TM

SERCOS III
Realtime Master Library
Documentation
(Cluster 32/64 Bit)

Date: Oct, 22.2013

SERCOS III Realtime Master Library Documentation

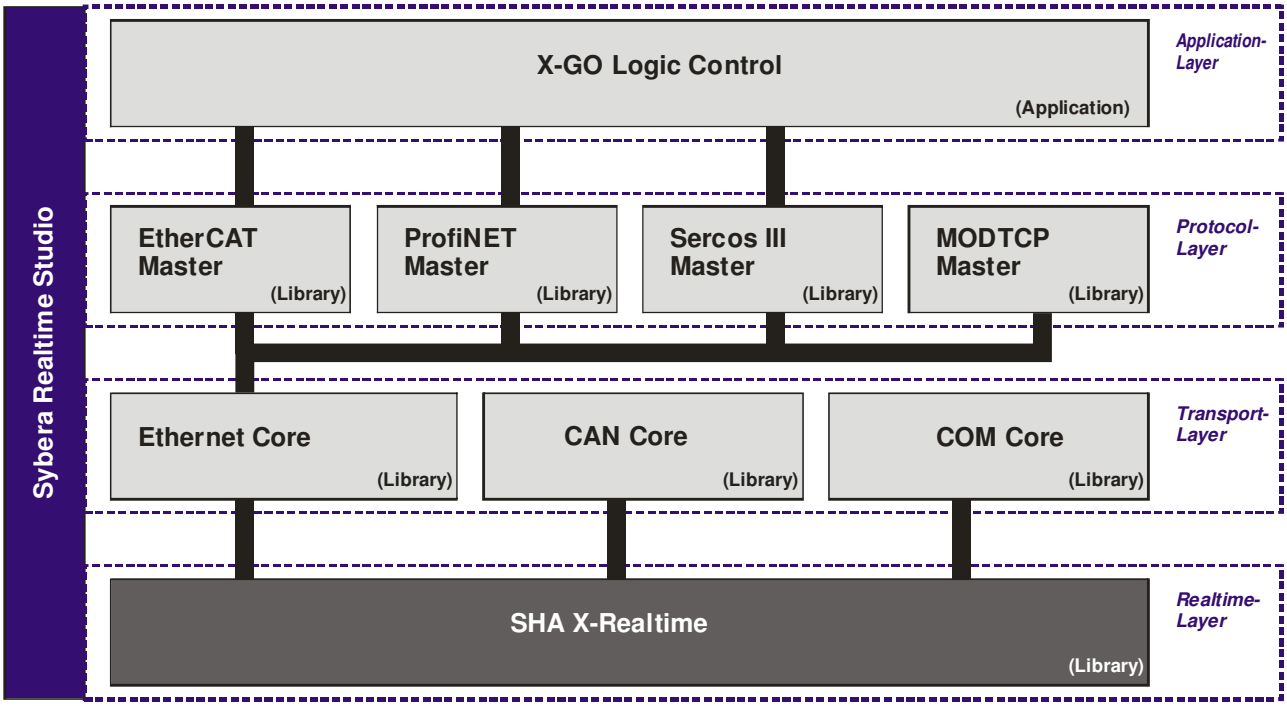
SYBERA Copyright © 2010

SYBERA Copyright © 2010

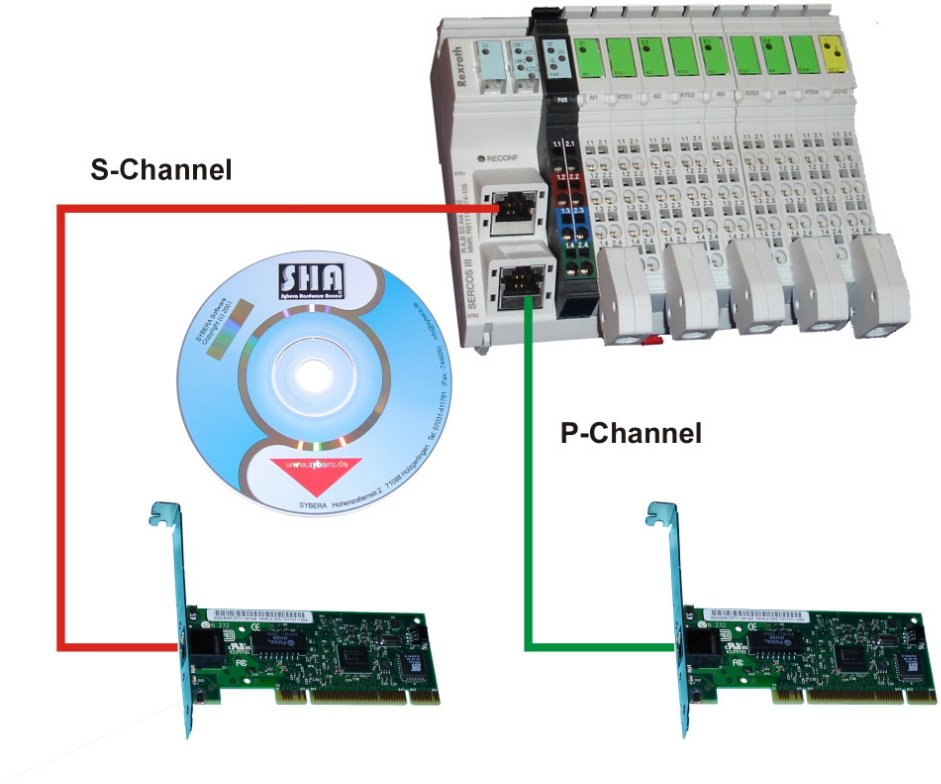
SYBERA Copyright © 2010

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 Moduls, 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.



SYBERA Copyright © 2010



With the PC-based SERCOS III master Sybera extends the program of realtime library software. The SERCOS III master fullfils the requirements of the specification 1.1.2 and is realized like the other Sybera master libraries as an open programming system. In particular it was taken care on the critical cycle timing of the SERCOS III protocol management.

Together with the PC based SERCOS III master and the X-Realtime engine there is no need of a separate SERCOS III controller hardware, because the master protocol handling is realized directly by a PC hardware with standard Ethernet adapter(s), and with Sampling rates under 100 usec. By the extension of the realtime ethernet transport layer (Ethernet Realtime Core) more than 70 standard ethernet adapters are supported.

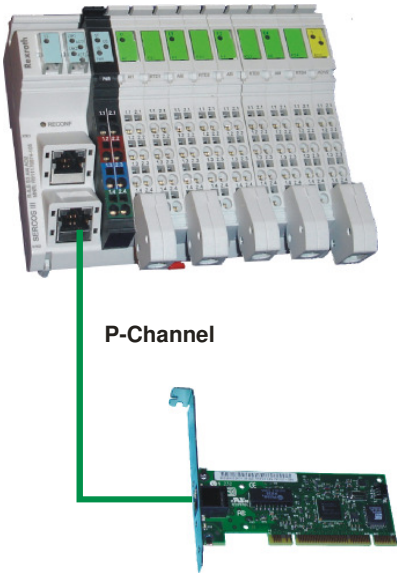
The SERCOS III master is offered by SYBERA as an open realtime library system and allows the developer the programming of a deterministic control software for SERCOS III devices. The developer should have the possibility, to program control projects by easy interface functions and should be able to realize graphical interfaces in combination. As with the standard applications the developer must have the possibility to realise the device initialisation and the logical connectivity by library functions. This is realized by an flexible IDN parametricing system.

SYBERA Copyright © 2010

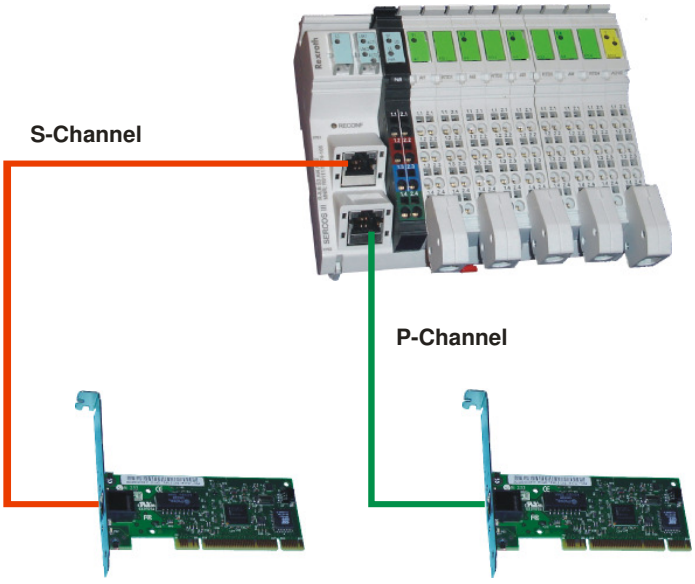
Beyond that it must be ensured, that influence can be taken with the library functions also on complex procedures, like e.g. condition change of the devices or error situations. The SERCOS protocol management, the error handling and device logistics are implemented in so called master protocol stacks. The protocol stacks thus form the connection between the physical transport layer (e.g. ethernet driver) and the application software.
The objective of protocol stacks is alike with all systems: The developer of SERCOS III projects shall not care for the protocol management, rather for the processing of the appropriate payload data of the linked stations.

The Sybera SERCOS III Master allows line or ring topology. For ring topology 2 PC ethernet adapters are required (may be different types).

Line Topology



Ring Topology



SYBERA Copyright © 2010

1.1 Product Features

- Intelligent Station Management
- Station Realtime Cycles upto 500 µsec
- Line or Ring Topology
- IDN Management
- Profile Management
- Phase Management

1.2 Supported Platforms

SHA was build to support serveral development platforms. Currently following platforms are supported:

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

1.3 Supported OS

- Windows XP, VISTA, 7, 8 (32 / 64 Bit)

SYBERA Copyright © 2010

2 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
2. Check the installed Ethernet adapter has given a correct IP address

Installation

3. Install SHA realtime system (separate software package)
4. Install ETH transport library (separate software package)
5. Optional: Install a second Ethernet X-Realtime NDIS Driver(s) (see manual)
6. Run the program SYSETUP(32/64) of the Sercos III 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
7. Optional: Check license with SYLICENCECHECK(32/64).EXE

Operation

8. Run SC3CHECK.EXE
9. Build the program with the library interface
10. Run the program

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

Note: In order to operate SYBERA software under Windows 8, 7, VISTA, it must be carried out with ADMINISTRATOR priviledges.

SYBERA Copyright © 2010

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.

