SYBERA

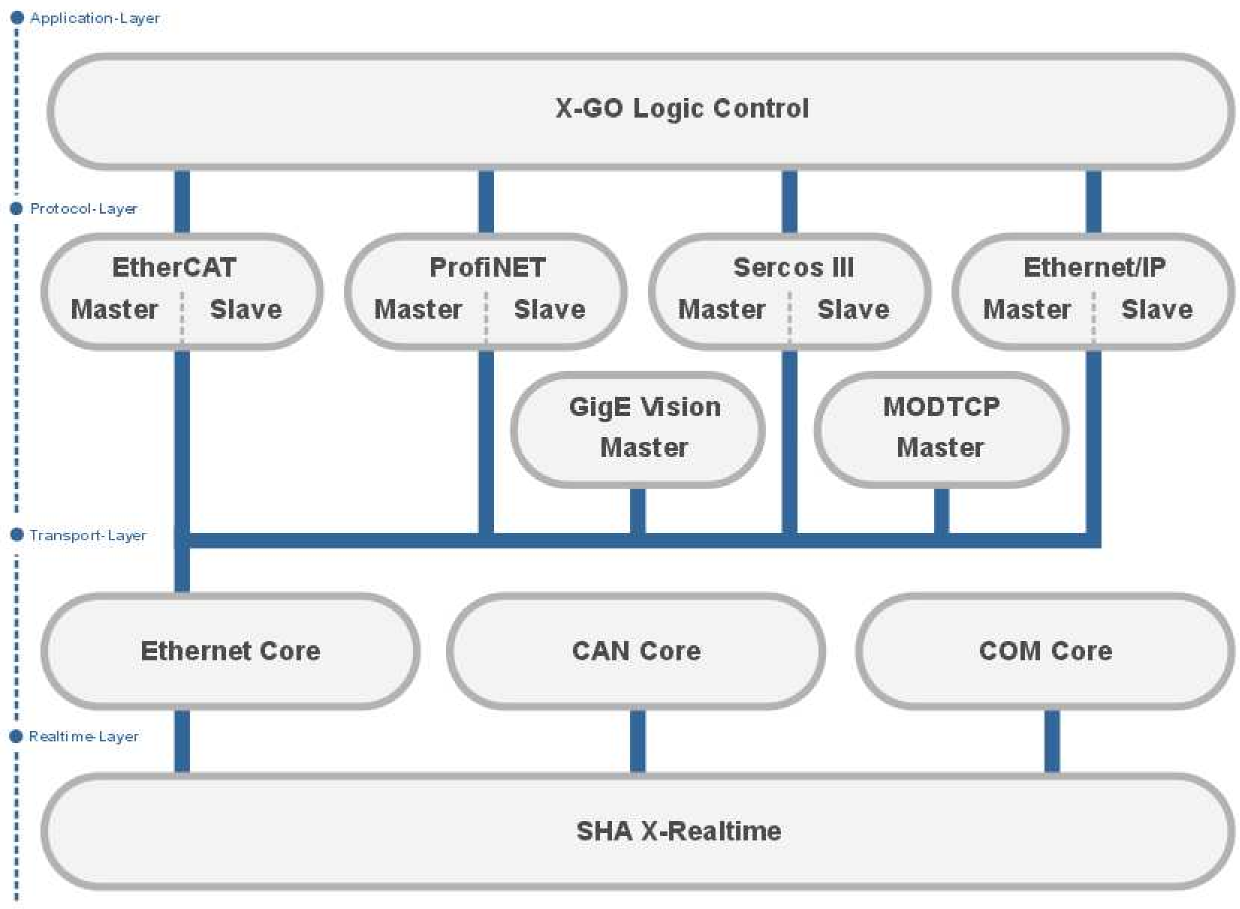
TM

CAN
Realtime Core Library
Documentation

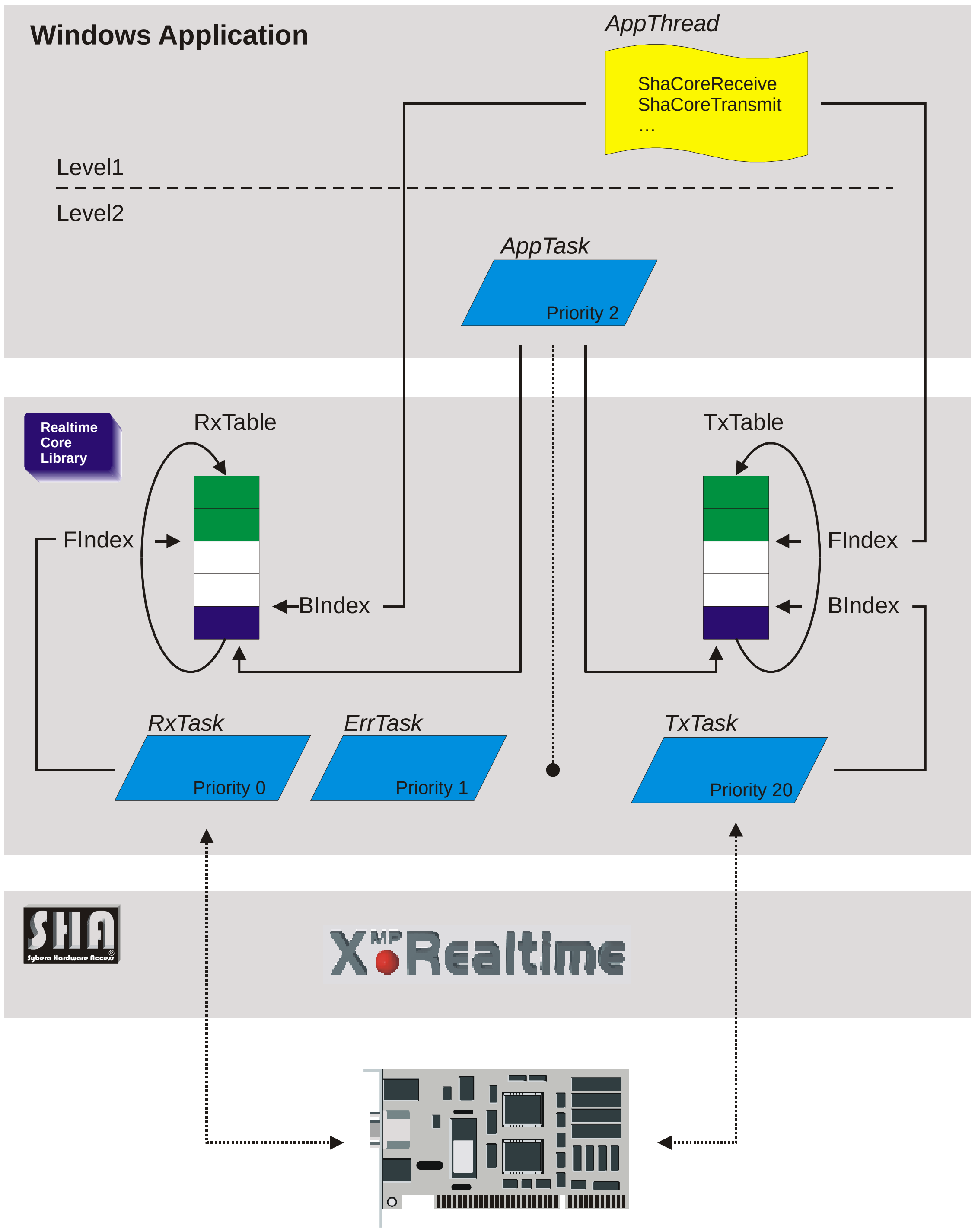
Date: Sept, 17.2015

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 intend-ed 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 funtional operation in realtime. Thus the RealtimeCores usually require simple passive harware. One of the great benefits is the adjustable scheduling time of incoming and outgoing data.



The AddOn realtime libraries allow handling of CAN data in realtime. At realtime level 1 incoming and outgoing data will be buffered inside RX and TX ringbuffers, controlled by forward and backward indexing. A simple proprietary core interface, as well as the standard COM interface is available for communication with the windows application.
Additional functional operation is possible at realtime level 2. Therefore a realtime task can be setup inside the application. The data exchanges is handled via shared memory area.



1.1 Supported Platforms

- VisualStudio C++
- Borland C++
- Borland Delphi
- CVI LabWindows

1.2 Supported OS

- Windows XP, VISTA, 7, 8, 10 (32/64 Bit)

1.3 Supported Hardware

- PCAN-PCI (Peak Systems)
- CPC-PCI (EMS)
- CPC-PCMCIA (EMS)
- PC-I04 (IXXAT)
- SYBCAN (SYBERA)

2 CAN Core Library Installation

For installation following steps are required:

Preparation

1. Provide a PC with corresponding CAN adapter and Windows operating system (with administrator privileges)

Installation

2. Install SHA realtime system (separate software package)

3. Install the RealtimeCore PORT-Driver

4. Run the program SYSETUP(32/64) of the master library (make sure the directory path has no space characters)

   On Installation the PEC information (PID, SERNUM and KEYCODE) must be entered. The PID for the evaluation version is 11223344, the SERNUM is 11223344, the KeyCode therefore is: 00001111-22223333

5. Optional: Check license with SYLICENCECHECK(32/64).EXE

Operation

6. Run SYCOMM(32/64).EXE (with admistrator privileges)

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: In order to operate SYBERA software under Windows 10, 8, 7, VISTA, it must be carried out with ADMINISTRATOR priviledges.

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.

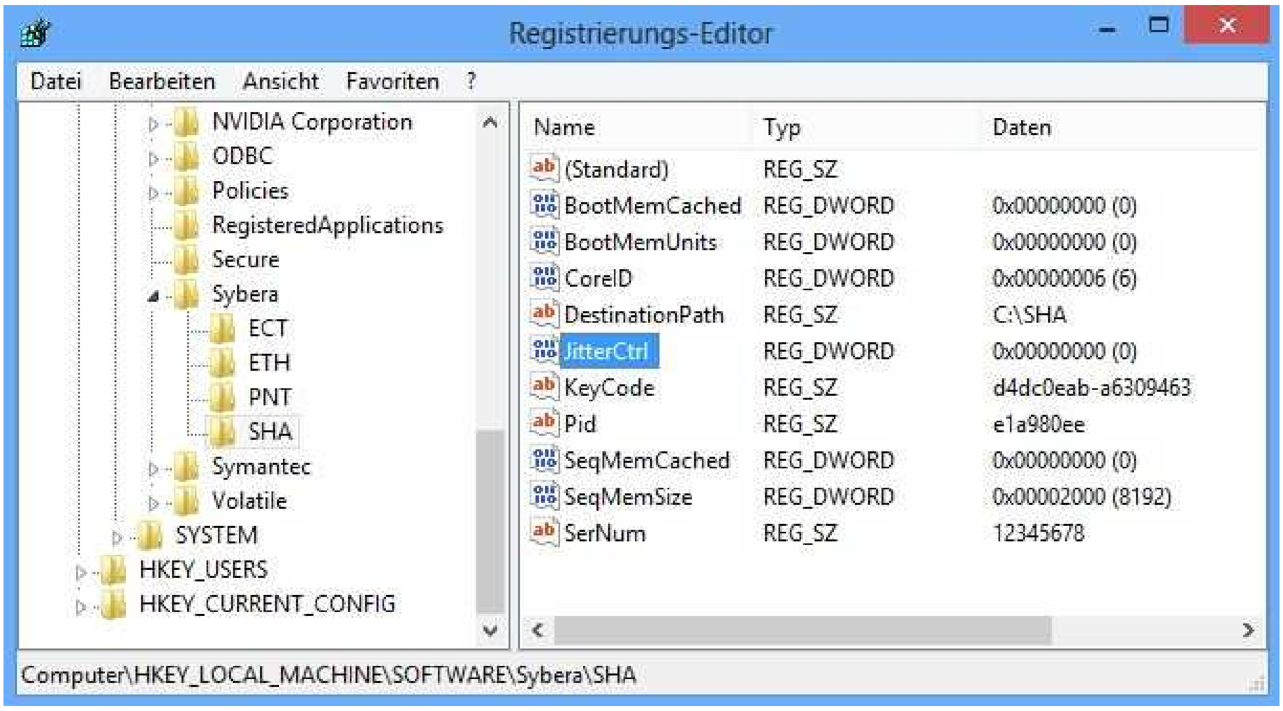
Enhanced SpeedStep — SpeedStep also modulates the CPU clock speed and voltage according to load, but it is invoked via another mechanism. The operating system must be aware of SpeedStep, as must the system BIOS, and then the OS can request frequency changes via ACPI. SpeedStep is more granular than C1E 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.

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 note-book.



