ID:0xc002, Len:  40, Cycle:62140 (Valid,Primary,Ok,Run)
3300 *REF*        FritzKue_03:23:96     CIMSYS_33:44:55     PNIO_PS    64 RTC1(legacy), ID:0xc001, Len:  40, Cycle:62140 (Valid,Primary,Ok,Run)
3301 0.002011000  FritzKue_03:23:96     CIMSYS_33:44:55     PNIO_PS    64 RTC1(legacy), ID:0xc001, Len:  40, Cycle:62140 (Valid,Primary,Ok,Run)
3302 0.003281000  CIMSYS_33:44:55       FritzKue_03:23:96   PNIO_PS    64 RTC1(legacy), ID:0xc002, Len:  40, Cycle:62268 (Valid,Primary,Ok,Run)
3303 0.004007000  FritzKue_03:23:96     CIMSYS_33:44:55     PNIO_PS    64 RTC1(legacy), ID:0xc001, Len:  40, Cycle:62268 (Valid,Primary,Ok,Run)
3304 0.006002000  FritzKue_03:23:96     CIMSYS_33:44:55     PNIO_PS    64 RTC1(legacy), ID:0xc001, Len:  40, Cycle:62268 (Valid,Primary,Ok,Run)
3305 0.007301000  CIMSYS_33:44:55       FritzKue_03:23:96   PNIO_PS    64 RTC1(legacy), ID:0xc002, Len:  40, Cycle:62396 (Valid,Primary,Ok,Run)
3306 0.008009000  FritzKue_03:23:96     CIMSYS_33:44:55     PNIO_PS    64 RTC1(legacy), ID:0xc001, Len:  40, Cycle:62396 (Valid,Primary,Ok,Run)
3307 0.010007000  FritzKue_03:23:96     CIMSYS_33:44:55     PNIO_PS    64 RTC1(legacy), ID:0xc001, Len:  40, Cycle:62396 (Valid,Primary,Ok,Run)
3308 0.011305000  CIMSYS_33:44:55       FritzKue_03:23:96   PNIO_PS    64 RTC1(legacy), ID:0xc002, Len:  40, Cycle:62525 (Valid,Primary,Ok,Run)
3309 0.011994000  FritzKue_03:23:96     CIMSYS_33:44:55     PNIO_PS    64 RTC1(legacy), ID:0xc001, Len:  40, Cycle:62525 (Valid,Primary,Ok,Run)
3310 0.014011000  FritzKue_03:23:96     CIMSYS_33:44:55     PNIO_PS    64 RTC1(legacy), ID:0xc001, Len:  40, Cycle:62525 (Valid,Primary,Ok,Run)
3311 0.015305000  CIMSYS_33:44:55       FritzKue_03:23:96   PNIO_PS    64 RTC1(legacy), ID:0xc002, Len:  40, Cycle:62653 (Valid,Primary,Ok,Run)
3312 0.016013000  FritzKue_03:23:96     CIMSYS_33:44:55     PNIO_PS    64 RTC1(legacy), ID:0xc001, Len:  40, Cycle:62653 (Valid,Primary,Ok,Run)
3313 0.018024000  FritzKue_03:23:96     CIMSYS_33:44:55     PNIO_PS    64 RTC1(legacy), ID:0xc001, Len:  40, Cycle:62653 (Valid,Primary,Ok,Run)
3314 0.019306000  CIMSYS_33:44:55       FritzKue_03:23:96   PNIO_PS    64 RTC1(legacy), ID:0xc002, Len:  40, Cycle:62781 (Valid,Primary,Ok,Run)
3315 0.019996000  FritzKue_03:23:96     CIMSYS_33:44:55     PNIO_PS    64 RTC1(legacy), ID:0xc001, Len:  40, Cycle:62781 (Valid,Primary,Ok,Run)
3316 0.022010000  FritzKue_03:23:96     CIMSYS_33:44:55     PNIO_PS    64 RTC1(legacy), ID:0xc001, Len:  40, Cycle:62781 (Valid,Primary,Ok,Run)
```

7 Error Handling

The Device library provides an error handling and tracing mechanism.