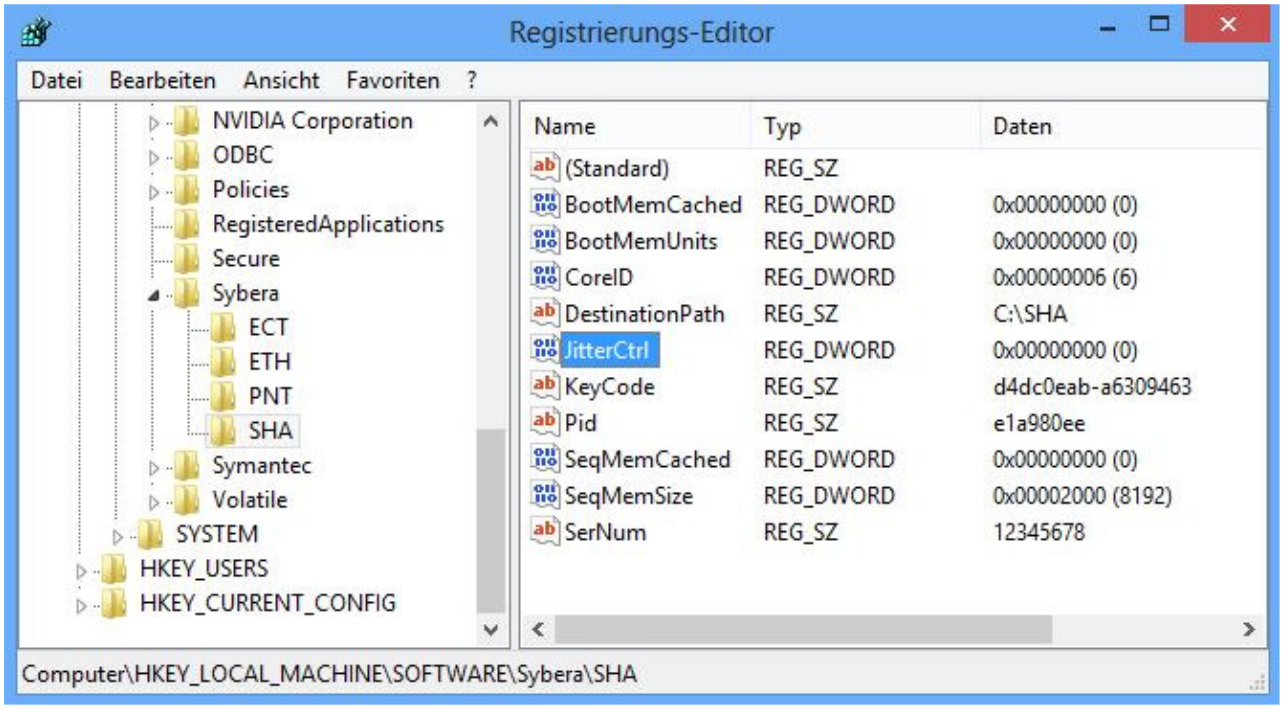
C1E enhanced halt state — Introduced in the Pentium 4 500J-series processors, the C1E halt state replaces the old C1 halt state used on the Pentium 4 and most other x86 CPUs. The C1 halt state is invoked when the operating system's idle process issues a HLT command. (Windows does this constantly when 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.

SYBERA Copyright © 2010

2.1 Jitter Control

Since a notebook has a quiet different jitter behaviour than desktop systems, an enhanced jitter control mechanism is required. Therefore SYBERA provides a registry entry called “JitterCtrl”. This entry allows an adaptive iteration to the best jitter behaviour of the notebook.



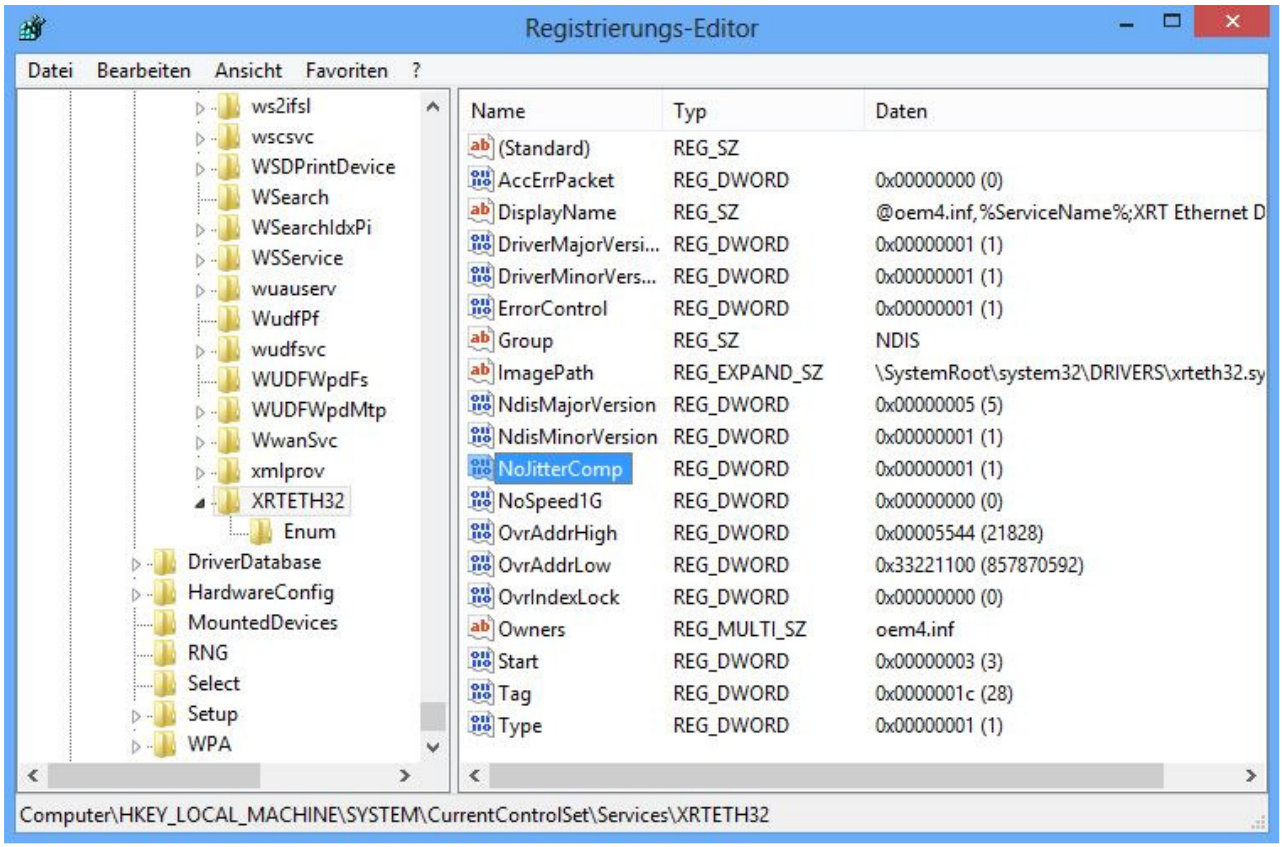
Following values are valid:

0: No enhanced jitter control
1: Enhaced Jitter Control, Step 1 (first choice together with BIOS settings)
2: Enhaced Jitter Control, Step 2 (for INTEL platforms only)
3: Enhaced Jitter Control, Step 3 (for INTEL platforms only, together with BIOS settings)

SYBERA Copyright © 2010

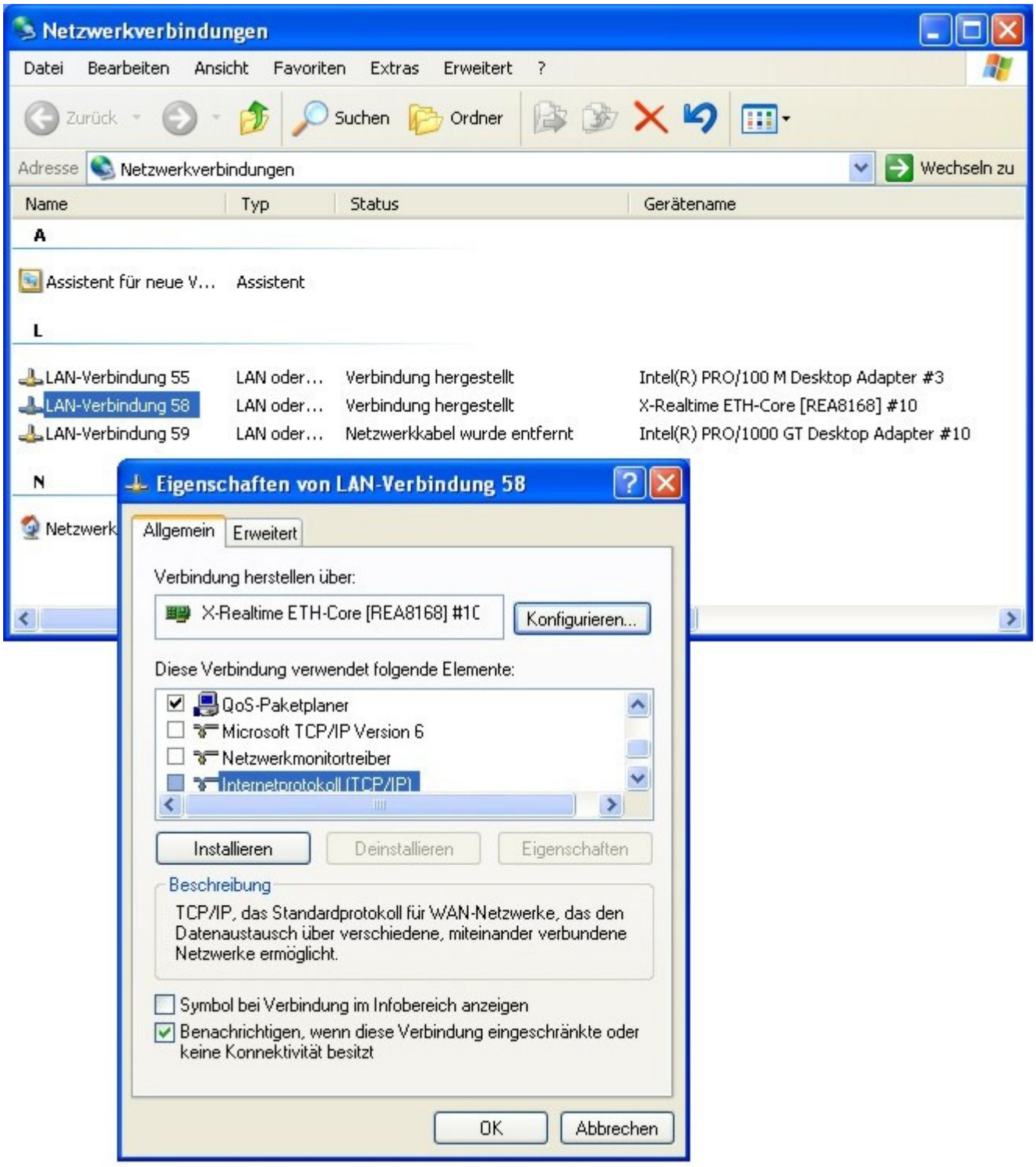
2.2 Dynamic Jitter Compensation

SYBERA uses the procedure "Dynamic Jitter Compensation" with active and passive feedback compensation within the realtime engine. Although the X-Real time engine of SYBERA allows a native maximum Jitter of approx. 15 µ sec (according to hardware platform), this behaviour may be reduced below 3 µsec by the dynamic jitter compensation.
For compatibility reason on some platforms it may be required to disable the dynamic jitter compensation. Therefore the registry value “NoJitterComp” has to be set to 1



SYBERA Copyright © 2010

Note: For proper operation its recommended to use the SERCOS III network as standalone network. This requires to turn off the Windows protocols for this network connection:



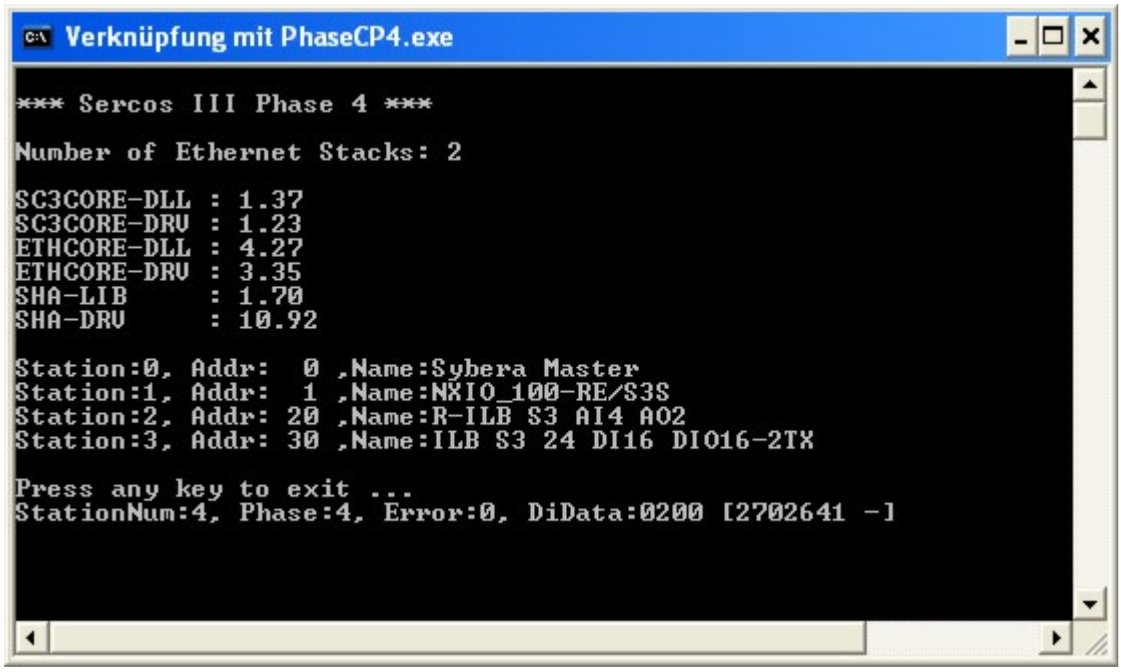
SYBERA Copyright © 2010

3 SERCOS III Realtime Master Library

The interface functions of the SERCOS III Realtime Master Library are exported by a dynamic link library. Following include files and libraries are required:

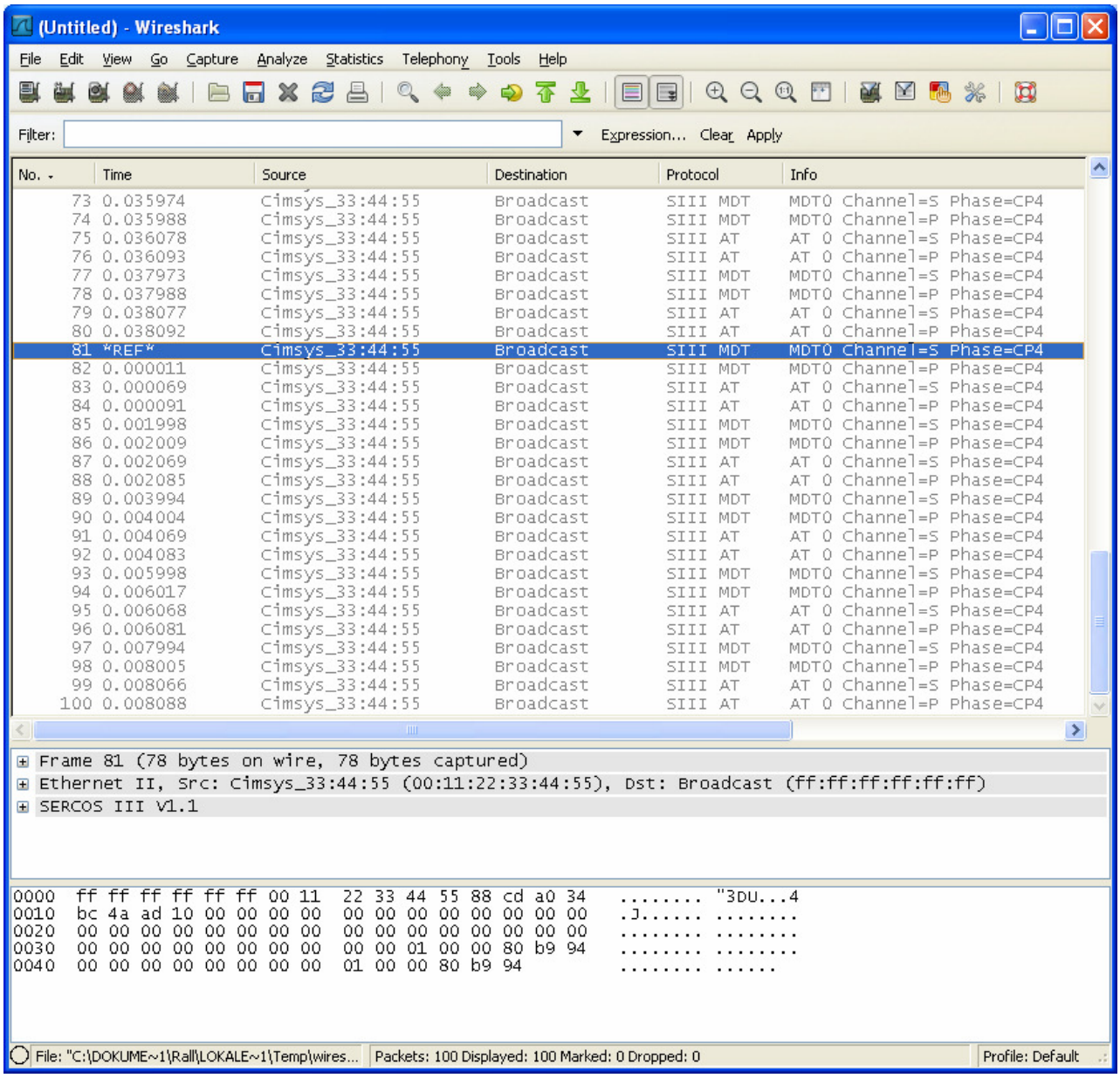
SHASC3CORE.DLL	SERCOS III Master DLL (Visual C++)
SHASC3CORE.LIB	SERCOS III Master LIB (Visual C++)
SHASC3COREOML.DLL	SERCOS III Master DLL (Borland C++ / Delphi)
SHASC3COREOML.LIB	SERCOS III Master LIB (Borland C++ / Delphi)
SC3DEVICE.PAR	Native Station Configuration File
SHASC3CORE.H	Exported Function Prototypes
SC3COREDEF.H	SERCOS III Basic Definitions
SC3SVCDEF.H	SERCOS III Service Definitions
SC3PROCDEF.H	SERCOS III Procedure Definitions
SC3IDNDEF.H	SERCOS III IDN Definitions
SC3MACROS.H	SERCOS III Macro Definitions

Sample Application



SYBERA Copyright © 2010

Sample Protocol Timing (Phase 4, Ring Topology)



SYBERA Copyright © 2010

3.1 Header File SC3COREDEF(32/64).H

The header file SC3COREDEF(32/64).H is required when handling SERCOS III telegrams by the interface functions or handling the EthernetCore Realtime Stack directly (Realtime Level2). It also defines the SERCOS III telegram commands and structures.

3.1.1 Structure SC3_PARAMS

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

```
typedef struct _SC3_PARAMS
{
      //SERCOS3 parameters
      UCHAR       Phase;                      //Communication Phase
      ULONG       CyclePeriod;                //Cycle Period [µsec]
      SC3_MDT     MdtList[MAX_TEL_NUM];       //MDT Telegram List
      USHORT      MdtLengths[MAX_TEL_NUM];    //MDT Telegram Lengths
      SC3_AT      AtList[MAX_TEL_NUM];        //AT Telegram List
      USHORT      AtLengths[MAX_TEL_NUM];     //AT Telegram Lengths
      BOOLEAN     bManualConfig;              //Manual station configuration

      //Global parameters
      FP_SC3_ENTER      fpSc3Enter;           //Function Pointer to Sc3Enter()
      FP_SC3_EXIT       fpSc3Exit;            //Function Pointer to Sc3Exit()
      ULONG             core_dll_ver;         //Core DLL version
      ULONG             core_drv_ver;         //Core driver version

      //Ethernet stack parameters
      ULONG       StackNum;                   //Number of Ethernet Stacks
      ETH_PARAMS  EthParams[MAX_STACK_NUM];   //Ethernet Core Parameters
                                              //for NIC0 and NIC1 (optional)

      //Station parameters
      ULONG             StationNum;   //Station Number
      PSTATION_INFO     pSystemList;   //Station List for Realtime Application
Task
      PSTATION_INFO     pUserList;      //Station List for Windows Application
Task

} SC3_PARAMS, *PSC3_PARAMS;
```

Note:

The structure ETH_PARAMS is part of the Ethernet Core Library and described in the the documentation of this core library. 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 elements of SC3_PARAMS.

SYBERA Copyright © 2010

3.1.2 Structure STATION_INFO

This structure keeps all information of each SERCOS III station (including master station 0) and may be required for further interface functions.

```
typedef struct _STATION_INFO
{
      char              szName[MAX_PATH_SIZE];        //Name of Station
      char              szDeviceID[MAX_PATH_SIZE];    //Device ID
      ULONG             Index;                        //Station Index
      USHORT            Addr;                         //Station Address
      SC3_TYPE_VER      TypeVer;                      //Station Type and Version

      SC3_ITEM_OFFS     MdtSvcOffs;                   //MDT Service Offset
      SC3_ITEM_OFFS     MdtDevOffs;                   //MDT Device Offset
      SC3_ITEM_OFFS     MdtConOffs[MAX_CON_NUM];      //MDT Connection Offsets
      USHORT                  MdtConLen[MAX_CON_NUM]; //MDT Connection Lengths
      ULONG             MdtConNum;                    //MDT Connection Number

      SC3_ITEM_OFFS     AtSvcOffs;                    //AT Service Offset
      SC3_ITEM_OFFS     AtDevOffs;                    //AT Device Offset
      SC3_ITEM_OFFS     AtConOffs[MAX_CON_NUM];       //AT Connection Offsets
      USHORT            AtConLen[MAX_CON_NUM];        //AT Connection Lengths
      ULONG             AtConNum;                     //AT Connection Number

      PSC3_MDT          pMdtList[MAX_TEL_NUM];        //MDT pointer list
      PSC3_AT           pAtList[MAX_TEL_NUM];         //AT pointer list

      BOOLEAN           bUpdate[MAX_UPDATE_DIR];      //Station Update Flags
      BOOLEAN           bDisable;                     //Station Disable Flag

} STATION_INFO, *PSTATION_INFO;
```

Note:

The SERCOS III library structures (SC3_MDT, SC3_AT, SC3_SVC_CTRL, …) are directly corresponding to the SERCOS III protocol elements and are described in detail inside the SERCOS III specification 1.1.2.