Name	Typ	Daten
(Standard)	REG_SZ	
BootMemCached	REG_DWORD	0x00000000 (0)
BootMemUnits	REG_DWORD	0x00000000 (0)
CoreID	REG_DWORD	0x00000006 (6)
DestinationPath	REG_SZ	C:\SHA
JitterCtrl	REG_DWORD	0x00000000 (0)
KeyCode	REG_SZ	d4dc0eab-a6309463
Pid	REG_SZ	e1a980ee
SeqMemCached	REG_DWORD	0x00000000 (0)
SeqMemSize	REG_DWORD	0x00002000 (8192)
SerNum	REG_SZ	12345678

Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Sybera\SHA

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)

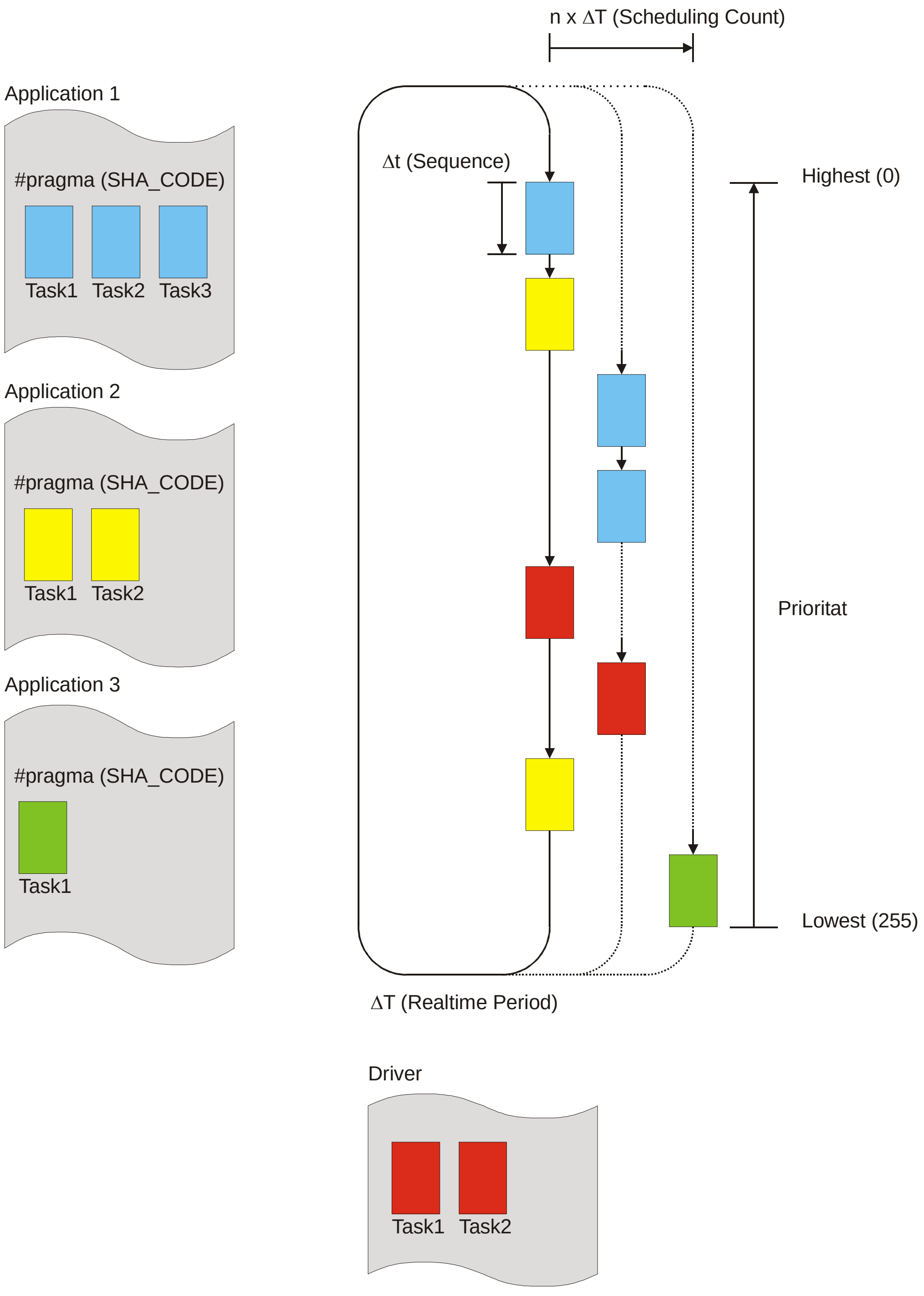
With the X-Realtime Engine, realtime task cycles are realizable upto 10 µsec (100 KHz) sampling rate. An integrated watchdog-system controls the realtime task and determines the remaining task-time. The SHA X-Failsafe-System offers additionally the possibility to keep a rescue task busy or to proceed a controlled shutdown, even on heavy exception errors (for example Blue-Screen). With the X-Failsafe-System, for example a robot-arm can be driven out from a hazard zone and an alarm signal is caused.

The realtime routine has to be equal to a RING0 EXECUTION routine for interrupt control (see Interrupt Access Module), however without a return value and it‘s not depending on the system load. With the X-Realtime routine the same programming methods and restric tions are valid like on each other RING0 EXECUTION routine.

With the X-Realtime system several tasks can be programmed within an application or within a device driver and will be automatically mapped to the X-Realtime system layer at runtime. Every task can be setup with its own scheduling cycle which interacts independently to any other task cycles. Additionaly each task can given and changed its own priority dynamically. So several applications with their own realtime tasks can run at once. Together with application task also device drivers can setup their own realtime tasks to run within the X-Realtime system.

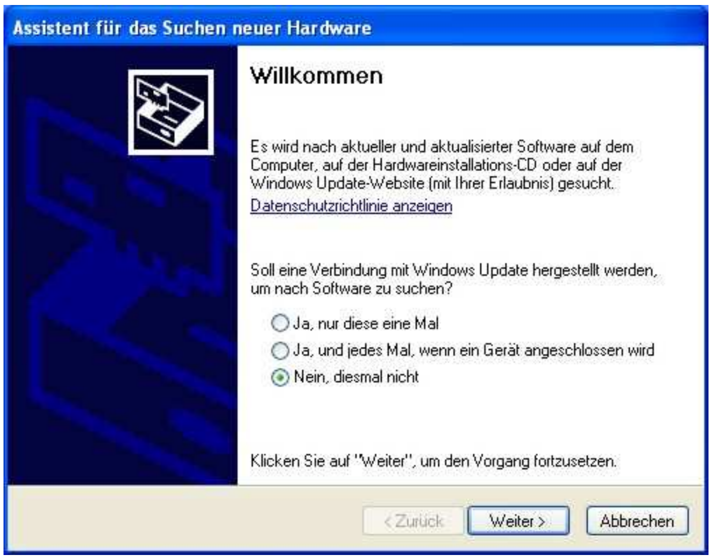
Note:
X-Realtime may not run with Firewall-Protection or Virus-Scan Software. Please disable / deinstall such protection programms.

2.2 X-Realtime Multitasking



2.3 CAN Core Port Driver Installation

The "CAN RealtimeCore" is based on SJA1000 CAN Adapters and will be installed as a PORT driver. After installing the PCI-Adapter, Windows asks for a PNP device driver:



Assistent für das Suchen neuer Hardware

Willkommen

Es wird nach aktueller und aktualisierter Software auf dem Computer, auf der Hardwareinstallations-CD oder auf der Windows Update-Website (mit Ihrer Erlaubnis) gesucht.

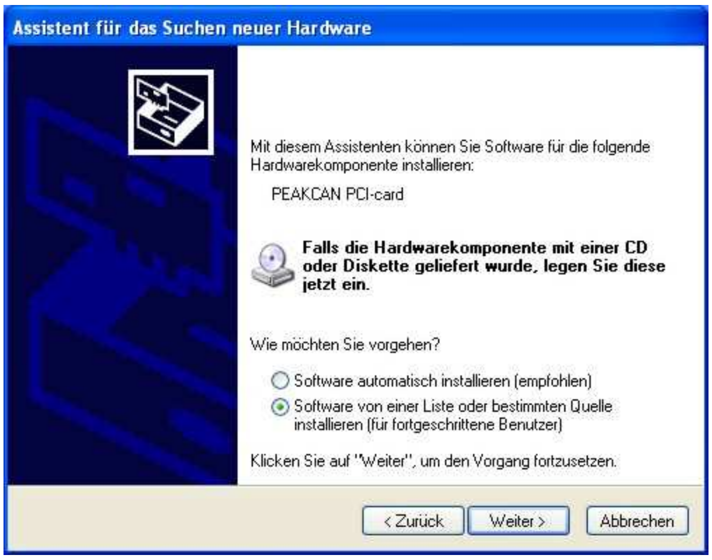
Datenschutzrichtlinie anzeigen

Soll eine Verbindung mit Windows Update hergestellt werden, um nach Software zu suchen?

- ○ Ja, nur diese eine Mal
- ○ Ja, und jedes Mal, wenn ein Gerät angeschlossen wird
- [✓ Nein, diesmal nicht]

Klicken Sie auf "Weiter", um den Vorgang fortzusetzen.

Choose to install the driver from a specified location:



Assistent für das Suchen neuer Hardware

Mit diesem Assistenten können Sie Software für die folgende Hardwarekomponente installieren:

PEAKCAN PCI-card

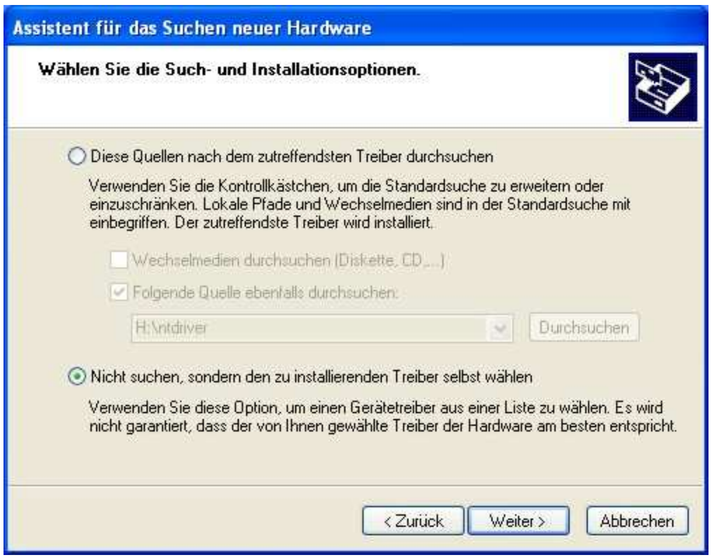
Falls die Hardwarekomponente mit einer CD oder Diskette geliefert wurde, legen Sie diese jetzt ein.

Wie möchten Sie vorgehen?

- ○ Software automatisch installieren (empfohlen)
- [✓ Software von einer Liste oder bestimmten Quelle installieren (für fortgeschrittene Benutzer)]

Klicken Sie auf "Weiter", um den Vorgang fortzusetzen.

Choose the drivers INF file from the CAN core WDM directory:


Assistent für das Suchen neuer Hardware

Wählen Sie die Such- und Installationsoptionen.

○ Diese Quellen nach dem zutreffendsten Treiber durchsuchen

Verwenden Sie die Kontrollkästchen, um die Standardsuche zu erweitern oder einzuschränken. Lokale Pfade und Wechselmedien sind in der Standardsuche mit einbegriffen. Der zutreffendste Treiber wird installiert.

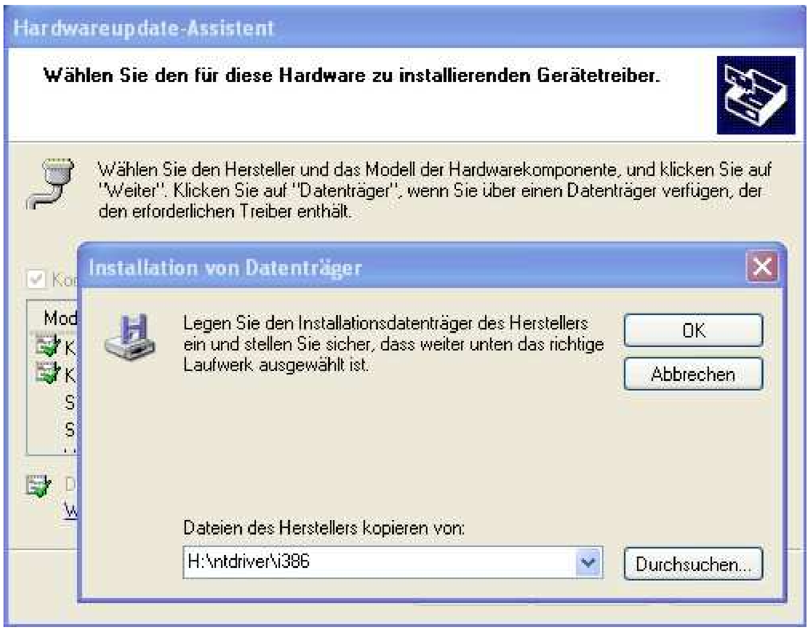
☐ Wechselmedien durchsuchen (Diskette, CD,...)

☑ Folgende Quelle ebenfalls durchsuchen:

H:\ntdriver

◉ Nicht suchen, sondern den zu installierenden Treiber selbst wählen

Verwenden Sie diese Option, um einen Gerätetreiber aus einer Liste zu wählen. Es wird nicht garantiert, dass der von Ihnen gewählte Treiber der Hardware am besten entspricht.


Hardwareupdate-Assistent

Wählen Sie den für diese Hardware zu installierenden Gerätetreiber.