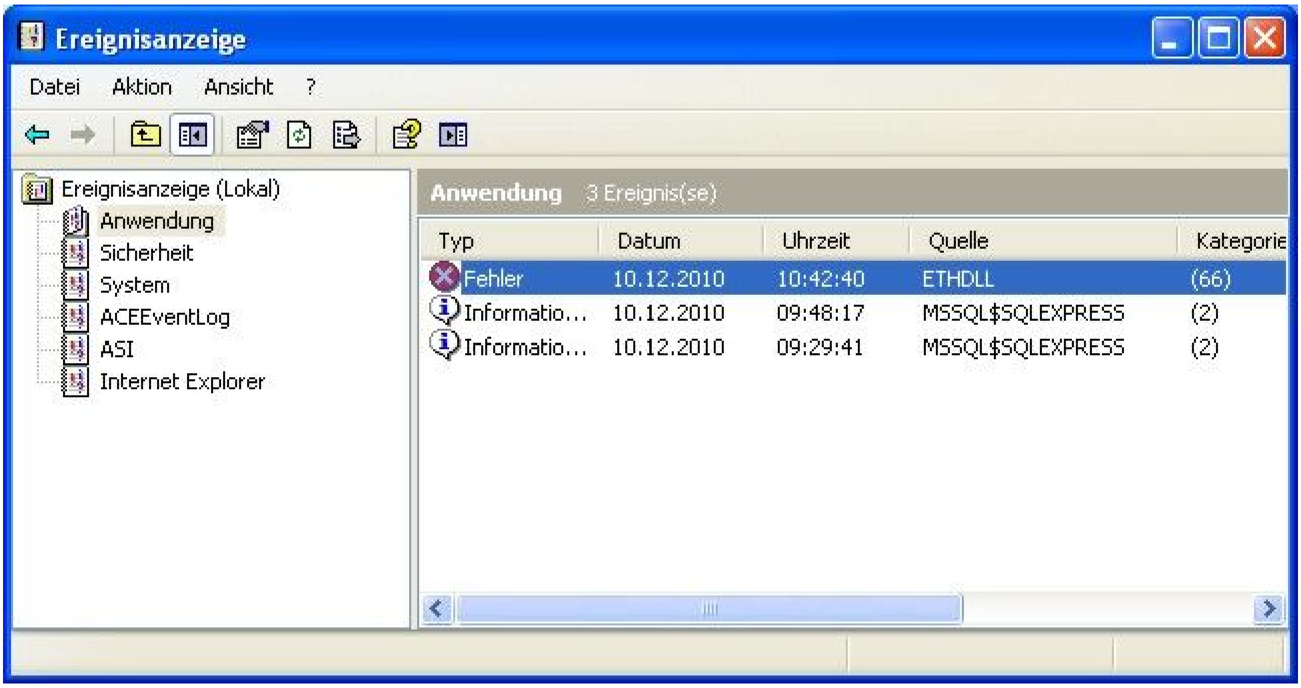
7.1 Debug LOG File

On execution the Device library creates a sequence file PNTDBG.LOG in Text-Format

Note: This file is not accessible while the application is running

7.2 Event File

On execution the Device library logs error event to the Windows Event Manager. The Device library logs Application and System events. These events can be exported to a file and provided for support purposes.



Typ	Datum	Uhrzeit	Quelle	Kategorie
Fehler	10.12.2010	10:42:40	ETHDLL	(66)
Informatio...	10.12.2010	09:48:17	MSSQL$SQLEXPRESS	(2)
Informatio...	10.12.2010	09:29:41	MSSQL$SQLEXPRESS	(2)

8 Related Dokuments

• manual_sha_e.pdf	(SHA Realtime Library)
• manual_eth_e.pdf	(ETH Realtime Library)
• profinet pld service.pdf	