SYBERA Copyright © 2010

3.2 Header File SC3MACROS(32/64).H

This header file defines all macros required for handling SERCOS III protocol data. The following describes some of the most important macros:

This Inline-Macro is to set the IDN structure:

```
__inline VOID __SetIDN(
                  PSC3_IDN pIdn,
                  ULONG IdnType,
                  ULONG ParamSet,
                  ULONG Dbn,
                  ULONG Si,
                  ULONG Se)
```

This Inline-Macro is to update elements of the IDN structure:

```
__inline VOID __UpdateIDN(
                  PSC3_IDN pIdn,
                  PULONG pIdnType,
                  PULONG pParamSet,
                  PULONG pDbn,
                  PULONG pSi,
                  PULONG pSe)
```

This Inline-Macro is to get elements of the IDN structure:

```
__inline VOID __GetIDN(
                  PSC3_IDN pIdn,
                  PULONG pIdnType,
                  PULONG pParamSet,
                  PULONG pDbn,
                  PULONG pSi,
                  PULONG pSe)
```

SYBERA Copyright © 2010

This Inline-Macro is to set the elements of the service control structure:

```
__inline VOID __SetSvcCtrl(
                        PSC3_SVC_CTRL pSvcCtrl,
                        BOOLEAN bMasterHS,
                        BOOLEAN bWrFlag,
                        BOOLEAN bLastTransmit,
                        USHORT ElementType,
                        ULONG SrvInfo)
```

This Inline-Macro is to update the elements of the service control structure:

```
__inline VOID __UpdateSvcCtrl(
                        PSC3_SVC_CTRL pSvcCtrl,
                        PBOOLEAN pbMasterHS,
                        PBOOLEAN pbWrFlag,
                        PBOOLEAN pbLastTransmit,
                        PUSHORT  pElementType,
                        PULONG   pSrvInfo)
```

This Inline-Macro is to get the elements of the service status structure:

```
__inline VOID __GetSvcStat(
                        PSC3_SVC_STAT pSvcStat,
                        PBOOLEAN pbSlaveHS,
                        PBOOLEAN pbBusy,
                        PBOOLEAN pbSvcError,
                        PBOOLEAN pbSvcValid,
                        PULONG pSvcInfo)
```

SYBERA Copyright © 2010

This Inline-Macro is to set the elements of the device control structure:

```
__inline VOID __SetDevCtrl(
                        PSC3_DEV_CTRL pDevCtrl,
                        BOOLEAN bTopoStat,
                        USHORT TopoCtrl,
                        BOOLEAN bTopoHs,
                        BOOLEAN bIdentReq)
```

This Inline-Macro is to update the elements of the device control structure:

```
__inline VOID __UpdateDevCtrl(
                        PSC3_DEV_CTRL pDevCtrl,
                        PBOOLEAN pbTopoStat,
                        PUSHORT pTopoCtrl,
                        PBOOLEAN pbTopoHs,
                        PBOOLEAN pbIdentReq)
```

This Inline-Macro is to get the elements of the device status structure:

```
__inline VOID __GetDevStat(
                        PSC3_DEV_STAT pDevStat,
                        PBOOLEAN pbPlActive,
                        PBOOLEAN pbCmdChange,
                        PBOOLEAN pbDevWarn,
                        PBOOLEAN pbDevErr,
                        PBOOLEAN pbSlaveValid,
                        PBOOLEAN pbErrCon,
                        PUSHORT pPortStat,
                        PUSHORT pTopoStat,
                        PBOOLEAN pbTopoHS,
                        PBOOLEAN pbCommWarn)
```

SYBERA Copyright © 2010

This Inline-Macro is to get a station by its address

```
__inline PSTATION_INFO __Sc3GetStation(
                                    PSTATION_INFO pStationList,
                                    ULONG StationNum,
                                    USHORT Addr)
```

//Macro to get MDT connection control and data pointer

```
#define     SC3_GET_MDT_CON(__pStation, __ConIndex, __ppConCtrl, __ppConData)
```

//Macro to get AT connection control and data pointer

```
#define     SC3_GET_AT_CON(__pStation, __ConIndex, __ppConCtrl, __ppConData)
```

This Inline-Macro is to set the station data of an IO device

```
__inline VOID __Sc3SetIoData(
                        PSTATION_INFO pStation,
                        ULONG Offs,
                        ULONG DataSize,
                        PUCHAR pData)
```

This Inline-Macro is to get the station data of an IO device

```
__inline PIDN_S_0_1500_X_2 __Sc3GetIoData(
                                    PSTATION_INFO pStation,
                                    ULONG Offs,
                                    ULONG DataSize,
                                    PUCHAR pData)
```

SYBERA Copyright © 2010

3.3 Debug Log File

The SERCOS III master library provides a buildin log sytem which produces a debug log file called *SC3DBG.LOG*. This file contains all nessecary information of the service sequence.

Sample:

```
ConfigureStationList
Station:1, IDN:S-0-0099.0.0 (w), Result:00000000, Len:2, Data: [03] [00]
Station:2, IDN:S-0-0099.0.0 (w), Result:00000000, Len:2, Data: [03] [00]
Station:3, IDN:S-0-0099.0.0 (w), Result:00000000, Len:2, Data: [03] [00]
Station:2, IDN:S-0-1300.0.4 (r), Result:00000000, Len:260, Data: [52] [2d] [49]
…
Station:1, IDN:S-0-1300.0.4 (r), Result:00000000, Len:260, Data: [4e] [58] [49]
…
Station:3, IDN:S-0-1300.0.4 (r), Result:00000000, Len:260, Data: [49] [4c] [42]
…
Station:3, IDN:S-0-1300.0.5 (r), Result:00000000, Len:260, Data: [32] [38] [39]
…
Station:2, IDN:S-0-1300.0.5 (r), Result:00000000, Len:260, Data: [52] [39] [31]
…
Station:1, IDN:S-0-1300.0.5 (r), Result:00000000, Len:260, Data: [4e] [58] [49]
…
Station:2, IDN:S-0-1000.0.0 (r), Result:00000000, Len:2, Data: [01] [01]
Station:3, IDN:S-0-1000.0.0 (r), Result:00000000, Len:2, Data: [01] [01]
Station:1, IDN:S-0-1000.0.0 (r), Result:00000000, Len:2, Data: [01] [01]

SetFgTiming
Station:1, IDN:S-0-1002.0.0 (w), Result:00000000, Len:4, Data: [80] [84] [1e]
[00]
Station:2, IDN:S-0-1002.0.0 (w), Result:00000000, Len:4, Data: [80] [84] [1e]
[00]
Station:3, IDN:S-0-1002.0.0 (w), Result:00000000, Len:4, Data: [80] [84] [1e]
[00]

SetFgBusDiag
Station:1, IDN:S-0-1003.0.0 (w), Result:00000000, Len:4, Data: [0a] [00] [00]
[00]
Station:3, IDN:S-0-1003.0.0 (w), Result:00000000, Len:4, Data: [0a] [00] [00]
[00]
Station:2, IDN:S-0-1003.0.0 (w), Result:00000000, Len:4, Data: [0a] [00] [00]
[00]

SetConnectFix
Station:1, IDN:S-0-1050.0.5 (r), Result:00000000, Len:2, Data: [06] [00]
Station:1, IDN:S-0-1050.1.5 (r), Result:00000000, Len:2, Data: [06] [00]
Station:1, IDN:S-0-1050.0.3 (w), Result:00000000, Len:2, Data: [26] [00]
Station:1, IDN:S-0-1050.1.3 (w), Result:00000000, Len:2, Data: [26] [08]
Station:2, IDN:S-0-1050.0.5 (r), Result:00000000, Len:2, Data: [0c] [00]
Station:2, IDN:S-0-1050.1.5 (r), Result:00000000, Len:2, Data: [08] [00]
Station:2, IDN:S-0-1050.0.3 (w), Result:00000000, Len:2, Data: [2c] [00]
Station:2, IDN:S-0-1050.1.3 (w), Result:00000000, Len:2, Data: [2c] [08]
Station:3, IDN:S-0-1050.0.5 (r), Result:00000000, Len:2, Data: [08] [00]
```

SYBERA Copyright © 2010

```
Station:3, IDN:S-0-1050.1.5 (r), Result:00000000, Len:2, Data: [06] [00]
Station:3, IDN:S-0-1050.0.3 (w), Result:00000000, Len:2, Data: [38] [00]
Station:3, IDN:S-0-1050.1.3 (w), Result:00000000, Len:2, Data: [34] [08]

SetFgTelSetup
Station:1, IDN:S-0-1013.0.0 (w), Result:00000000, Len:2, Data: [08] [00]
Station:2, IDN:S-0-1013.0.0 (w), Result:00000000, Len:2, Data: [0e] [00]
Station:3, IDN:S-0-1013.0.0 (w), Result:00000000, Len:2, Data: [14] [00]
Station:2, IDN:S-0-1009.0.0 (w), Result:00000000, Len:2, Data: [1e] [00]
Station:3, IDN:S-0-1009.0.0 (w), Result:00000000, Len:2, Data: [22] [00]
Station:1, IDN:S-0-1009.0.0 (w), Result:00000000, Len:2, Data: [1a] [00]
Station:2, IDN:S-0-1014.0.0 (w), Result:00000000, Len:2, Data: [0e] [00]
Station:1, IDN:S-0-1014.0.0 (w), Result:00000000, Len:2, Data: [08] [00]
Station:3, IDN:S-0-1014.0.0 (w), Result:00000000, Len:2, Data: [14] [00]
Station:3, IDN:S-0-1011.0.0 (w), Result:00000000, Len:2, Data: [22] [00]
Station:2, IDN:S-0-1011.0.0 (w), Result:00000000, Len:2, Data: [1e] [00]
Station:1, IDN:S-0-1011.0.0 (w), Result:00000000, Len:2, Data: [1a] [00]
Station:3, IDN:S-0-1010.0.0 (w), Result:00000000, Len:8, Data: [3a] [00] [00] …
Station:2, IDN:S-0-1010.0.0 (w), Result:00000000, Len:8, Data: [3a] [00] [00] …
Station:1, IDN:S-0-1010.0.0 (w), Result:00000000, Len:8, Data: [3a] [00] [00] …
Station:1, IDN:S-0-1012.0.0 (w), Result:00000000, Len:8, Data: [40] [00] [00] …
Station:2, IDN:S-0-1012.0.0 (w), Result:00000000, Len:8, Data: [40] [00] [00] …
Station:3, IDN:S-0-1012.0.0 (w), Result:00000000, Len:8, Data: [40] [00] [00] …

SetFgNrt
Station:1, IDN:S-0-1017.0.0 (w), Result:00000000, Len:8, Data: [00] [00] [00] …
Station:2, IDN:S-0-1017.0.0 (w), Result:00000000, Len:8, Data: [00] [00] [00] …
Station:3, IDN:S-0-1017.0.0 (w), Result:00000000, Len:8, Data: [00] [00] [00] …
Station:2, IDN:S-0-1504.0.20 (w), Result:00000000, Len:8, Data: [00] [00] [00] …

Station(1) (NXIO_100-RE/S3S) (NXIO_100-RE/S3S)
MDT-SVC, Offs:8, Index:0, Len:6
MDT-DEV, Offs:26, Index:0, Len:4
MDT-CON(0), Offs:38, Index:0, Len:6
AT-SVC, Offs:8, Index:0, Len:6
AT-DEV, Offs:26, Index:0, Len:4
AT-CON(0), Offs:38, Index:0, Len:6

Station(2) (R-ILB S3 AI4 AO2) (R911170874)
MDT-SVC, Offs:14, Index:0, Len:6
MDT-DEV, Offs:30, Index:0, Len:4
MDT-CON(0), Offs:44, Index:0, Len:8
AT-SVC, Offs:14, Index:0, Len:6
AT-DEV, Offs:30, Index:0, Len:4
AT-CON(0), Offs:44, Index:0, Len:12

Station(3) (ILB S3 24 DI16 DIO16-2TX) (2897570)
MDT-SVC, Offs:20, Index:0, Len:6
MDT-DEV, Offs:34, Index:0, Len:4
MDT-CON(0), Offs:52, Index:0, Len:6
AT-SVC, Offs:20, Index:0, Len:6
AT-DEV, Offs:34, Index:0, Len:4
AT-CON(0), Offs:56, Index:0, Len:8

IDN:S-0-0127.0.0
IDN:S-0-0127.0.0
IDN:S-0-0127.0.0
```