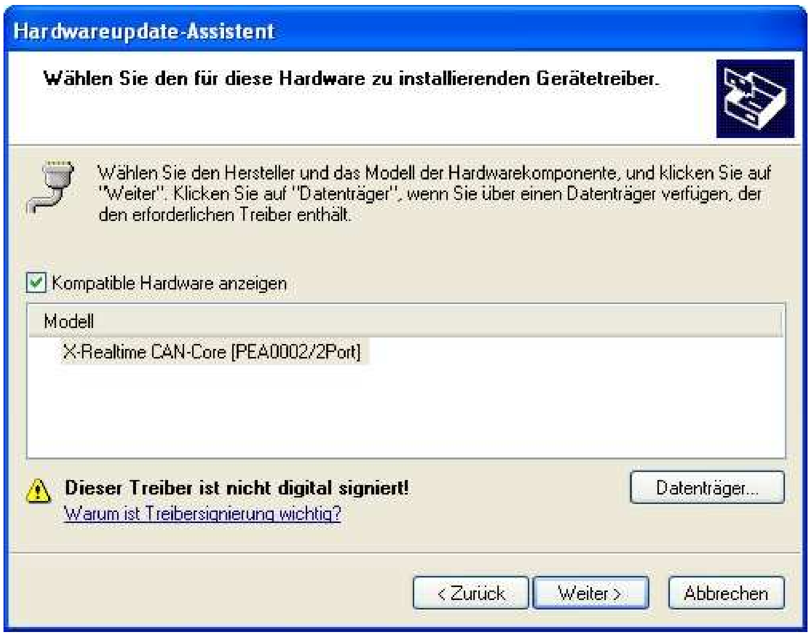
Wählen Sie den Hersteller und das Modell der Hardwarekomponente, und klicken Sie auf "Weiter". Klicken Sie auf "Datenträger", wenn Sie über einen Datenträger verfügen, der den erforderlichen Treiber enthält.

Installation von Datenträger

Legen Sie den Installationsdatenträger des Herstellers ein und stellen Sie sicher, dass weiter unten das richtige Laufwerk ausgewählt ist.

Dateien des Herstellers kopieren von:

H:\ntdriver\i386



Hardwareupdate-Assistent

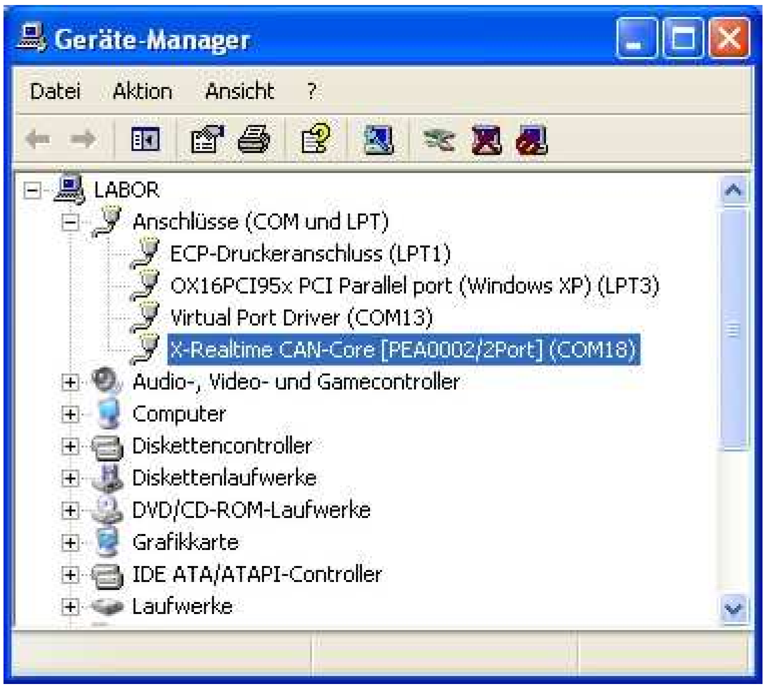
Wählen Sie den für diese Hardware zu installierenden Gerätetreiber.

Wählen Sie den Hersteller und das Modell der Hardwarekomponente, und klicken Sie auf "Weiter". Klicken Sie auf "Datenträger", wenn Sie über einen Datenträger verfügen, der den erforderlichen Treiber enthält.

☑ Kompatible Hardware anzeigen

Modell
X-Realtime CAN-Core [PEA0002/2Port]

Dieser Treiber ist nicht digital signiert!
Warum ist Treibersignierung wichtig?



Geräte-Manager

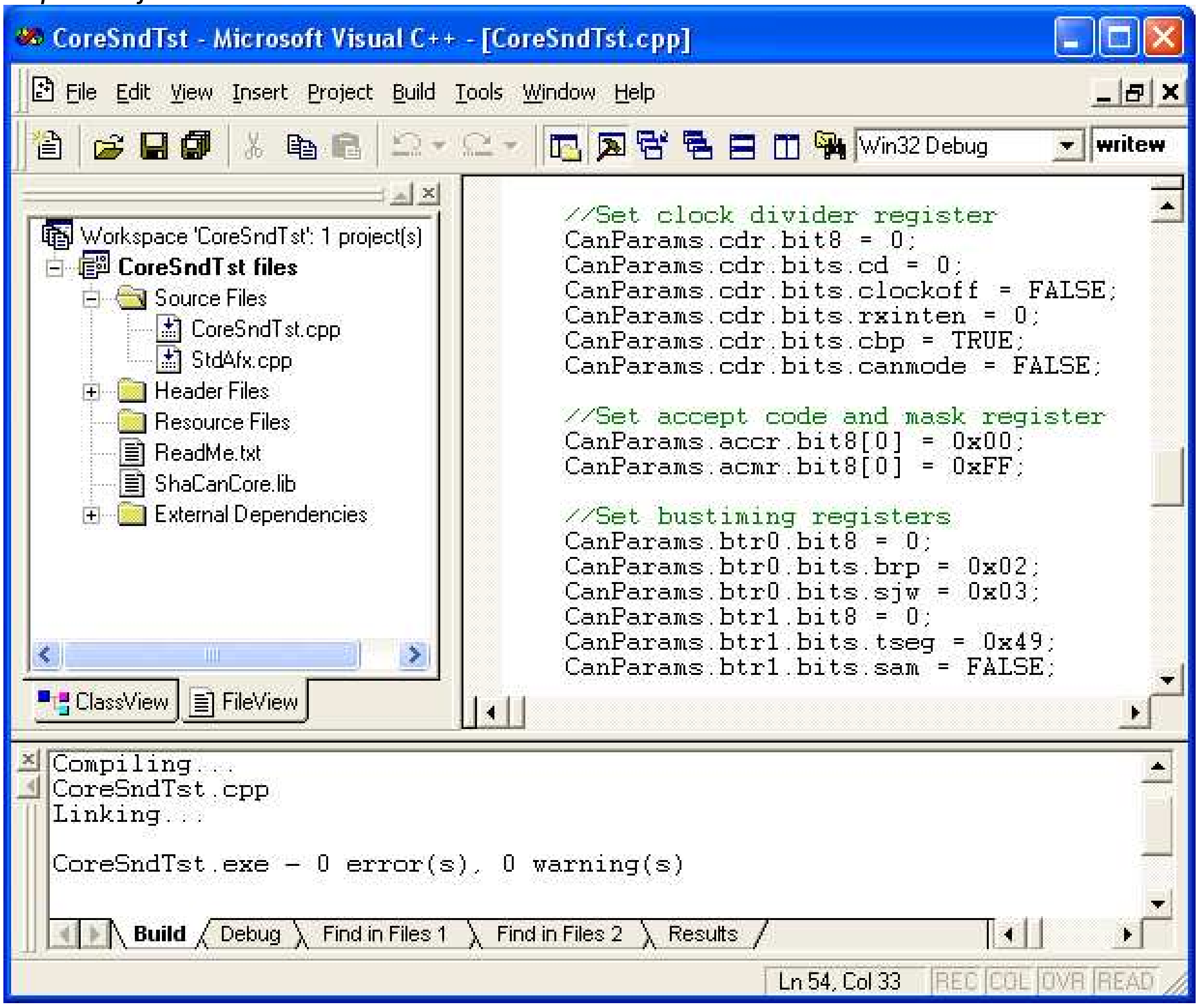
```
LABOR
  Anschlüsse (COM und LPT)
    ECP-Druckeranschluss (LPT1)
    OX16PCI95x PCI Parallel port (Windows XP) (LPT3)
    Virtual Port Driver (COM13)
    X-Realtime CAN-Core [PEA0002/2Port] (COM18)
  Audio-, Video- und Gamecontroller
  Computer
  Diskettencontroller
  Diskettenlaufwerke
  DVD/CD-ROM-Laufwerke
  Grafikkarte
  IDE ATA/ATAPI-Controller
  Laufwerke
```

Check, if the new driver was installed correctly. If the driver is not running, reboot the system and check again.

3 CAN RealtimeCore Library

The AddOn realtime library "CAN Realtime Core" allows sending and receiving of CAN frames, as well as the functional operation of CAN frames in realtime. The CAN Realtime Core library allows handling of serial data in realtime level 1 (ring buffered serial bytes) or realtime level 2 (functional handling within realtime task). The CAN realtime core allows making use of the standard COM interface of Windows, without changing the software, or programming via a simple proprietary interface.

Sample Project:

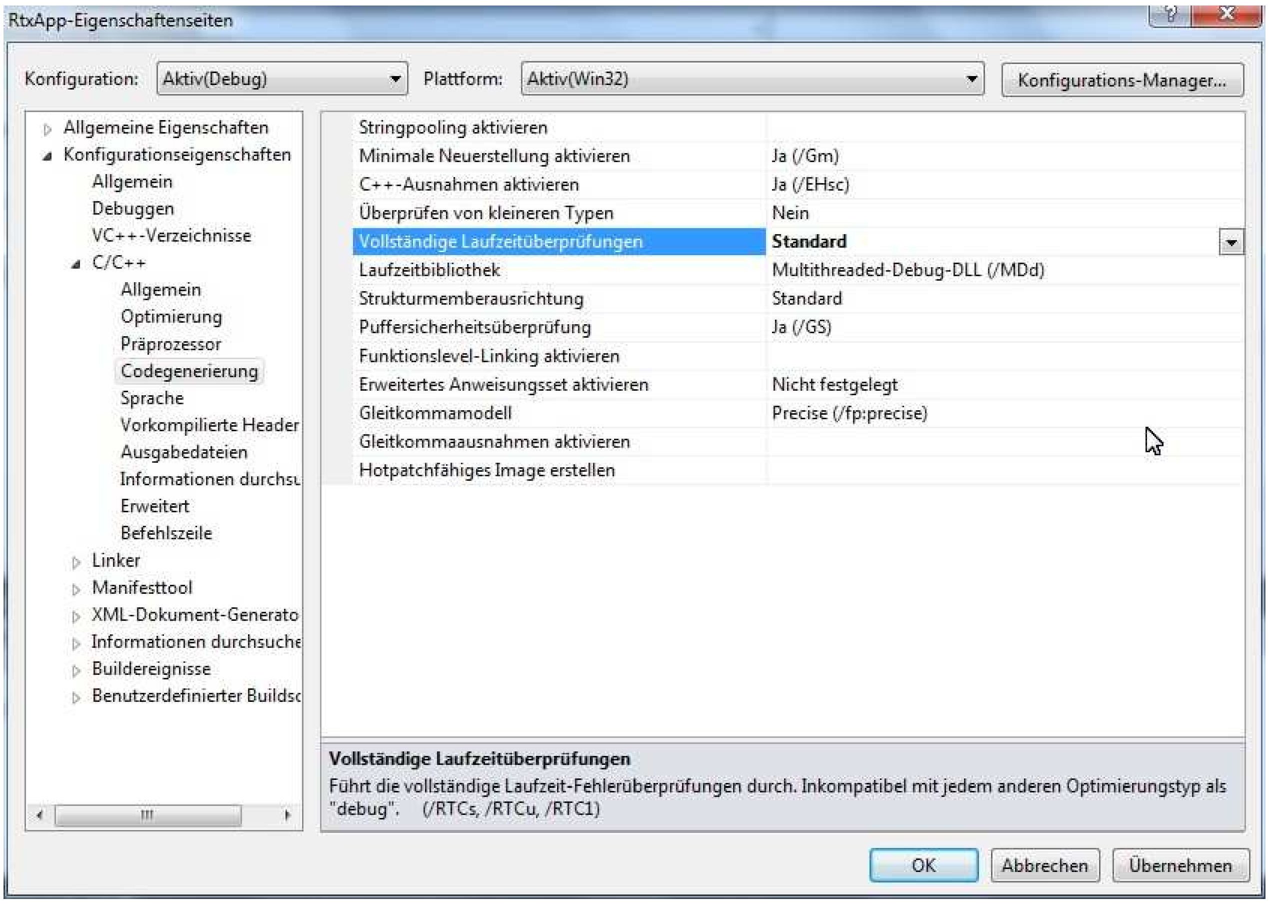


```
//Set clock divider register
CanParams.cdr.bit8 = 0;
CanParams.cdr.bits.cd = 0;
CanParams.cdr.bits.clockoff = FALSE;
CanParams.cdr.bits.rxinten = 0;
CanParams.cdr.bits.cbp = TRUE;
CanParams.cdr.bits.canmode = FALSE;

//Set accept code and mask register
CanParams.accr.bit8[0] = 0x00;
CanParams.acmr.bit8[0] = 0xFF;

//Set bustiming registers
CanParams.btr0.bit8 = 0;
CanParams.btr0.bits.brp = 0x02;
CanParams.btr0.bits.sjw = 0x03;
CanParams.btr1.bit8 = 0;
CanParams.btr1.bits.tseg = 0x49;
CanParams.btr1.bits.sam = FALSE;
```

```
Compiling...
CoreSndTst.cpp
Linking...

CoreSndTst.exe - 0 error(s), 0 warning(s)
```

3.1.1 Visual Studio 2010 Compiler Settings

With Visual Studio 2010 a change in the COMPILER settings was introduced. To make the Virtual Code Mapping (VCM) working correctly, the settings must be changed:


RtxApp-Eigenschaftenseiten

Konfiguration: Aktiv(Debug) Plattform: Aktiv(Win32) Konfigurations-Manager...

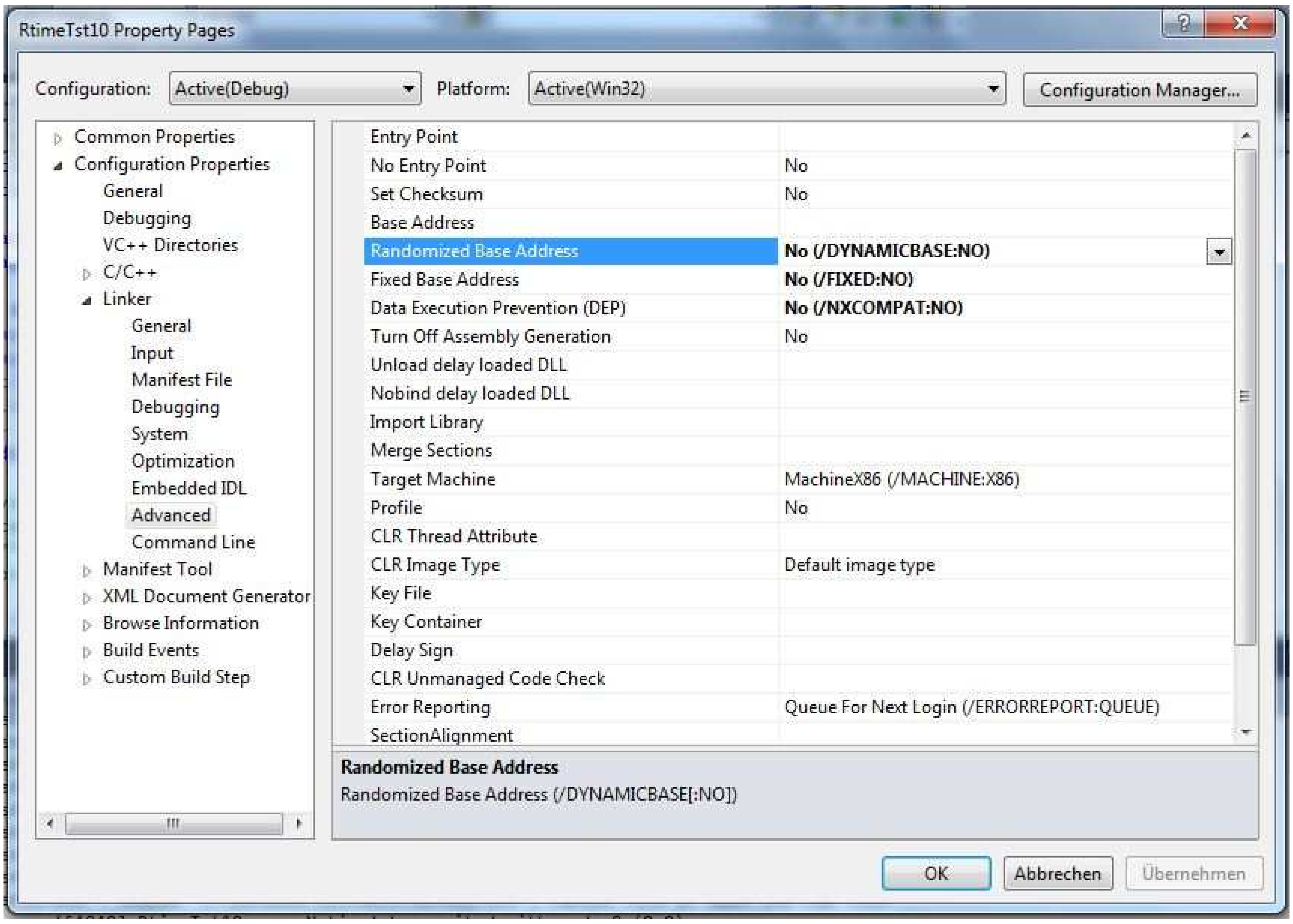
- Allgemeine Eigenschaften
- Konfigurationseigenschaften
  - Allgemein
  - Debuggen
  - VC++-Verzeichnisse
  - C/C++
    - Allgemein
    - Optimierung
    - Präprozessor
    - Codegenerierung
    - Sprache
    - Vorkompilierte Header
    - Ausgabedateien
    - Informationen durchsu
    - Erweitert
    - Befehlszeile
  - Linker
  - Manifesttool
  - XML-Dokument-Generato
  - Informationen durchsuche
  - Buildereignisse
  - Benutzerdefinierter Buildsc

Stringpooling aktivieren	
Minimale Neuerstellung aktivieren	Ja (/Gm)
C++-Ausnahmen aktivieren	Ja (/EHsc)
Überprüfen von kleineren Typen	Nein
Vollständige Laufzeitüberprüfungen	**Standard**
Laufzeitbibliothek	Multithreaded-Debug-DLL (/MDd)
Strukturmemberausrichtung	Standard
Puffersicherheitsüberprüfung	Ja (/GS)
Funktionslevel-Linking aktivieren	
Erweitertes Anweisungsset aktivieren	Nicht festgelegt
Gleitkommamodell	Precise (/fp:precise)
Gleitkommaausnahmen aktivieren	
Hotpatchfähiges Image erstellen	

Vollständige Laufzeitüberprüfungen
Führt die vollständige Laufzeit-Fehlerüberprüfungen durch. Inkompatibel mit jedem anderen Optimierungstyp als "debug". (/RTCs, /RTCu, /RTC1)

3.1.2 Visual Studio 2010 Linker Settings

With Visual Studio 2010 a change in the LINKER settings was introduced. To make the Virtual Code Mapping (VCM) working correctly, the settings must be changed:

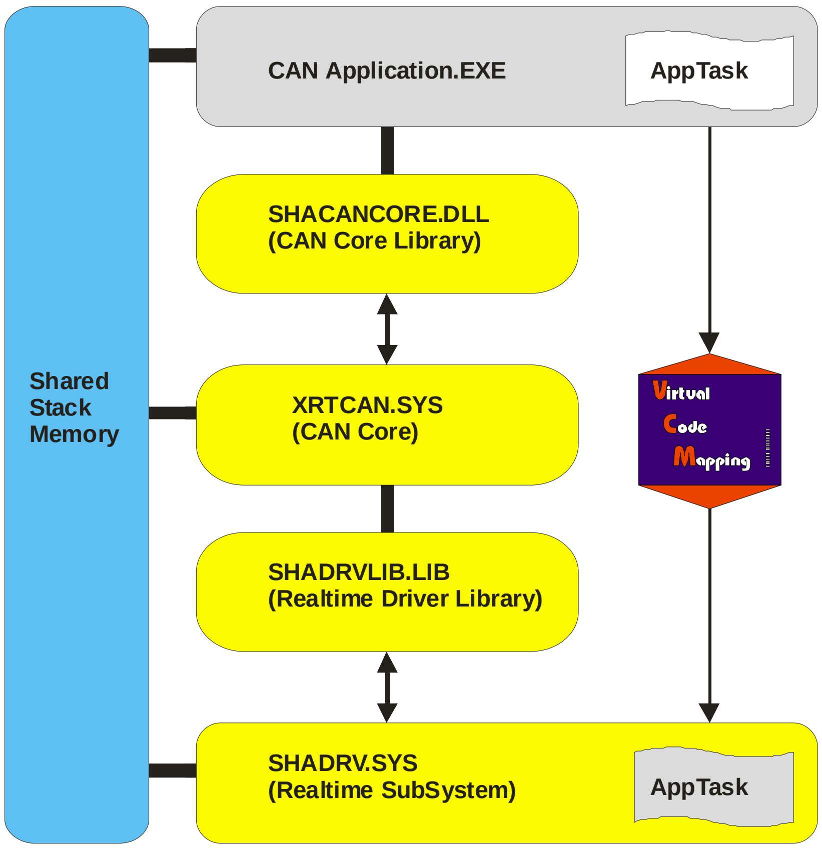

RtimeTst10 Property Pages

Configuration: Active(Debug) Platform: Active(Win32)

Entry Point	
No Entry Point	No
Set Checksum	No
Base Address	
Randomized Base Address	**No (/DYNAMICBASE:NO)**
Fixed Base Address	**No (/FIXED:NO)**
Data Execution Prevention (DEP)	**No (/NXCOMPAT:NO)**
Turn Off Assembly Generation	No
Unload delay loaded DLL	
Nobind delay loaded DLL	
Import Library	
Merge Sections	
Target Machine	MachineX86 (/MACHINE:X86)
Profile	No
CLR Thread Attribute	
CLR Image Type	Default image type
Key File	
Key Container	
Delay Sign	
CLR Unmanaged Code Check	
Error Reporting	Queue For Next Login (/ERRORREPORT:QUEUE)
SectionAlignment	

Randomized Base Address
Randomized Base Address (/DYNAMICBASE[:NO])

3.2 Realtime Driver Stack


CAN Application.EXE
AppTask
SHACANCORE.DLL (CAN Core Library)
Shared Stack Memory
XRTCAN.SYS (CAN Core)
Virtual Code Mapping
SHADRVLIB.LIB (Realtime Driver Library)
SHADRV.SYS (Realtime SubSystem)
AppTask

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++, eMbedded VC, Borland C++ Builder and LabWindows CVI

DP : Borland Delphi

3.2.1 VisualC++ and LabWindows CVI

Project files for VisualC++ and LabWindows:

`LIB\SHACANCORE.LIB`	Project Import Library
`LIB\SHACANCORE.DLL`	Dynamic Link Library (copied to \WINNT\SYSTEM32)
`INC\SHACANCORE.H`	Exported function prototypes
`INC\CANMACROS.H`	Exported CAN Macros
`INC\CANCOREDEF.H`	CAN core definitions
`INC\SJA1000DEF.H`	SJA1000 definitions

3.2.2 Borland C++Builder

Project files for Borland C++ Builder:

`LIB\SHACANCOREOML.LIB`	Project Import Library
`LIB\SHACANCOREOML.DLL`	Dynamic Link Library (copied to \WINNT\SYSTEM32)
`INC\SHACANCORE.H`	Exported function prototypes
`INC\CANMACROS.H`	Exported CAN Macros
`INC\CANCOREDEF.H`	CAN core definitions
`INC\SJA1000DEF.H`	SJA1000 definitions

3.2.3 Borland Delphi

Project files for Borland Delphi:

`LIB\SHACANCOREOML.LIB`	Project Import Library
`LIB\SHACANCOREOML.DLL`	Dynamic Link Library (copied to \WINNT\SYSTEM32)
`INC\SHACANCORE.PAS`	Exported function prototypes
`INC\CANMACROS.PAS`	Exported CAN functions
`INC\CANCOREDEF.PAS`	CAN core definitions
`INC\SJA1000DEF.PAS`	SJA1000 definitions

3.3 Header File CANCOREDEF.H

This header file defines all structures required for handling the core interface and realtime stack data. The structure elements are combined hierachically.

3.3.1 Structure CAN_PARAMS

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

```
typedef struct _CAN_PARAMS
{
      //Input parameters
      PCHAR            port_name;        //Port name (e.g. CAN1)
      ULONG            period;           //Realtime scheduling period
      ULONG            sched_cnt;        //Application scheduling count
      ULONG            timeout;          //Timeout constant in msec
      UCHAR            baud_rate;        //SJA CAN baudrate (optional)
      UCHAR            mode;             //SJA CAN mode (BASECAN, PELICAN)

      //Output parameters
      ULONG            port_pa;          //Base Port Address
      ULONG            port_id;          //Port identifier
      ULONG            memory_tag;       //Memory TAG
      ULONG            remain_time;      //Remaining realtime
      ULONG            core_dll_ver;     //Core DLL version
      ULONG            core_drv_ver;     //Core driver version
      ULONG            sha_drv_ver;      //SHA driver version
      ULONG            sha_lib_ver;      //SHA library version
      CAN_ERR_CNTS     err_cnts;         //CAN error counters

      //Input - Output parameters
      CAN_FRAME        frame;            //CAN frame

      //SJA parameters (optional)
      SJA_PARAMS       sja_params;       //SJA register parameters

      //Realtime level2 input parameters
      FP_RING0         fpAppTask;        //Function pointer to realtime
                                         //application task
      //Realtime level2 output parameters
      PCAN_STACK       pSystemStack;     //CAN_STACK structure for realtime
                                         //application task
      PCAN_STACK       pUserStack;       //CAN_STACK structure for windows
                                         //application task
} CAN_PARAMS, *PCAN_PARAMS;
```

3.3.2 Structure CAN_STACK

This structure is required when accessing the ethernet data of the Realtime Core directly (Realtime Level2). The structure elements are combined hierachically:

```
typedef struct _CAN_STACK
{
      CAN_STACK_HDR     hdr;
      CAN_TABLE         tx_table;
      CAN_TABLE         rx_table;

} CAN_STACK, *PCAN_STACK;


      typedef struct _CAN_STACK_HDR
      {
            BOOLEAN           run_flag;
            BOOLEAN           err_flag;
            BOOLEAN           lev2_flag;

      } CAN_STACK_HDR, *PCAN_STACK_HDR;


      typedef struct _CAN_TABLE
      {
            ULONG       findex;           //Forward index
            ULONG       bindex;           //Back Index
            CAN_ENTRY   list[MAX_ENTRIES];

      } CAN_TABLE, *PCAN_TABLE;


            typedef struct _CAN_ENTRY
            {
                  CAN_FRAME   frame;            //CAN frame
                  BOOLEAN     bOccupied;        //table flag
                  __int64     TscCnt;           //Time Scale Counter

            } CAN_ENTRY, *PCAN_ENTRY;
```

3.4 Header File CANMACROS.H

This header file defines all macros required for handling realtime level 2.