SYBERA Copyright © 2010

```
IDN:S-0-0128.0.0
IDN:S-0-0128.0.0
IDN:S-0-0128.0.0

RunFlag:1, ErrFlag:1
Sys:0, Phy:0, RxFifo:2, RxOvfl:2, RxCrc:2, Tx:0

RunFlag:0, ErrFlag:0
Sys:0, Phy:0, RxFifo:0, RxOvfl:0, RxCrc:0, Tx:0
```

SYBERA Copyright © 2010

4 SERCOS III Library Interface

The header file SHASC3CORE.H defines all required prototypes and parameters of the Ethernet Core Library. 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 and Borland C++ Builder

4.1.1 Sha(32/64)Sc3GetVersion

This function retrieves the version information strings of the SERCOS III Master Library, the Ethernet Core Library, the Ethernet Core Driver, the SHA Dll, the SHA Library and the SHA Driver. The memory for the information strings must be allocated first.

VC
```
ULONG Sha(32/64)(32/64)Sc3GetVersion (PSC3_PARAMS);
```

Sample:

```
//Display version information
Sha(32/64)(32/64)Sc3GetVersion(&Sc3Params);
printf("ECTCORE-DLL : %.2f\nECTCORE-DRV : %.2f\n",
      Sc3Params.core_dll_ver / (double)100,
      Sc3Params.core_drv_ver / (double)100);

printf("ETHCORE-DLL : %.2f\nETHCORE-DRV : %.2f\n",
      Sc3Params.EthParams[0].core_dll_ver / (double)100,
      Sc3Params.EthParams[0].core_drv_ver / (double)100);

printf("SHA-LIB     : %.2f\nSHA-DRV     : %.2f\n",
      Sc3Params.EthParams[0].sha_lib_ver / (double)100,
      Sc3Params.EthParams[0].sha_drv_ver / (double)100);
```

SYBERA Copyright © 2010

4.1.2 Sha(32/64)Sc3Create

This function initializes the SERCOS III Realtime and Station Management. On success the returning value is ERROR_SUCCESS, otherwise the returning value corresponds to that with GetLastError().

VC `ULONG Sha(32/64)Sc3Create (PSC3_PARAMS);`

Sample:

```
//Required SC3 parameters
memset(&Sc3Params, 0, sizeof(SC3_PARAMS));
Sc3Params.EthParams[0].dev_num = 0;             //Set first NIC device
Sc3Params.EthParams[1].dev_num = 1;             //Set second NIC device
      (optional)
Sc3Params.EthParams[0].fpAppTask = AppTask;     //Set realtime application task
Sc3Params.EthParams[1].fpAppTask = NULL;        //No second application task
Sc3Params.CyclePeriod = 2000;                   //Set cycle period [µsec]
      (optional)
Sc3Params.bManualConfig = TRUE;                 //Set manual configuration flag

//Enable SC3 realtime core
if (ERROR_SUCCESS == Sha(32/64)Sc3Create(&Sc3Params))
{
    //Init global ethernet elements
    __ppUserStack[0]   = Sc3Params.EthParams[0].pUserStack;   //Nic0 User Stack
    __ppUserStack[1]   = Sc3Params.EthParams[1].pUserStack;   //Nic1 User Stack
    __ppSystemStack[0] = Sc3Params.EthParams[0].pSystemStack; //Nic0 System
    Stack
    __ppSystemStack[1] = Sc3Params.EthParams[1].pSystemStack; //Nic1 System
    Stack
    __StackNum         = Sc3Params.StackNum;                  //Number of Nic
    Stacks
    __pUserList        = Sc3Params.pUserList;                 //User Station
    List
    __pSystemList      = Sc3Params.pSystemList;               //System Station
    List
    __fpSc3Enter       = Sc3Params.fpSc3Enter;                //SC3 Enter
    Function
    __fpSc3Exit        = Sc3Params.fpSc3Exit;                 //SC3 Exit
    Function
}
```

Note: After calling *Sha(32/64)Sc3Create* the realtime task (with SERCOS III wrapper) is running and a station list has been created. If the flag *bManualConfig* is set, the sercos III system must be initialized manually by IDN service functions. If the flag *bManualConfig* is cleared (default), the sercos III system is automatically initialized by the function *Sha(32/64)Sc3Enable* and starts up until phase 4.

SYBERA Copyright © 2010

SYBERA Copyright © 2010

4.1.3 Sha(32/64)Sc3Destroy

This function closes the SERCOS III communication.

VC
```
ULONG Sha(32/64)Sc3Destroy(PSC3_PARAMS);
```

4.1.4 Sha(32/64)Sc3Enable

This function initializes the SERCOS III station list upto Phase4 and must follow the function Sha(32/64)Sc3Create.

VC
```
ULONG Sha(32/64)Sc3Enable(PSC3_PARAMS);
```

Note: Following library functions are implemented by this function:

- Sc3InitStationList()
- Sc3ConfigureStationList()
    - Sc3ChangeToPhaseCP0()
    - Sc3ChangeToPhaseCP1()
    - Sc3ChangeToPhaseCP2()
    - Sc3ReadIdn(…)
    - Sc3WriteIdn(…)
    - Sc3ReadIdnAsync(…)
    - Sc3WriteIdnAsync(…)
    - Sc3WaitForAllIdnsAsync(…)
- Sc3EnableStationList()
    - Sc3ExecProcedure(…)
    - Sc3ChangeToPhaseCP3()
    - Sc3ChangeToPhaseCP4()

SYBERA Copyright © 2010

4.1.5 Sha(32/64)Sc3Disable

This function disables the SERCOS III station list

VC `ULONG Sha(32/64)Sc3Disable(PSC3_PARAMS);`

Sample:

```
//*****************************************************************
//
//    This code is strictly reserved by SYBERA. It´s used only for
//    demonstration purposes. Any modification or integration
//    isn´t allowed without permission by SYBERA.
//
//  Copyright (c) 2010 SYBERA
//
//  Tis code uses following configuration:
//
//    Station 0, Address  0 : Sybera SERCOS III Master
//    Station 1, Address 10 : ILB S3 24 DI16 DIO16-2TX
//    Station 2, Address 20 : ILB S3 AI4 AO2
//
//*****************************************************************


#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include "c:\sha\shaexp.h"
#include "c:\eth\EthCoreDef.h"
#include "c:\eth\EthMacros.h"
#include "..\..\..\Inc\vc_cb\Sc3CoreDef.h"
#include "..\..\..\Inc\vc_cb\Sc3Macros.h"
#include "..\..\..\Inc\vc_cb\Sc3IdnDef.h"
#include "..\..\..\Inc\vc_cb\Sha(32/64)Sc3Core.h"

//Declare global elements
PETH_STACK  __ppUserStack[MAX_STACK_NUM]   = { NULL, NULL }; //Ethernet Stack
                                                             //List
PETH_STACK  __ppSystemStack[MAX_STACK_NUM] = { NULL, NULL }; //Ethernet Stack
                                                             //List)
ULONG       __StackNum = 0;               //Number of Ethernet Stacks
PSTATION_INFO     __pUserList = NULL;     //Station List (used outside Realtime
                                          //Task)
PSTATION_INFO     __pSystemList = NULL;   //Station List (used inside Realtime
                                          //Task)
ULONG             __StationNum = 0;       //Number of Stations
ULONG             __Phase = 0;            //Current communication phase
ULONG             __Error = 0;            //Error condition
FP_SC3_ENTER      __fpSc3Enter = NULL;    //Function pointer to Wrapper Sc3Enter
```

SYBERA Copyright © 2010

```
FP_SC3_EXIT       __fpSc3Exit = NULL;     //Function pointer to Wrapper Sc3Exit
ULONG             __Sc3State = SC3_STATE_STOP;  //Initial Wrapper State
ULONG             __ReadyCnt = 0;               //Station Update Counter
ULONG             __CycleCnt = 0;               //Realtime Cycle Counter
char              __ActChar[4] = { '\\', '|', '/', '-' };
USHORT            __DiData = 0;
USHORT            __DoData = 0;

//Init Task Timer (usec units)
INIT_TASK_TIMER(100, 1000, 1000, FALSE);


void static AppTask(void)
{
      PSTATION_INFO pStation;
      BOOLEAN bResult;

      //Call SC3 enter function
      __Sc3State = __fpSc3Enter(
                        &__StationNum,    //Out: current station number
                        &__Phase,         //Out: current phase
                        &__Error);        //Out: error condition

      //Check normal operation state
      if (__Sc3State == SC3_STATE_READY)
      {
            //DBG_INITIAL_BREAK();

            //Check ready count to toggle outputs
            if (__ReadyCnt++ == 0)
            {
                  //*********************************
                  //Do the logical station operation
                  if ((__pSystemList) &&
                        (__StationNum))
                  {
                        //Get station due to address
                        pStation = __Sc3GetStation(__pSystemList, __StationNum,
      30);
                        if (pStation)
                        {
                              //Get IO inputs and set IO outputs (pStation,
      Offset,
                              //DataSize, pData)
                              __Sc3GetIoData(pStation, 2, 2, (PUCHAR)&__DiData);
                              __Sc3SetIoData(pStation, 0, 2, (PUCHAR)&__DoData);
                        }

                        //Check timer intervall and increase digital outputs
                        CHECK_TASK_TIMER(&bResult)
                        if (bResult) { __DoData++; }
                  }
                  //*********************************
            }
      }
      else
            __ReadyCnt = 0;
```