//Macro for coding SJA1000 BaseCAN frame
```
#define CAN_SET_FRAME10(__pFrame, __Addr, __pData, __len)  \
{                                                          \
      ULONG _i = 0;                                        \
      PUCHAR _pData     = (PUCHAR)__pData;                 \
      PCAN_FRAME  _pFrame = (PCAN_FRAME)__pFrame;          \
      _pFrame->bc.id.dlc = (UCHAR)__len;                   \
      _pFrame->bc.id.rtr = FALSE;                          \
      _pFrame->bc.id.id0_2  = (USHORT)(__Addr & 0x07);     \
      _pFrame->bc.id.id3_10 = (USHORT)(__Addr >> 3) & 0xFF;     \
      for (_i=0; _i<__len; _i++)                           \
            _pFrame->bc.data[_i] = _pData[_i];             \
}                                                          \
```

//Macro for decoding SJA1000 BaseCAN frame
```
#define CAN_GET_FRAME10(__pFrame, __pAddr, __pData, __pLen)      \
{                                                                \
      ULONG _i = 0;                                              \
      PUSHORT _pAddr    = (PUSHORT)__pAddr;                      \
      PUCHAR  _pData    = (PUCHAR)__pData;                       \
      PULONG  _pLen     = (PULONG)__pLen;                        \
      PCAN_FRAME _pFrame = (PCAN_FRAME)__pFrame;                 \
      if (_pAddr)                                                \
      {                                                          \
            *_pAddr  = _pFrame->bc.id.id0_2;                     \
            *_pAddr += _pFrame->bc.id.id3_10 << 3;               \
      }                                                          \
      if (_pLen)  *_pLen = _pFrame->bc.id.dlc;                   \
      if (_pData)                                                \
            for (_i=0; _i<(_pFrame->bc.id.dlc); _i++)            \
                  _pData[_i] = _pFrame->bc.data[_i];             \
}                                                                \
```

//Macro for coding SJA1000 SFF frame
```
#define CAN_SET_FRAME11(__pFrame, __Addr, __pData, __len)        \
{                                                                \
      ULONG _i = 0;                                              \
      PUCHAR _pData = (PUCHAR)__pData;                           \
      PCAN_FRAME _pFrame = (PCAN_FRAME)__pFrame;                 \
      _pFrame->pc.fi.dlc = (UCHAR)__len;                         \
      _pFrame->pc.fi.rtr = FALSE;                                \
      _pFrame->pc.fi.ff  = FF_SFF;                               \
      _pFrame->pc.ff.sff.id.id18_20 = (USHORT)(__Addr & 0x07);         \
      _pFrame->pc.ff.sff.id.id21_28 = (USHORT)(__Addr >> 3) & 0xFF;    \
      for (_i=0; _i<__len; _i++)                                       \
            _pFrame->pc.ff.sff.data[_i] = _pData[_i];                  \
}                                                                      \
```

//Macro for decoding SJA1000 SFF frame
```
#define CAN_GET_FRAME11(__pFrame, __pAddr, __pData, __pLen)      \
{                                                                \
      ULONG _i = 0;                                              \
      PUSHORT _pAddr    = (PUSHORT)__pAddr;                      \
      PUCHAR  _pData    = (PUCHAR)__pData;                       \
      PULONG  _pLen     = (PULONG)__pLen;                        \
      PCAN_FRAME _pFrame = (PCAN_FRAME)__pFrame;                 \
      if (_pFrame->pc.fi.ff == FF_SFF)                           \
      {                                                          \
            if (_pAddr)                                          \
            {                                                    \
                  *_pAddr  = _pFrame->pc.ff.sff.id.id18_20;      \
                  *_pAddr += _pFrame->pc.ff.sff.id.id21_28 << 3; \
            }                                                    \
            if (_pLen)  *_pLen = _pFrame->pc.fi.dlc;             \
            if (_pData)                                          \
                  for (_i=0; _i<(_pFrame->pc.fi.dlc); _i++)      \
                        _pData[_i] = _pFrame->pc.ff.sff.data[_i];\
      }                                                          \
}                                                                \
```

//Macro for coding SJA1000 EFF frame
```
#define CAN_SET_FRAME13(__pFrame, __Addr, __pData, __len)  \
{                                                          \
      ULONG       _i = 0;                                  \
      PUCHAR      _pData = (PUCHAR)__pData;                \
      PCAN_FRAME  _pFrame = (PCAN_FRAME)__pFrame;          \
      _pFrame->pc.fi.dlc = (UCHAR)__len;                   \
      _pFrame->pc.fi.rtr = FALSE;                          \
      _pFrame->pc.fi.ff  = FF_EFF;                         \
      _pFrame->pc.ff.eff.id.id0_4   = (ULONG)(__Addr & 0x1F);        \
      _pFrame->pc.ff.eff.id.id5_12 = (ULONG)(__Addr >> 5)  & 0xFF;   \
      _pFrame->pc.ff.eff.id.id13_20= (ULONG)(__Addr >> 13) & 0xFF;   \
      _pFrame->pc.ff.eff.id.id21_28= (ULONG)(__Addr >> 21) & 0xFF;   \
      for (_i=0; _i<__len; _i++)                                     \
            _pFrame->pc.ff.sff.data[_i] = _pData[_i];                \
}                                                                    \
```

//Macro for decoding SJA1000 SFF frame
```
#define CAN_GET_FRAME13(__pFrame, __pAddr, __pData, __pLen)      \
{                                                                \
      ULONG _i = 0;                                              \
      PUSHORT _pAddr    = (PUSHORT)__pAddr;                      \
      PUCHAR  _pData    = (PUCHAR)__pData;                       \
      PULONG  _pLen     = (PULONG)__pLen;                        \
      PCAN_FRAME _pFrame = (PCAN_FRAME)__pFrame;                 \
      if (_pFrame->pc.fi.ff == FF_EFF)                           \
      {                                                          \
            if (_pAddr)                                          \
            {                                                    \
                  *_pAddr  = _pFrame->pc.ff.eff.id.id0_4;              \
                  *_pAddr += _pFrame->pc.ff.eff.id.id5_12  << 5;       \
                  *_pAddr += _pFrame->pc.ff.eff.id.id13_20 << 13;      \
                  *_pAddr += _pFrame->pc.ff.eff.id.id21_28 << 21;      \
            }                                                          \
            if (_pLen)  *_pLen = _pFrame->pc.fi.dlc;                   \
            if (_pData)                                                \
                  for (_i=0; _i<(_pFrame->pc.fi.dlc); _i++)            \
                        _pData[_i] = _pFrame->pc.ff.sff.data[_i];      \
      }                                                                \
}                                                                      \
```

//Macro to get frame size
```
#define CAN_GET_FRAMESIZE(__pFrame, __pSize, __Mode)             \
{                                                                \
      PULONG      _pSize      = (PULONG)__pSize;                 \
      PCAN_FRAME  _pFrame = (PCAN_FRAME)__pFrame;                \
      if (__Mode == BASECAN)  { *_pSize =  _pFrame->bc.id.dlc + 2; }   \
      else { *_pSize = (_pFrame->pc.fi.ff == FF_SFF) ?                 \
            (_pFrame->pc.fi.dlc + 3) : (_pFrame->pc.fi.dlc + 5); }     \
}                                                                      \
```

//Macro to start or stop realtime tasks
```
#define CAN_TASK_CONTROL(__pStack, __bRun)                       \
{                                                                \
      PCAN_STACK        _pStack = (PCAN_STACK)__pStack;          \
      PCAN_STACK_HDR    _pHdr = (PCAN_STACK_HDR)&_pStack->hdr;   \
      _pHdr->err_flag = FALSE;                                   \
      _pHdr->run_flag = __bRun;                                  \
}                                                                \
```

//Macro to enable or disable realtime level2 and save condition
```
#define CAN_LEVEL2_CONTROL(__pStack, __bCond)                    \
{                                                                \
      PCAN_STACK        _pStack = (PCAN_STACK)__pStack;          \
      PCAN_STACK_HDR    _pHdr = (PCAN_STACK_HDR)&_pStack->hdr;   \
      _pHdr->lev2_flag = __bCond;                                \
}                                                                \
```

//Macro to control TX and RX stack index
```
#define CAN_STACK_CONTROL(__pStack, __TxIndex, __RxIndex)       \
{                                                               \
      PCAN_STACK        _pStack = (PCAN_STACK)__pStack;                    \
      PCAN_STACK_HDR    _pHdr = (PCAN_STACK_HDR)&_pStack->hdr;             \
      PCAN_TABLE        _pTxTable = (PCAN_TABLE)&_pStack->tx_table;        \
      PCAN_TABLE        _pRxTable = (PCAN_TABLE)&_pStack->rx_table;        \
      PCAN_ENTRY        _pTxEntry = (PCAN_ENTRY)&_pTxTable->list[__TxIndex]; \
      PCAN_ENTRY        _pRxEntry = (PCAN_ENTRY)&_pRxTable->list[__RxIndex]; \
      BOOLEAN           _bRun = _pHdr->run_flag;                           \
      _pHdr->run_flag = FALSE;                                             \
      _pTxEntry->bOccupied = FALSE;                                        \
      _pRxEntry->bOccupied = FALSE;                                        \
      _pTxTable->bindex = __TxIndex;                                       \
      _pRxTable->bindex = __RxIndex;                                       \
      _pTxTable->findex = __TxIndex;                                       \
      _pRxTable->findex = __RxIndex;                                       \
      _pHdr->run_flag = _bRun;                                             \
}                                                                          \
```

//Macro to check for idle stack
```
#define CAN_CHECK_STACK_IDLE(__pStack, __pbIdle)                     \
{                                                                    \
      PCAN_STACK  _pStack = (PCAN_STACK)__pStack;                    \
      PCAN_TABLE  _pTxTable = (PCAN_TABLE)&_pStack->tx_table;        \
      PCAN_TABLE  _pRxTable = (PCAN_TABLE)&_pStack->rx_table;        \
      BOOLEAN           _bRun = _pHdr->run_flag;                     \
      *__pbIdle = FALSE;                                             \
      if ((_pTxTable->findex == _pTxTable->bindex) &&                \
            (_pRxTable->findex == _pRxTable->bindex))                \
            *__pbIdle = TRUE;                                        \
}                                                                    \
```

//Macro to get current TX entry pointer
```
#define CAN_GET_TXENTRY_PTR(__pStack, __ppEntry)                     \
{                                                                    \
      PCAN_STACK  _pStack = (PCAN_STACK)__pStack;                    \
      PCAN_TABLE  _pTable = (PCAN_TABLE)&_pStack->tx_table;          \
      ULONG       _Index = (ULONG)_pTable->bindex;                   \
      PCAN_ENTRY  _pEntry = (PCAN_ENTRY)&_pTable->list[_Index];      \
      *__ppEntry = _pEntry;                                          \
}                                                                    \
```

//Macro to get current RX entry pointer
```
#define CAN_GET_RXENTRY_PTR(__pStack, __ppEntry)                     \
{                                                                    \
      PCAN_STACK  _pStack = (PCAN_STACK)__pStack;                    \
      PCAN_TABLE  _pTable = (PCAN_TABLE)&_pStack->rx_table;          \
      ULONG       _Index      = (ULONG)_pTable->findex;              \
      PCAN_ENTRY  _pEntry = (PCAN_ENTRY)&_pTable->list[_Index];      \
      *__ppEntry = _pEntry;                                          \
}                                                                    \
```

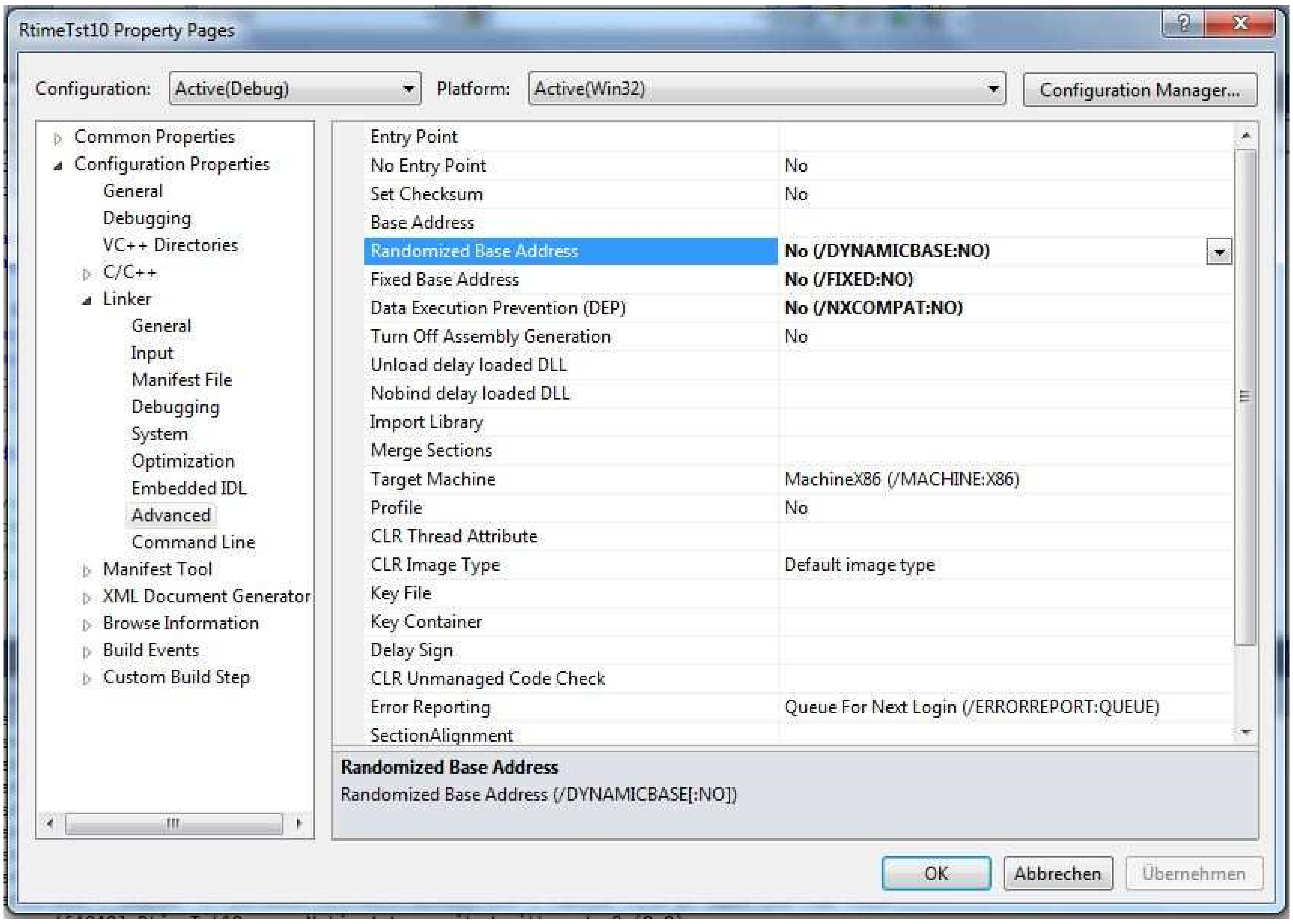
//Macro to check error flag
```
#define CAN_CHECK_ERROR(__pStack, __pbError)                             \
{                                                                        \
      PCAN_STACK        _pStack = (PCAN_STACK)__pStack;                  \
      PCAN_STACK_HDR    _pHdr = (PCAN_STACK_HDR)&_pStack->hdr;           \
      *__pbError = _pHdr->err_flag;                                      \
}                                                                        \
```

//Macro to set error condition
```
#define CAN_SET_ERROR(__pStack)                                          \
{                                                                        \
      PCAN_STACK        _pStack = (PCAN_STACK)__pStack;                  \
      PCAN_STACK_HDR    _pHdr = (PCAN_STACK_HDR)&_pStack->hdr;           \
      _pHdr->err_flag = TRUE;                                            \
      _pHdr->run_flag = FALSE;                                           \
}                                                                        \
```

3.5 CAN Core Interface

The header file SHACANCORE.H defines all required prototypes and parameters of the CAN Core Library. In the following all function prototypes will be discussed by samples.

Note:
With Visual Studio 2010 a change in the LINKER settings was introduced. To make the Virtual Code Mapping (VCM) working correctly, the settings must be changed:



Entry Point	
No Entry Point	No
Set Checksum	No
Base Address	
Randomized Base Address	**No (/DYNAMICBASE:NO)**
Fixed Base Address	**No (/FIXED:NO)**
Data Execution Prevention (DEP)	**No (/NXCOMPAT:NO)**
Turn Off Assembly Generation	No
Unload delay loaded DLL	
Nobind delay loaded DLL	
Import Library	
Merge Sections	
Target Machine	MachineX86 (/MACHINE:X86)
Profile	No
CLR Thread Attribute	
CLR Image Type	Default image type
Key File	
Key Container	
Delay Sign	
CLR Unmanaged Code Check	
Error Reporting	Queue For Next Login (/ERRORREPORT:QUEUE)
SectionAlignment	

Randomized Base Address
Randomized Base Address (/DYNAMICBASE[:NO])

3.5.1 ShaCanCreate

This function opens the communication port and installs the realtime ringbuffer system. On success the returning value is ERROR_SUCCESS, otherwise the returning value corresponds to that from GetLastError(). The usable interface types are defined in the header file "GLOBDEF.H".

VC `ULONG ShaCanCreate(CAN_PARAMS);`

Note:
Since CAN ports are managed as COM ports, COM1 through COM9 can be directly referenced just like a filename from programs and from the command line. However, COM10 and above must be referenced with the following syntax:

“\\\\.\COM10”

Sample:

```
memset(&Params, 0, sizeof(CAN_PARAMS));
Params.port_name = "\\\\.\\COM18"; //Port name
Params.period = 100;                     //Realtime scheduling period
Params.sched_cnt = 1;                    //Realtime scheduling count
Params.baud_rate = CAN_BAUD_500K;        //CAN baudrate
Params.mode = PELICAN;                   //CAN mode
Params.timeout = 3000;                   //Read timeout constant in msec
Params.fpAppTask = NULL;                 //No realtime level2 task

//Init CAN
if (ERROR_SUCCESS == ShaCanCreate(&Params))
{ …
}
```

3.5.2 ShaCanDestroy

This function closes the communication port and releases all resources for communication.

```
VC      void ShaCanDestroy(PCAN_PARAMS pCanParams);
```

3.5.3 ShaCanReset

This function resets the communication port and releases all resources for communication.

```
VC      BOOLEAN ShaCanReset(PCAN_PARAMS pCanParams);
```

3.5.4 ShaCanCheckStatus

This function gets the status of the CAN comminication.

```
VC      void ShaCanCheckStatus(PCAN_PARAMS pCanParams);
```

3.5.5 ShaCanTransmitFrame

This function read data from the Realtime RingBuffer. The function requires to allocate a buffer for the sending data.

```
VC      void ShaCanTransmitFrame(PCAN_PARAMS pCanParams);
```

3.5.6 ShaCanReceiceFrame

This function read data from the Realtime RingBuffer. The function requires to allocate a buffer for the receive data.

```
VC      void ShaCanReceiveFrame(PCAN_PARAMS pCanParams);
```

3.6 Sample program (Realtime Level 2):

```
//*****************************************************************
//*** Sample with Beckhoff CAN Modules BK5150, KL1104, KL2404   ***
//*****************************************************************

#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include "common.h"
#include "c:\can\ShaCanCore.h"
#include "c:\sha\shaexp.h"

//*****************************************************************
#if defined(_MSC_VER) && !defined(__BORLANDC__)
    //This is required for VisualC++
#else
    //This is required for C++Builder
    #pragma hdrstop
    #pragma option -w-bbf   //switch off bit field warning
    #pragma option -w-asc   //switch off the assembler warning
    #pragma option -Od      //disable all optimization (for Delphi highspeed
                            // access)
    #include <condefs.h>

//Get externals
USELIB("C:\CAN\shacancoreoml.lib");
USELIB("C:\SHA\shadlloml.lib");
//---------------------------------------------------------------------------
#endif
//*****************************************************************

#define CAN_OP          "\x01\x00\x00\x00\x00\x00\x00\x00"
#define CAN_STOP        "\x02\x00\x00\x00\x00\x00\x00\x00"
#define CAN_PRE_OP      "\x80\x00\x00\x00\x00\x00\x00\x00"
#define CAN_INIT        "\x81\x00\x00\x00\x00\x00\x00\x00"
#define CAN_RESET       "\x82\x00\x00\x00\x00\x00\x00\x00"

//Define can addresses
#define FIXED_CAN_ADDR        0x05
#define DI_CAN_ADDR           0x180 + FIXED_CAN_ADDR
#define DO_CAN_ADDR           0x200 + FIXED_CAN_ADDR
#define AI_CAN_ADDR           0x380 + FIXED_CAN_ADDR
#define AO_CAN_ADDR           0x400 + FIXED_CAN_ADDR


enum _CAN_UPDATE_TYPE
{
      KL_DO_UPDATE = 1,
      KL_DI_UPDATE,
      KL_AO_UPDATE,
      KL_AI_UPDATE
};

//Some globals
```

```
HANDLE hTermEvent = NULL;
HANDLE hCanEvent = NULL;

//*****************************************************************
//*** Realtime section ********************************************
//*****************************************************************

//Common elements
PCAN_STACK  pSystemStack = NULL;
PCAN_STACK  pUserStack = NULL;
PPLC_MEM    pUserPlc = NULL;
PPLC_MEM    pSystemPlc = NULL;
UCHAR       CanUpdateType = 0;
USHORT      CanAddr = 0;
UCHAR       TestData[8] = { 0,0,0,0, 0,0,0,0 };
BOOLEAN     bToggle = FALSE;


void static RxFunc(void)
{
      //Get RX frame elements from CAN stack structure
      PCAN_FRAME pRxFrame = &pSystemStack->rx_table.list[0].frame;
      PBOOLEAN pbRxOccupied = &pSystemStack->rx_table.list[0].bOccupied;

      //Get receive frame if table is occupied
      if (*pbRxOccupied == TRUE)
      {
            //Get can address
            CAN_GET_FRAME11(pRxFrame, &CanAddr, NULL, NULL);

            //Check digital inputs
            if (CanAddr == DI_CAN_ADDR)
            {
            CAN_GET_FRAME11(pRxFrame,NULL,pSystemPlc->CanFrameKlDi.bit8,NULL);
            CanUpdateType = KL_DI_UPDATE;
            }

            //Check analog inputs
            if (CanAddr == AI_CAN_ADDR)
            {
            CAN_GET_FRAME11(pRxFrame,NULL,pSystemPlc->CanFrameKlAi.bit8,NULL);
            CanUpdateType = KL_AI_UPDATE;
            }

            //Set occupied flag
            *pbRxOccupied = FALSE;
      }

      //Reset update type
      CanUpdateType = 0;
}


void static LogicalFunc(void)
{
      //Sample: Set output A1 due input A1
      if (pSystemPlc->CanFrameKlDi.bits.kl1104_2a.a1)
            pSystemPlc->CanFrameKlDo.bits.kl2404_2a.a1 = TRUE;
      else
            pSystemPlc->CanFrameKlDo.bits.kl2404_2a.a1 = FALSE;

      //Sample: Toggle output B1
      if (bToggle)      { bToggle = FALSE; }
      else              { bToggle = TRUE; }
      pSystemPlc->CanFrameKlDo.bits.kl2404_2a.b1 = bToggle;

      //Set update type
      CanUpdateType = KL_DO_UPDATE;
}


void static TxFunc(void)
{
      //Get RX frame elements from CAN stack structure
      PCAN_FRAME pTxFrame = &pSystemStack->tx_table.list[0].frame;
      PBOOLEAN pbTxOccupied = &pSystemStack->tx_table.list[0].bOccupied;

      //Get receive frame if table is occupied
      if (*pbTxOccupied == FALSE)
      {
            if (CanUpdateType == KL_DO_UPDATE)
            {
            //Set stack frame
            CAN_SET_FRAME11(pTxFrame,DO_CAN_ADDR,
                              pSystemPlc->CanFrameKlDo.bit8,8);
            }

            if (CanUpdateType == KL_AO_UPDATE)
            {
                  //Set stack frame
                  CAN_SET_FRAME11(pTxFrame, AO_CAN_ADDR,
                                    pSystemPlc->CanFrameKlAo.bit8,8);
            }

            //Reset update type
            CanUpdateType = 0;

            //Set occupied flag
            *pbTxOccupied = TRUE;
      }
}


void static AppTask(void)
{
//    DBG_INITIAL_BREAK();

      //Check if CAN stack is valid
      if (!pSystemStack)
            return;

      RxFunc();
      LogicalFunc();
      TxFunc();
}
```

```
//*****************************************************************
//*****************************************************************
//*****************************************************************

BOOLEAN CanInit(void)
{
      CAN_PARAMS Params;

      //*******************************
      //*** Required CAN parameters ***
      //*******************************
      memset(&Params, 0, sizeof(CAN_PARAMS));
      Params.port_name = "\\\\.\\COM18"; //Port name
      Params.period = 100;                //Realtime scheduling period
      Params.sched_cnt = 1;               //Realtime scheduling count
      Params.baud_rate = CAN_BAUD_500K;   //CAN baudrate
      Params.mode = PELICAN;              //CAN mode
      Params.timeout = 3000;              //Read timeout constant in msec
      Params.fpAppTask = NULL;            //No realtime level2 task

      //Init CAN
      if (ERROR_SUCCESS == ShaCanCreate(&Params))
      {
            //Reset level2 condition
            //Enable task scheduler
            CAN_LEVEL2_CONTROL(Params.pUserStack, FALSE);
            CAN_TASK_CONTROL(Params.pUserStack, TRUE);

            //Print SHA driver, DLL and CORE versions
            ShaCanGetVersion(&Params);
            printf("CORE-DLL Version: %.2f\nCORE-DRV Version: %.2f\n
                  SHA-LIB Version: %.2f\nSHA-DRV Version: %.2f\n",
                  Params.core_dll_ver / (double)100,
                  Params.core_drv_ver / (double)100,
                  Params.sha_lib_ver / (double)100,
                  Params.sha_drv_ver / (double)100);

            //Set and send init frame
            CAN_SET_FRAME11(&Params.frame, 0, CAN_OP, 8);
            ShaCanTransmitFrame(&Params);

            //Cleanup CAN
            ShaCanDestroy(&Params);
            return TRUE;
      }

      //Something failed
      return FALSE;
}


DWORD WINAPI CanThread(LPVOID)
{
      CAN_PARAMS Params;
      UCHAR c = '\\';

      //*******************************
      //*** Required CAN parameters ***
      //******************************
      memset(&Params, 0, sizeof(CAN_PARAMS));
      Params.port_name = "\\\\.\\COM18"; //Port name
      Params.period = 100;               //Realtime scheduling period
      Params.sched_cnt = 1000;           //Realtime scheduling count
      Params.baud_rate = CAN_BAUD_500K;  //CAN baudrate
      Params.mode = PELICAN;             //CAN mode
      Params.timeout = 3000;             //Read timeout constant in msec
      Params.fpAppTask = AppTask;        //Realtime level2 task

      //Init CAN
      if (ERROR_SUCCESS == ShaCanCreate(&Params))
      {
            //Init realtime elements
            pSystemStack = Params.pSystemStack;
            pUserStack   = Params.pUserStack;

            //Set level2 condition
            //Set TX and RX stack location to 0
            //Enable task scheduler
            CAN_LEVEL2_CONTROL(pUserStack, TRUE);
            CAN_STACK_CONTROL(pUserStack, 0, 0);
            CAN_TASK_CONTROL(pUserStack, TRUE);

            //Check termination event
            while (WAIT_OBJECT_0 != WaitForSingleObject(hTermEvent, 0))
            {
                  //Check for errors and get remaining realtime
                  ShaCanCheckStatus(&Params);
                  if (Params.sja_params.sr.bits.es)
                  {
                  printf("SJA SR:   [%02x]\n", Params.sja_params.sr.bit8);
                  printf("SJA ECCR: [%02x]\n", Params.sja_params.eccr.bit8);
                  printf("SJA ALCR: [%02x]\n", Params.sja_params.alcr.bit8);

                  //Reset the transmission
                  ShaCanResetTransmission(&Params);
                  }

                  //Print active char
                  c = (c == '\\') ? '/' : '\\';
                  printf("Realtime Period: %i, Remaining Time: %i [%c]\r",
                              Params.period, Params.remain_time, c);
                  Sleep(100);
            }
            //Cleanup CAN
            ShaCanDestroy(&Params);
      }

      //Set termination event
      SetEvent(hCanEvent);
      return 0;
}


void main(void)
{
      HANDLE hMem = NULL;
```