SYBERA Copyright © 2010

```
	//Call SC3 exit function
	__fpSc3Exit();

	//Increase loop count
	__CycleCnt++;
}


void main(void)
{
	SC3_PARAMS Sc3Params;
	ULONG c = 0;

	printf("\n*** SERCOS III Phase 4 ***\n\n");

	//Required SC3 parameters
	memset(&Sc3Params, 0, sizeof(SC3_PARAMS));
	Sc3Params.EthParams[0].dev_num = 0;	//Set first NIC device
	Sc3Params.EthParams[1].dev_num = 1;	//Set second NIC device
	(optional)
	Sc3Params.EthParams[0].fpAppTask = AppTask;	//Set realtime application
	task
	Sc3Params.EthParams[1].fpAppTask = NULL;	//No second application
	task
	Sc3Params.CyclePeriod = 2000;			//Set cycle period [µsec]
							//(optional)

	//*****************************
	//Create SC3 realtime core
	//*****************************
	if (ERROR_SUCCESS == Sha(32/64)Sc3Create(&Sc3Params))
	{
		//Init global ethernet elements
		__ppUserStack[0]	= Sc3Params.EthParams[0].pUserStack;
		__ppUserStack[1]	= Sc3Params.EthParams[1].pUserStack;
		__ppSystemStack[0]	= Sc3Params.EthParams[0].pSystemStack;
		__ppSystemStack[1]	= Sc3Params.EthParams[1].pSystemStack;
		__StackNum		= Sc3Params.StackNum;
		__pUserList		= Sc3Params.pUserList;
		__pSystemList		= Sc3Params.pSystemList;
		__fpSc3Enter		= Sc3Params.fpSc3Enter;
		__fpSc3Exit		= Sc3Params.fpSc3Exit;

		//Display ethernet stack information
		printf("Number of Ethernet Stacks: %i\n\n", __StackNum);

		//Display version information
		Sha(32/64)Sc3GetVersion(&Sc3Params);
		printf("SC3CORE-DLL : %.2f\nSC3CORE-DRV : %.2f\n",
			Sc3Params.core_dll_ver / (double)100,
			Sc3Params.core_drv_ver / (double)100);
		printf("ETHCORE-DLL : %.2f\nETHCORE-DRV : %.2f\n",
			Sc3Params.EthParams[0].core_dll_ver / (double)100,
			Sc3Params.EthParams[0].core_drv_ver / (double)100);
		printf("SHA-LIB     : %.2f\nSHA-DRV     : %.2f\n",
			Sc3Params.EthParams[0].sha_lib_ver / (double)100,
			Sc3Params.EthParams[0].sha_drv_ver / (double)100);
		printf("\n");
```

SYBERA Copyright © 2010

```
            //*****************************
            //Enable Stations
            //*****************************
            if (ERROR_SUCCESS == Sha(32/64)Sc3Enable(&Sc3Params))
            {
                  //Display station information
                  for (ULONG i=0; i<__StationNum; i++)
                        printf("Station:%i, Addr:%3i ,Name:%6s\n",
                        i, __pUserList[i].Addr, __pUserList[i].szName);

                  //Wait for key pressed
                  printf("\nPress any key to exit ...\n");
                  while (!kbhit())
                  {
                        //Check SC3 condition
                        printf("StationNum:%i, Phase:%i, Error:%i, DiData:%04x
                              [%i %c]\r", __StationNum, __Phase, __Error,
                              __DiData, __CycleCnt, __ActChar[++c%4]);

                        //Do some delay
                        Sleep(100);
                  }

                  //Disable Stations
                  Sha(32/64)Sc3Disable(&Sc3Params);
            }
            //Destroy SC3 core
            Sha(32/64)Sc3Destroy(&Sc3Params);
      }
      //Wait for key pressed
      printf("\nPress any key to exit ...\n");
      while (!kbhit())
      {
            //Check SC3 condition
            printf("Phase: %i, Condition: %i  [%c]\r", __Phase, __Error,
      __ActChar[++c%4]);

            //Do some delay
            Sleep(100);
      }
}
```

SYBERA Copyright © 2010

5 Realtime Operation

After creating the SERCOS III system (*Sha(32/64)Sc3Create*) with a station list, the realtime tasks become active. The application realtime task is decorated by Realtime SERCOS III Wrapper functions:

```
//Call SC3 enter function (Return: wrapper state)
__Sc3State = __fpSc3Enter(
                        &__StationNum,    //Out: current station number
                        &__Phase,         //Out: current phase
                        &__Error);        //Out: current error condition


typedef ULONG     (__cdecl *FP_SC3_ENTER)(PULONG, PULONG, PULONG);
typedef VOID      (__cdecl *FP_SC3_EXIT) (VOID);
```

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

```
//Define SC3 Wrapper States
enum _SC3_STATE
{
      SC3_STATE_INIT = 0,           //0 - Initial state
      SC3_STATE_BUSY,               //1 - Ethernet currently busy
      SC3_STATE_UPDATE,             //2 - Station update still in progress
      SC3_STATE_READY,              //3 - All stations are updated
      SC3_STATE_ERROR,              //4 - An update error occured
      SC3_STATE_PRESTOP,            //5 - Reserved for internal use
      SC3_STATE_STOP = 100          //100 - Realtime Wrapper is stopped
};
```

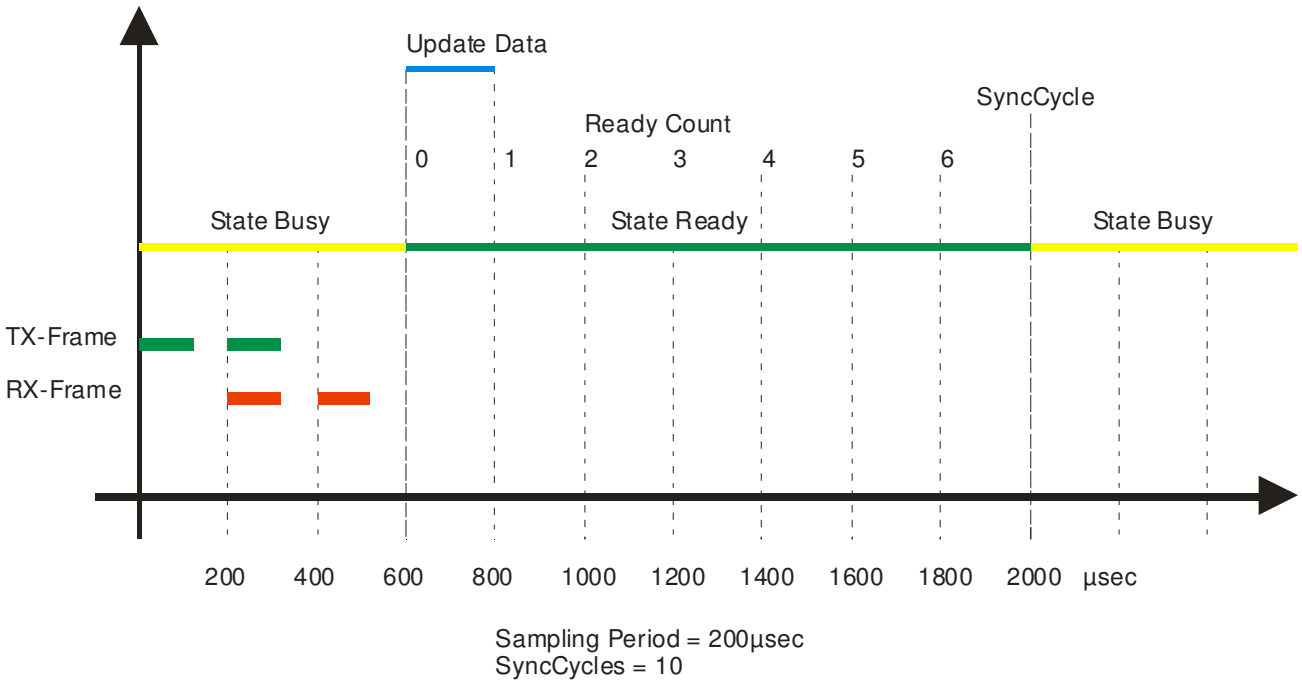
SYBERA Copyright © 2010

The Wrapper Functions require as parameters the Ethernet Core Stack pointer (e.g.__pSystemStack), the station list pointer (e.g.__pSystemList) and the number of stations (e.g. __StationNum). These parameters and others are returned when initializing the SERCOS III Realtime Library and set as global elements.

```
//Declare global elements
PETH_STACK  __ppUserStack[MAX_STACK_NUM]   = { NULL, NULL }; //Ethernet Stack
                                                             //List (used outside
                                                             //Realtime Task)
PETH_STACK  __ppSystemStack[MAX_STACK_NUM] = { NULL, NULL }; //Ethernet Stack
                                                             //List (used inside
                                                             //Realtime Task)
ULONG            __StackNum = 0;         //Number of Ethernet Stacks
PSTATION_INFO    __pUserList = NULL;     //Station List (used outside Realtime
                                         //Task)
PSTATION_INFO    __pSystemList = NULL;   //Station List (used inside Realtime
                                         //Task)
ULONG            __StationNum = 0;       //Number of Stations
ULONG            __Phase = 0;            //Current communication phase
ULONG            __Error = 0;            //Error condition
FP_SC3_ENTER     __fpSc3Enter = NULL;    //Function pointer to Wrapper Sc3Enter
FP_SC3_EXIT      __fpSc3Exit = NULL;     //Function pointer to Wrapper Sc3Exit
ULONG            __Sc3State = SC3_STATE_STOP;   //Initial Wrapper State
```

SYBERA Copyright © 2010

The Realtime Wrapper itself returns the wrapper state with each sampling period. When the Wrapper indicate the state SC3_STATE_READY it means, that all stations are updated and the current number of stations (May depend on HotPlug), the current phase and the current error condition (resetting on phase switch) are returned. Within one synchronisation cycle (e.g. 2msec), the data should updated just once. Since in the following sample the state SC3_STATE_READY would last 7 sampling periods, its useful to keep track by a ready counter and update the data just once within one synchronisation cycle:



Sampling Period = 200µsec
SyncCycles = 10

SYBERA Copyright © 2010

```
void static AppTask(void)
{
      PSTATION_INFO pStation;

      //Call SC3 enter function
      __Sc3State = __fpSc3Enter(
                              &__StationNum,    //Out: current station number
                              &__Phase,         //Out: current phase
                              &__Error);        //Out: error condition

      //Check normal operation state
      if (__Sc3State == SC3_STATE_READY)
      {
            //Check ready count to toggle outputs
            if (__ReadyCnt++ == 0)
            {
                  //*********************************
                  //Do the logical station operation
                  if ((__pSystemList) &&
                      (__StationNum)  &&
                      (__Phase == PHASE_CP4)))
                  {
                        //Get station due to address
                        pStation = __Sc3GetStation(__pSystemList, __StationNum,
      30);
                        if (pStation)
                        {
                              //Get IO inputs and set IO outputs
                              //(pStation, Offset, DataSize, pData)
                              __Sc3GetIoData(pStation, 2, 2, (PUCHAR)&__DiData);
                              __Sc3SetIoData(pStation, 0, 2, (PUCHAR)&__DoData);
                        }
                  }
                  //*********************************
            }
      }
      else
            //Reset ready count
            __ReadyCnt = 0;

      //Call SC3 exit function
      __fpSc3Exit();

      //Increase loop count
      __CycleCnt++;
}
```

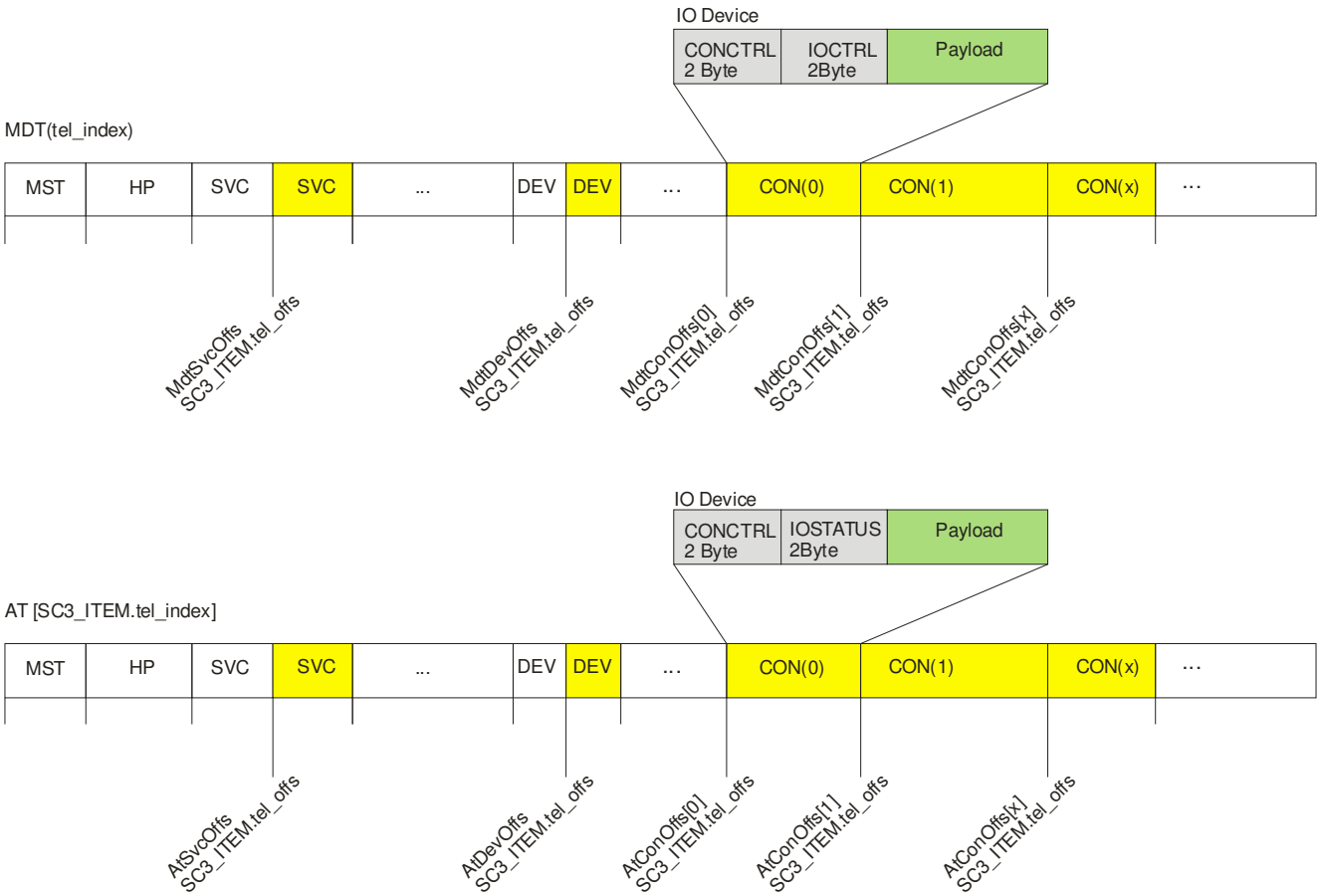
SYBERA Copyright © 2010

5.1 Realtime Data Access

The realtime data access is provided by the *SC3_ITEM* elements in the structure *STATION_INFO*. To get easy payload access for IO devices, the macros *__Sc3GetIoData* and *__Sc3SetIoData* can be used. This macro also toggles automatically the *new_data* flag within the IO control structure.

```
typedef struct _SC3_ITEM_OFFS
{
      USHORT tel_offs  : 12;  //Telegram Offset
      USHORT tel_index : 2;   //Telegram Index
      USHORT reserved  : 2;

} SC3_ITEM_OFFS, *PSC3_ITEM_OFFS;
```



SYBERA Copyright © 2010

6 SERCOS III Library LowLevel Interface

The SERCOS III Library LowLevel Interface provides all functions to control SERCOS III slave devices in detail. Several LowLevel Interface groups are provided by this library.

6.1 SERCOS III LowLevel Command Functions

The LowLevel command functions allow phase handling and other base functionality of the SERCOS III protocol.

6.1.1 Read SERCOS III Station Address

This command allows to reads the address of a SERCOS III station.

```
ULONG Sc3ReadStationAddress(
                        ULONG StationIndex,
                        PUSHORT pAddr)
```

Note:
This command is equal to the following command:

```
//Set service IDN
SC3_IDN Idn;
__SetIDN(&Idn, IDN_TYPE_STD, 0, 1040, 0, 0);

//Read IDN service
Sc3ReadIdn(StationIndex, &Idn, NULL, (PUCHAR)pAddr, sizeof(USHORT), NULL);
```

6.1.2 Write SERCOS III Station Address

This command allows to write the address of a SERCOS III station.

```
ULONG Sc3WriteStationAddress(
                        ULONG StationIndex,
                        USHORT Addr)
```

Note:
This command is equal to the following command:

```
//Set service IDN
SC3_IDN Idn;
__SetIDN(&Idn, IDN_TYPE_STD, 0, 1040, 0, 0);

//Write IDN service
Sc3WriteIdn(StationIndex, &Idn, NULL, (PUCHAR)&Addr, sizeof(USHORT));
```

SYBERA Copyright © 2010

6.1.3 Change to Phase CP0

This command allows to change to phase CP0. The command may be called at any time.

```
ULONG Sc3ChangeToPhaseCP0(VOID)
```

6.1.4 Change to Phase CP1

This command allows to change to phase CP1. The command must follow the phase CP0.

```
ULONG Sc3ChangeToPhaseCP1(VOID)
```

6.1.5 Change to Phase CP2

This command allows to change to phase CP2. The command must follow the phase CP1.

```
ULONG Sc3ChangeToPhaseCP2(VOID)
```

6.1.6 Change to Phase CP3

This command allows to change to phase CP3. The command must follow the phase CP2.

```
ULONG SHAAPI Sc3ChangeToPhaseCP3(VOID)
```

6.1.7 Change to Phase CP4

This command allows to change to phase CP4. The command must follow the phase CP3.

```
ULONG SHAAPI Sc3ChangeToPhaseCP3(VOID)
```

SYBERA Copyright © 2010

6.2 SERCOS III LowLevel Synchronous Service Functions

The LowLevel synchronous service functions allow IDN handling for each station separately (synchronously).

6.2.1 Write IDN

This command allows to write a IDN service to a SERCOS III station and returns the service result. The IDN service commands check automatically for variable or fixed size data, and returns (optionally) the data attribute.

```
ULONG Sc3WriteIdn(
                  ULONG StationIndex,
                  PSC3_IDN pIdn,
                  PSC3_ATTRIB pAttrib,
                  PUCHAR pBuffer,
                  ULONG Len)
```

Sample:

```
//Write station address
__SetIDN(&Idn, IDN_TYPE_STD, 0, 1040, 0, 0);
ULONG Result = Sc3WriteIdn(i, &Idn, NULL, (PUCHAR)&StationAddr, sizeof(USHORT));
```

6.2.2 Read IDN

This command allows to read a IDN service from a SERCOS III station and returns the service result. The IDN service commands check automatically for variable or fixed size data, and returns (optionally) the data attribute and the number of bytes read.

```
ULONG Sc3ReadIdn(
                  ULONG StationIndex,
                  PSC3_IDN pIdn,
                  PSC3_ATTRIB pAttrib,
                  PUCHAR pBuffer,
                  ULONG Len,
                  PULONG pBytesRead)
```

Sample:

```
//Read component name
__SetIDN(&Idn, IDN_TYPE_STD, 0, 1300, 0, 1);
ULONG Result = Sc3ReadIdn(i, &Idn, NULL, (PUCHAR)szCompName, MAX_PATH_SIZE,
      NULL);
```

SYBERA Copyright © 2010

6.2.3 Write Fixed Size Data Block Element

This command allows to write a fixed size data block element and returns the service result.

```
ULONG Sc3WriteFixedSizeDbe(
                        ULONG StationIndex,
                        USHORT Dbe,
                        PUCHAR pBuffer,
                        ULONG Len)
```

6.2.4 Write Variable Size Data Block Element

This command allows to write a variable size data block element and returns the service result.

```
ULONG Sc3WriteVariableSizeDbe(
                        ULONG StationIndex,
                        USHORT Dbe,
                        PUCHAR pBuffer,
                        ULONG Len)
```

6.2.5 Read Fixed Size Data Block Element

This command allows to read a fixed size data block element and returns the service result.

```
ULONG Sc3ReadFixedSizeDbe(
                        ULONG StationIndex,
                        PSC3_IDN pIdn,
                        USHORT Dbe,
                        PUCHAR pBuffer,
                        ULONG Len)
```

6.2.6 Read Variable Size Data Block Element

This command allows to read a variable size data block element and returns the service result.

```
ULONG Sc3ReadVariableSizeDbe(
                        ULONG StationIndex,
                        PSC3_IDN pIdn,
                        USHORT Dbe,
                        PUCHAR pBuffer,
                        PULONG pLen)
```

SYBERA Copyright © 2010

6.2.7 Transmit Service

This command allows to transmit a SERCOS III service directly and returns the service result. This command is used for extending LowLevel functionality.

```
ULONG Sc3TransmitService(
                        ULONG StationIndex,
                        PSC3_SVC_CTRL pSvcCtrl,
                        PSC3_SVC_STAT pSvcStat)
```

Sample:

```
SC3_SVC_CTRL SvcCtrl;
SC3_SVC_STAT SvcStat;
TYPE32 Info;
ULONG Result;
BOOLEAN bLast = TRUE;

//Set Service Control  HS     WR     LT     ET   INFO
__SetSvcCtrl(&SvcCtrl, FALSE, FALSE, bLast, Dbe, pIdn->bit32);
Result = Sc3TransmitService(StationIndex, &SvcCtrl, &SvcStat);
__GetSvcStat(&SvcStat, NULL, NULL, NULL, NULL, &Info.bit32);
//Get Service Status   HS    BUSY  ERR   VAL   INFO
```

SYBERA Copyright © 2010

6.3 SERCOS III LowLevel Asynchronous Service Functions

The LowLevel synchronous service functions allow IDN handling of multiple stations in parallel (asynchronously). Asynchronous IDN handling reduces the startup time enormously. Therefor the program sequence must synchronize to the service results.