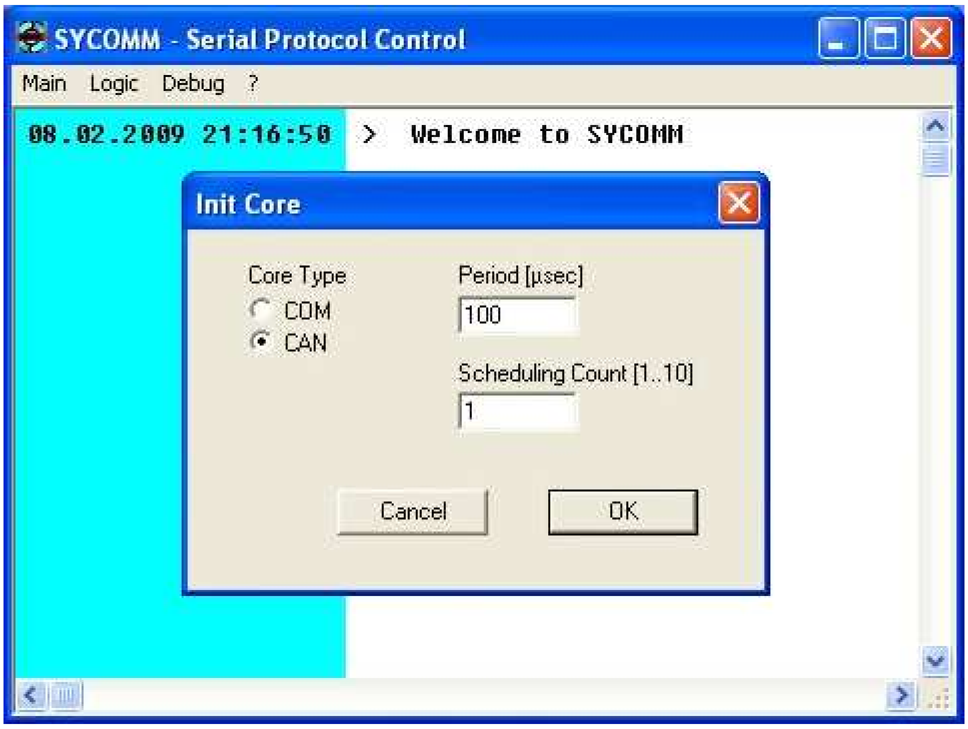
```
	DWORD MemBasePA;
	ULONG MemSize = sizeof(PLC_MEM);
	HANDLE hCanThread;

	printf("\n*** PLC Version 2.0 ***\n\n");

	//Allocate tagged memory
	if (ERROR_SUCCESS == ShaAllocMemWithTag(
		PLC_MEM_TAG,      //(Optional) Memory TAG for attachments
		IFT_PCIBUS,       //(Optional) Interface type
		TRUE,             //Set busmaster DMA
		FALSE,            //Allow cached physical memory
		0,                //(Optional) Required channel for system DMA
		&MemBasePA,       //Physical address
		(long*)&MemSize,  //Memory range
		(void**)&pUserPlc,//Pointer to user memory for USER mode
		(void**)&pSystemPlc,//Pointer to user memory for RING0 Execution
		&hMem))           //Handle to memory device
	{
		//Reset memory
		memset(pUserPlc, 0, sizeof(PLC_MEM));

		//Initialize CAN modules
		CanInit();

		//Create syncronisation events
		hTermEvent = CreateEvent(NULL,TRUE,FALSE,NULL);//Create manual
		                                               //resseting event
		hCanEvent = CreateEvent(NULL,FALSE,FALSE,NULL);//Create automatic
		                                               //resseting event
		if ((hTermEvent)  &&
			(hCanEvent))
		{
			//Create threads
			hCanThread = CreateThread(NULL,0,CanThread,NULL,0,NULL);
			if (hCanThread)
			{
				//Terminate application
				printf("Press 'q' to exit ...\n");
				while (getch() != 'q') { Sleep(100); }
			}

			//Syncronize thread termination
			SetEvent(hTermEvent);
			WaitForSingleObject(hCanEvent, INFINITE);
			ResetEvent(hTermEvent); //Reset manual event
		}
		//Release global tagged memory
		ShaFreeMem(hMem);
	}
}
```

3.7 Sample program (Standard COM Interface):

```
//*****************************************************************
//
//    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) 2006 SYBERA
//
//*****************************************************************
#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include "c:\can\ShaCanCore.h"

//Define CAN messages
#define CAN_OP          "\x01\x00\x00\x00\x00\x00\x00\x00"
#define CAN_STOP        "\x02\x00\x00\x00\x00\x00\x00\x00"
#define CAN_PRE_OP      "\x80\x00\x00\x00\x00\x00\x00\x00"
#define CAN_INIT        "\x81\x00\x00\x00\x00\x00\x00\x00"
#define CAN_RESET       "\x82\x00\x00\x00\x00\x00\x00\x00"
#define CAN_OUTPUTS_ON  "\x0F\x00\x00\x00\x00\x00\x00\x00"
#define CAN_OUTPUTS_OFF "\x00\x00\x00\x00\x00\x00\x00\x00"

//Define CAN addresses
#define FIXED_CAN_ADDR       0x05
#define DI_CAN_ADDR          0x180 + FIXED_CAN_ADDR
#define DO_CAN_ADDR          0x200 + FIXED_CAN_ADDR
#define AI_CAN_ADDR          0x380 + FIXED_CAN_ADDR
#define AO_CAN_ADDR          0x400 + FIXED_CAN_ADDR


HANDLE hComm = NULL;


BOOLEAN CommSet(void)
{
      DCB   Dcb;
      COMMTIMEOUTS Cto;

      //Init DBC structure
      Dcb.fBinary = TRUE;

      //Set COM parameters
      Dcb.BaudRate     = CAN_BAUD_500K;
      Dcb.StopBits     = ONESTOPBIT;
      Dcb.ByteSize     = 8;
      Dcb.Parity       = NOPARITY;
      Dcb.fParity      = (Dcb.Parity == NOPARITY) ? FALSE : TRUE;

      //Set hardware flow control
      Dcb.fDtrControl        = DTR_CONTROL_HANDSHAKE; //DTR_CONTROL_ENABLE;
      Dcb.fRtsControl        = RTS_CONTROL_HANDSHAKE; //RTS_CONTROL_ENABLE;
      Dcb.fOutxDsrFlow  = FALSE;
      Dcb.fOutxCtsFlow  = FALSE;

      //Set DCB settings
      if (!(SetCommState(hComm, &Dcb)))
            return FALSE;

      //Set timeout behaviour for Overlapping
      Cto.ReadIntervalTimeout = -1;
      Cto.ReadTotalTimeoutMultiplier = 0;
      Cto.ReadTotalTimeoutConstant = 1000;
      Cto.WriteTotalTimeoutMultiplier = 0;
      Cto.WriteTotalTimeoutConstant = 0;

      //Set TIMEOUT settings
      if (!(SetCommTimeouts(hComm, &Cto)))
            return FALSE;

      //Everything is OK
      return TRUE;
}


void main(void)
{
      CAN_FRAME Frame;
      ULONG BytesWritten;

      printf("*** CAN Core Test for COM interface ***\n\n");

      //open COM file (Note: Names of COM ports > 10 need to be prefixed by
      "\\\\.\\")
      hComm = CreateFile(
                  "\\\\.\\COM18",
                  GENERIC_WRITE | GENERIC_READ,
                  0, NULL,
                  OPEN_EXISTING,
                  0, NULL);

      if (hComm != INVALID_HANDLE_VALUE)
      {
            //Set COM parameters
            if (CommSet())
            {
                  //Purge all communication buffers
                  if (PurgeComm(hComm, PURGE_TXCLEAR | PURGE_RXCLEAR))
                  {
                        //Set and send init frame
                        CAN_SET_FRAME11(&Frame, 0, CAN_OP, 8);
                        if (WriteFile(hComm, Frame.bit8, sizeof(CAN_FRAME),
                                    &BytesWritten, NULL))
                        {
                              printf("Press any key to exit ...\n");
                              while (!kbhit())
                              {
                                    //Set all outputs
                                    CAN_SET_FRAME11(&Frame, DO_CAN_ADDR,
                                                      CAN_OUTPUTS_ON, 8);
                                    if (!(WriteFile(hComm, Frame.bit8,
                                                            sizeof(CAN_FRAME),
                                                            &BytesWritten, NULL)))
                                          break;

                                    //Do some delay
                                    Sleep(100);

                                    //Set all outputs
                                    CAN_SET_FRAME11(&Frame, DO_CAN_ADDR,
                                                            CAN_OUTPUTS_OFF, 8);
                                    if (!(WriteFile(hComm, Frame.bit8,
                                                            sizeof(CAN_FRAME),
                                                            &BytesWritten, NULL)))
                                          break;

                                    //Do some delay
                                    Sleep(100);
                              }
                        }
                  }
            }
            //Close COM file
            CloseHandle(hComm);
         }
      }
```

4 SYCOMM Protocol Control

The SYCOMM Software is used for contolling and analyzing serial protocols under realtime conditions. Thereby all protocols can be managed by an easy-to-use textbased SCRIPT language which is parsed at runtime. Additionally a singlestep mode allows comfortable protocol debugging.



```
08.02.2009 21:16:50  >  Welcome to SYCOMM
```

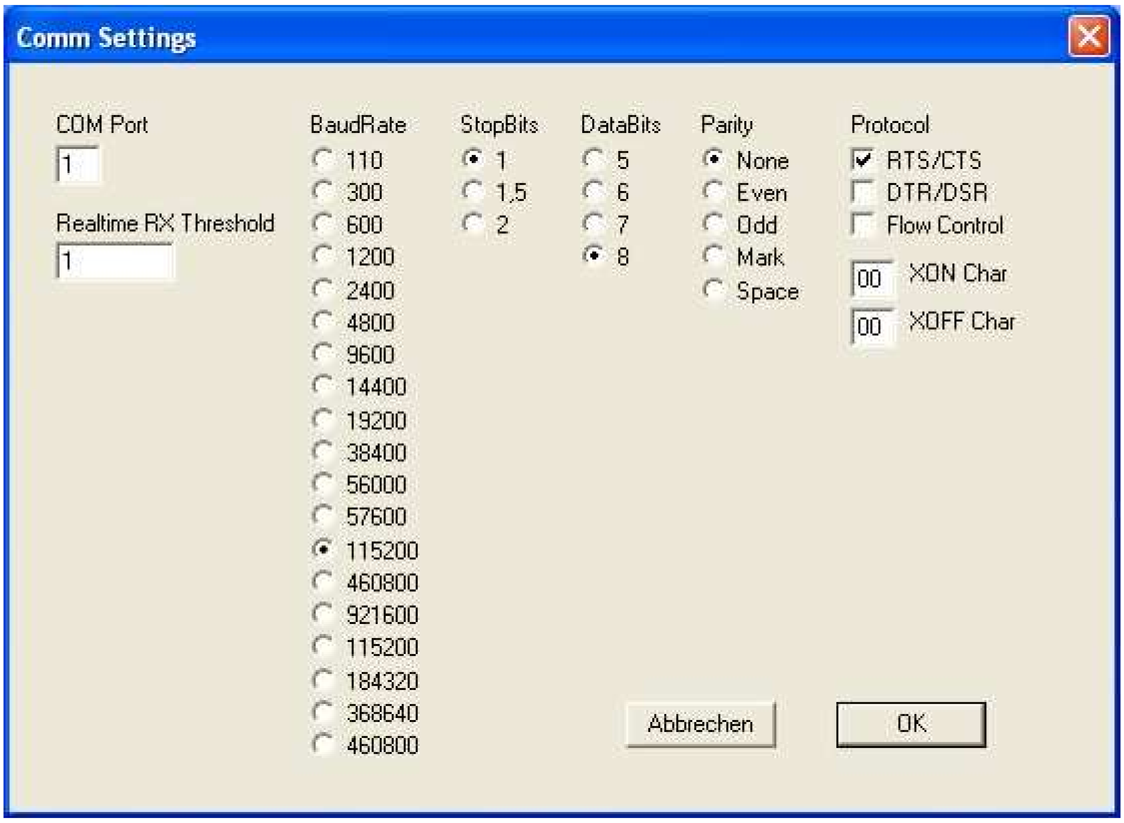
Core Type
( ) COM
[✓ (•) CAN]

Period [µsec]: 100

Scheduling Count [1..10]: 1

4.1 COMM Settings

When running SYCOMM a dialogbox for the serial communication settings appears. Withit all required settings for serial devices are handled. All parameters will be stored within the file COMM.PAR.