6.3.1 Write IDN asynchronously

This command allows to write IDN services in parallel to SERCOS III stations. and return the operation result (not the service result). The IDN service commands check automatically for variable or fixed size data and returns the data attribute (optionally, after synchronisation, the pointer value is valid).

```
ULONG Sc3WriteIdnAsync(
                  ULONG StationIndex,
                  PSC3_IDN pIdn,
                  PSC3_ATTRIB pAttrib,
                  PUCHAR pBuffer,
                  ULONG Len)
```

6.3.2 Read IDN asynchronously

This command allows to read IDN services in parallel from SERCOS III stations. and return the operation result (not the service result). The IDN service commands check automatically for variable or fixed size data and returns the data attribute (optionally, after synchronisation, the pointer value is valid).

```
ULONG Sc3ReadIdnAsync(
                  ULONG Index,
                  PSC3_IDN pIdn,
                  PSC3_ATTRIB pAttrib,
                  PUCHAR pBuffer,
                  ULONG Len,
                  PULONG pBytesRead)
```

SYBERA Copyright © 2010

6.3.3 Wait for a single IDN asyncronously

This command waits for a single service (by a station index) being completed asynchronously. This function returns the operation result and the service error (optionally).

```
ULONG Sc3WaitForIdnAsync(
                        ULONG Index,
                        PULONG pError)
```

6.3.4 Wait for all IDNs asyncronously

This command waits for all asynchronously started services being completed. This function returns the operation result and a service error list (optionally).

```
ULONG Sc3WaitForAllIdnsAsync(
                        ULONG Offs,
                        PULONG pErrorList)
```

Sample:

```
SC3_IDN Idn;
ULONG ErrorList[MAX_STATION_NUM];
ULONG Result = ERROR_SUCCESS;
ULONG i;

//Loop through all slave stations (offs = 1, since station 0 is the master
      station)
for (i=1; i<__StationNum; i++)
{
      //Read station name asynchronous
      __SetIDN(&Idn, IDN_TYPE_STD, 0, 1300, 0, 4);
      if (Result == SC3_ERROR_SUCCESS)
          Result = Sc3ReadIdnAsync(i, &Idn, NULL, (PUCHAR)__pUserList[i].szName,
                                   MAX_PATH_SIZE, NULL);
}

//Wait for completion of all IDNs (offs = 1, since station 0 is the master
      station)
Result = Sc3WaitForAllIdnsAsync(1, ErrorList);
```

SYBERA Copyright © 2010

6.4 SERCOS III LowLevel Procedure Functions

The LowLevel functions allow handling of procedure services (synchronously).

6.4.1 Write procedure service

This command allows to write a procedure service to a SERCOS III station and returns the service result.

```
ULONG Sc3WriteProcIdn(
                  ULONG StationIndex,
                  PSC3_IDN pIdn,
                  SC3_PROC_CTRL ProcCtrl,
                  PSC3_PROC_ACK pProcAck,
                  PBOOLEAN pbCmdChg)
```

6.4.2 Close procedure service

This command closes a procedure service of a SERCOS III station and returns the service result.

```
ULONG Sc3CloseProcIdn(
                  ULONG StationIndex,
                  PSC3_IDN pIdn)
```

6.4.3 Execute procedure service

This command executes a procedure service to a SERCOS III station with timeout and returns the service result.

```
ULONG Sc3ExecProcedure(
                  ULONG StationIndex,
                  PSC3_IDN pIdn,
                  ULONG TimeoutTicks)
```

Sample:

```
//Set CP3 transition check command
__SetIDN(&Idn, IDN_TYPE_STD, 0, 127, 0, 0);
if (SC3_ERROR_SUCCESS != Sc3ExecProcedure(i, &Idn, TIMEOUT_TICKS))
```

SYBERA Copyright © 2010

7 Device Configuration

Usualy device information is provided by a corresponding XML configuration file. Since the development of software with the SERCOS III Master Library has special needs for programming, the XML file must be parsed and translated into a native format. Therefore the SERCOS III Master Library provides a configuration file called **SC3CONFIG.PAR**, whitch is located in the directory **windows****system32** after installation. The SC3CONFIG.PAR is a text based file with sections for Device Name, Device ID, configuration parameters and data description. All sections of a device must follow the item order:

NAME -> DEVICEID -> PARAM -> CONFIG -> DESC

```
;*****************************************************
[NAME]
HCS01.1E-W0003-A-02-E-S3-EC-NN-NN-NN-FW

[DEVICEID]
FWA-INDRV*-MPE-16

[PARAM]
S-0-32.0.0 (02 00)

[CONFIG]
S-0-0134.0.0  <0,0>
S-0-0036.0.0  <0,0>
S-0-0047.0.0  <0,0>
S-0-0135.0.0  <1,1>
S-0-0040.0.0  <1,1>
S-0-0386.0.0  <1,1>

[DESC]
00 00 01 01 02 00
00 00 04 02 02 00
00 00 04 03 04 00
00 00 04 03 04 00
00 01 01 01 02 00
00 01 04 02 02 00
00 01 04 03 04 00
00 01 04 03 04 00
;*****************************************************
```

SYBERA Copyright © 2010

7.1 Section [NAME]

This section contains the name of the device:

```
[NAME]
HCS01.1E-W0003-A-02-E-S3-EC-NN-NN-NN-FW
```

7.2 Section [DEVICEID]

This section contains the device ID of the station:

```
[DEVICEID]
FWA-INDRV*-MPE-16
```

7.3 Section [PARAM]

This section contains the IDN and binary data for configuration:

```
[PARAM]
S-0-32.0.0 (02 00)
```

Meaning:

```
IDN  (Byte1, Byte2, … )
```

SYBERA Copyright © 2010

7.4 Section [CONFIG]

This section contains the connection IDN data (description of payload data) output/input data description of the device:

```
[CONFIG]
S-0-0134.0.0  <0,0>
S-0-0036.0.0  <0,0>
S-0-0047.0.0  <0,0>
S-0-0135.0.0  <1,1>
S-0-0040.0.0  <1,1>
S-0-0386.0.0  <1,1>
```

Meaning:

```
IDN  <Connection Index, Telegram Type>
```

7.5 Section [DESC]

This section contains the output/input data description of the device:

```
[DESC]
00 00 01 01 02 00
00 00 04 02 02 00
00 00 04 03 04 00
00 00 04 03 04 00
00 01 01 01 02 00
00 01 04 02 02 00
00 01 04 03 04 00
00 01 04 03 04 00
```

SYBERA Copyright © 2010

Meaning:

```
Telegram Type   00 : MDT (Output)
                01 : AT (Input)

Data Item       00 : DATA_ITEM_NONE
                01 : DATA_ITEM_CCON
                02 : DATA_ITEM_IOCTRL
                03 : DATA_ITEM_IOSTAT
                04 : DATA_ITEM_VALUE
                05 : DATA_ITEM_SCALE
                06 : DATA_ITEM_DIAG
                07 : DATA_ITEM_NAME

Data Type       00 : DATA_TYPE_NONE
                01 : DATA_TYPE_U8
                02 : DATA_TYPE_U16
                03 : DATA_TYPE_U32
                04 : DATA_TYPE_U64
                05 : DATA_TYPE_I8
                06 : DATA_TYPE_I16
                07 : DATA_TYPE_I32
                08 : DATA_TYPE_I64
                09 : DATA_TYPE_F32
                10 : DATA_TYPE_F64
```

```
Connection Index
   Telegram Type
      Data Item
         Data Type
            Data Len
|  |  |  |  |     |
 00 00 01 01 02 00
 00 00 04 02 02 00
 00 00 04 03 04 00
 00 00 04 03 04 00
 00 01 01 01 02 00
 00 01 04 02 02 00
 00 01 04 03 04 00
 00 01 04 03 04 00
```

SYBERA Copyright © 2010

8 Error Handling

The master library provides an error handling and tracing mechanism.

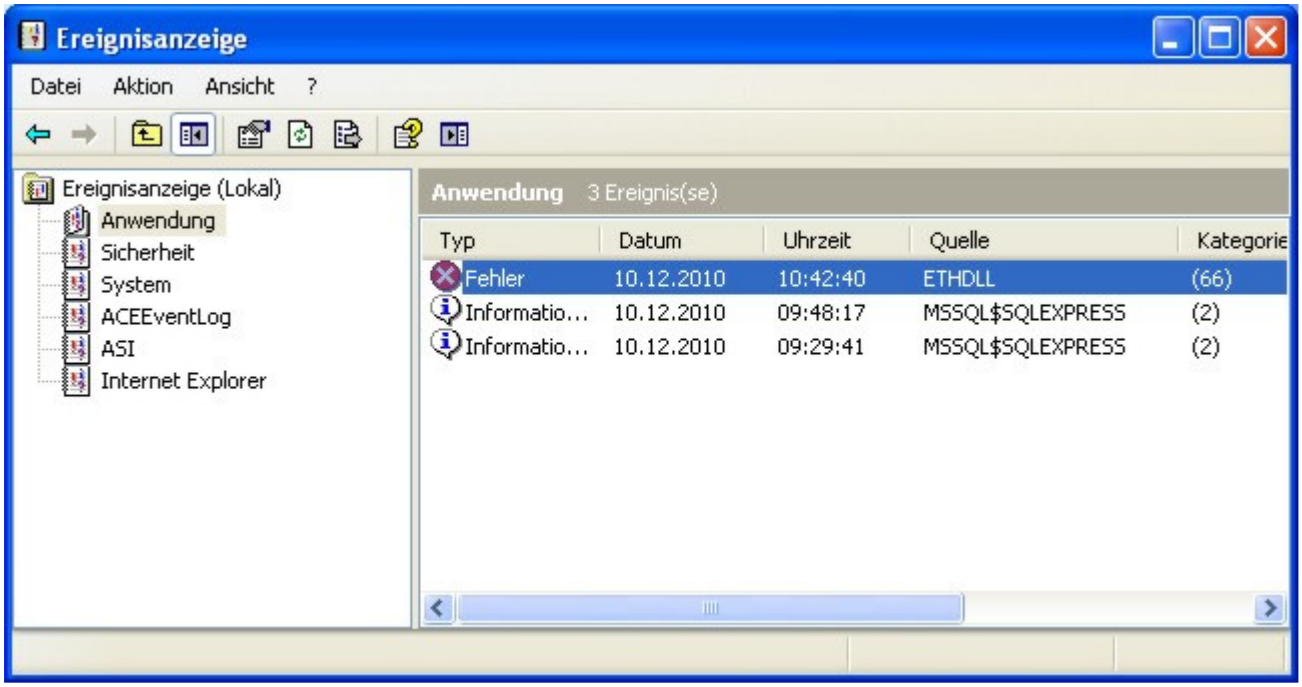
8.1 Debug LOG File

On execution the master library creates a sequence file SC3DBG.LOG in Text-Format

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

8.2 Event File

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



SYBERA Copyright © 2010

9 Related Dokuments

- manual_sha_e.pdf (SHA Realtime Library)
- manual_eth_e.pdf (ETH Realtime Library)