Comm Settings

COM Port: 1

Realtime RX Threshold: 1

BaudRate: 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 56000, 57600, [✓ 115200], 460800, 921600, 115200, 184320, 368640, 460800

StopBits: [✓ 1], 1,5, 2

DataBits: 5, 6, 7, [✓ 8]

Parity: [✓ None], Even, Odd, Mark, Space

Protocol: [✓ RTS/CTS], DTR/DSR, Flow Control

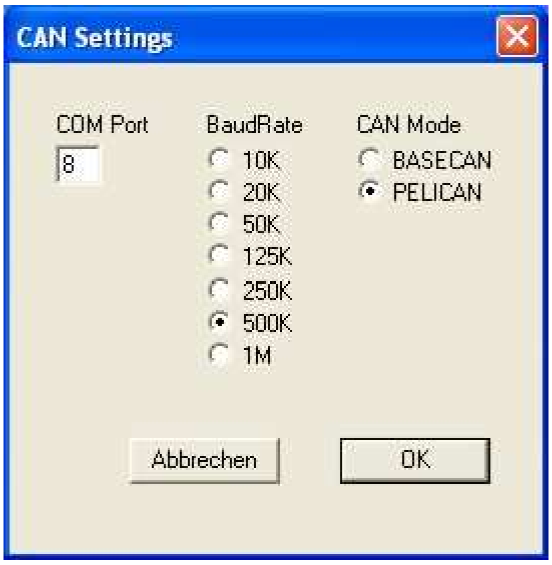
00 XON Char

00 XOFF Char

Abbrechen OK

4.2 CAN Settings

When running SYCOMM a dialogbox for the serial communication settings appears. Withit all required settings for serial devices are handled. All parameters will be stored within the file CAN.PAR.



CAN Settings

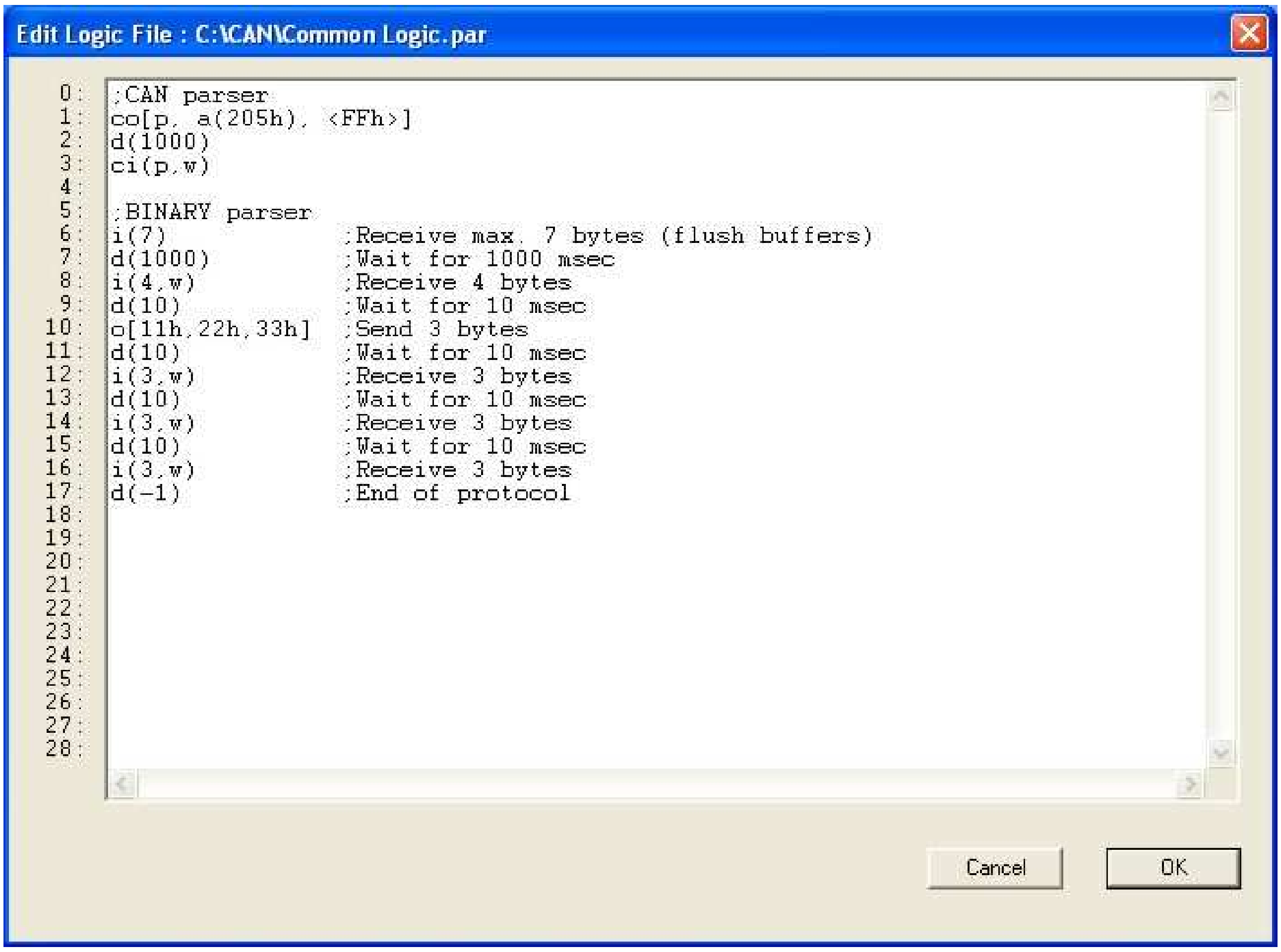
COM Port
8

BaudRate
10K
20K
50K
125K
250K
[✓ 500K]
1M

CAN Mode
BASECAN
[✓ PELICAN]

4.3 Protokoll SCRIPT

The control of the serial protocol is managed by the a simple SCRIPT language. The SCRIPT language allows control serial HOST and SLAVE protocols.



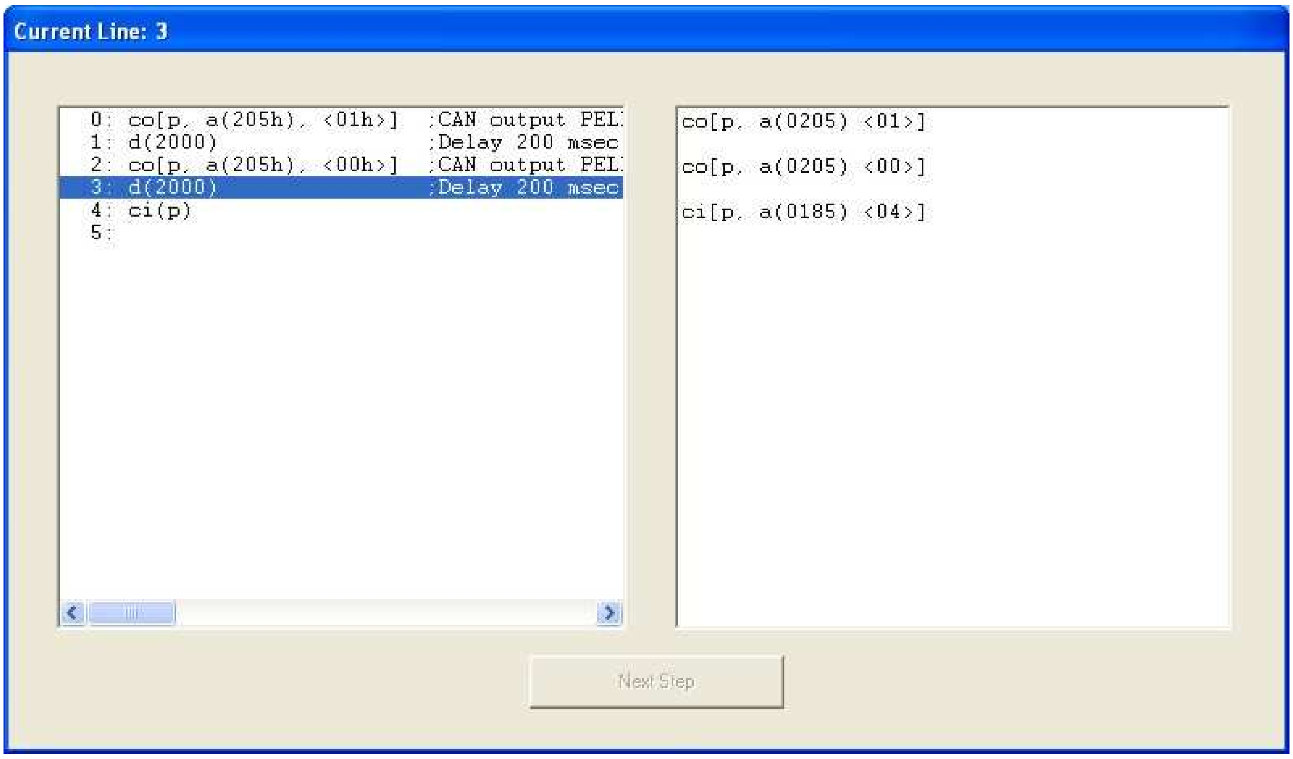
```
;CAN parser
co[p, a(205h), <FFh>]
d(1000)
ci(p,w)

;BINARY parser
i(7)               ;Receive max. 7 bytes (flush buffers)
d(1000)            ;Wait for 1000 msec
i(4,w)             ;Receive 4 bytes
d(10)              ;Wait for 10 msec
o[11h,22h,33h]     ;Send 3 bytes
d(10)              ;Wait for 10 msec
i(3,w)             ;Receive 3 bytes
d(10)              ;Wait for 10 msec
i(3,w)             ;Receive 3 bytes
d(10)              ;Wait for 10 msec
i(3,w)             ;Receive 3 bytes
d(-1)              ;End of protocol
```

Comments are separated by [;] of the commands, but at each line only one command is allowed. Following script commands are valid:

`i(x)`	: Receiving max.x bytes (dec)
`i(xh)`	: Receiving max. x bytes (hex)
`i(x,w)`	: Receiving x bytes and wait until completion
`i(x,o)`	: Receiving x bytes (only once)
`o[x,y,z,…]`	: Sending output bytes
`o[x,y,z,…, o]`	: Sending output bytes (only once)
`d(x)`	: Delay of x Cycle Counts (e.g. Period = 100µsec -> x * 1000 msec)
`d(x,o)`	: Delay of x Cycle Counts (only once)
`d(-1)`	: Stop the protocol
`co[p, a(xh), <x1, x2, x3, …, x8>]`	: Output PELICAN mode Address x (hex) SDO data (upto 8 Bytes)
`co[b, a(x), <x1, x2, x3, …, x8>]`	: Output BASECAN mode Address x (dec) SDO data (upto 8 Bytes)
`co[b, a(x), <x1, x2, x3, …, x8>, o]`	: Output BASECAN mode Address x (dec) SDO data (upto 8 Bytes) Only once
`ci(p)`	: Input PELICAN mode
`ci(p,w)`	: Input PELICAN mode and wait until receiption
`ci(p,w,o)`	: Input PELICAN mode and wait until receiption (only once)
`ci(b,w)`	: Input BASECAN mode

4.4 Step Mode Monitor

The step mode monitor (left window) displays the current line to be executed. To keep track when debugging asynchronous protocols it may be required to step through the protocol. Therefor the SingleStep mode can be activated in the menu [Protocol]. Each step will proceed by any key press. In SingleStep mode a line may be selected to be executed. This will is done by simple clicking on the corresponding line.



```
0: co[p, a(205h), <01h>]   ;CAN output PEL
1: d(2000)                 ;Delay 200 msec
2: co[p, a(205h), <00h>]   ;CAN output PEL
3: d(2000)                 ;Delay 200 msec
4: ci(p)
5:
```

```
co[p, a(0205) <01>]

co[p, a(0205) <00>]

ci[p, a(0185) <04>]
```

The Protocol Data Monitor (right window) allows to check the CAN protocol data sequence. Each line displays its current content.