CAN and Windows CE

Using CAN within Windows CE - Using Windows CE for CAN

ir. M.W. Nelisse
TNO Institute of Applied Physics, Delft, The Netherlands

CAN is a successful fieldbus, used in many different application areas. Windows CE is a rather new product in the family of MS Windows operating systems. Windows CE targets not only PC companion devices, like handheld PC's, palm PC's and auto PC's, but also true embedded systems, like point-of-sale terminals and set-top boxes. For both fields the combination of CAN and Windows CE offer interesting possibilities. The handheld and palm PC devices can be used by technicians as maintenance tools, for example to spy on the ongoing communication in a CAN bus system, or can be used by system operators as small but elaborated terminals, for example to change parameters and process control settings. To facilitate the development of CAN based Windows CE applications, an extension to TNO's CAN Development Environment (CDE) was made. This CDE now has the possibility to run CAN sub-systems as a set of parallel running Windows applications communicating over a virtual CAN bus. As a result development times of CAN applications can be shortened and it makes hardware/software in the loop simulations of CAN systems much easier.

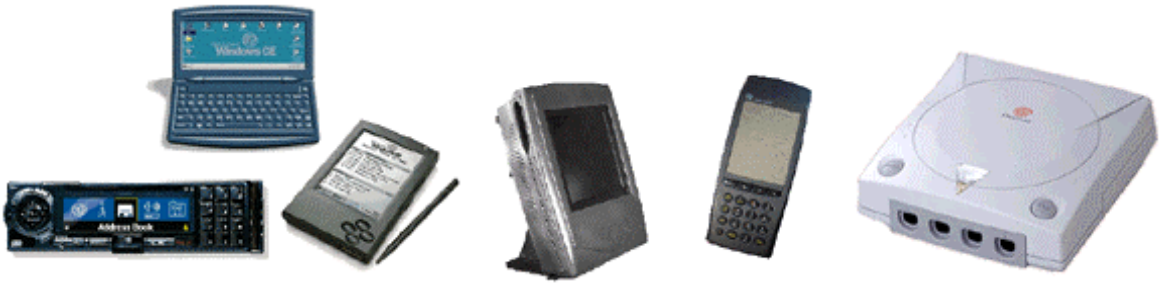
Introduction

Both CAN and MS Windows are very successful products within their application fields: fieldbus systems versus personal computing. Currently within CAN bus systems, MS Windows is mainly utilised as a user interface shell for engineering, maintenance and presentation applications. Due to the resource requirements of Windows 95/98/NT, the typical computers running these applications are desktop, laptop or some times embedded PC's. However many CAN bus nodes are build around dedicated computers, embedded systems, which very often have only a limited amount of resources. Windows CE is a rather new product in the family of MS Windows operating systems, which offers good possibilities also in the area of embedded systems.

Windows CE

In 1996 Microsoft introduced the Windows CE operating system on the market. From the beginning Windows CE was build around a sub-set of the well-known Win32 API, supported multiple hardware platforms, could be run from non-volatile memory and was designed to support real-time applications. In version 2.0 (released in 1997) Windows CE also got a customisable kernel and followed a modular concept in order to allow downscaling of the kernel to the features supported by a specific hardware platform. In an upcoming version (expected in 1999) more enhancement are announced especially in the hard real-time area.

As Windows NT can be seen as a super-set of Windows 95/98, Windows CE can be seen as a sub-set. However unlike Windows 95/98/NT, Windows CE not only targets PC like devices, called PC companions, but also true embedded computer systems.



PC companion devices

Within Windows CE three typical PC companion devices are defined now:

- handheld PC's
- palm PC's
- auto PC's

The handheld PC is a computer with a small form-factor and meant for mobile usage. Therefore it usually contains pocket versions of programs like Word, Excel, PowerPoint, Outlook, Explorer and Internet Explorer and data on the handheld PC can be easily replicated with data on the desktop PC. The user interface also resembles the shell used on Windows 95/98/NT. The device typically contains a QWERTY keyboard, small touch screen display, serial port, IrDA infrared port and very often also a PCMCIA slot, microphone and speaker.

The palm PC is meant to give access to vital business and personal information. Therefore it primarily offers information available from the Calendar, Contacts, Tasks, Inbox and Internet tools and data can again be replicated with a desktop PC. Because of the role as a 'personal information manager' the device has (compared to a handheld PC) a smaller form-factor, a smaller screen and lacks a keyboard (although it has some special function switches).

The auto PC is an integrated information and digital audio system for the automobile. It allows control of standard audio features like radio and CD-ROM, but also offer access to electronic mail, real-time traffic reports, route planning and with a cell-phone interface to conduct telephone conversations. This can all be done hands-free via speech-recognition and computer speech output. These kinds of devices therefore typically have a small screen, navigational keys, numeric keypad, AM/FM tuner, CD-ROM player, serial port, IrDA infrared port and a USB port.

Embedded systems

Compared with Windows 95/98/NT, Windows CE offers completely new possibilities especially in the area of embedded systems. Not only does it support multiple platforms (based on 80x86, SH3/SH4, MIPS, ARM and PowerPC processors), could be run from non-volatile memory (no hard-disk required) and was designed to support real-time applications, but it also support a mechanism to create a kernel only consisting of the modules necessary for the features supported by the specific hardware platform.

These points together now make it better possible to select a hardware environment for an embedded system with real-time requirements but without huge resource requirements, and still running a Windows based operating system. Current applications include point-of-sale terminals, navigation systems, game players, set-top boxes, web-phones, etc.

Windows CE and CAN

These above-mentioned characteristics make Windows CE an interesting operating system for both CAN maintenance tools and CAN embedded devices. For example: it can be used as the operating system in a maintenance tool running on a handheld PC, or it can be used as the kernel in a dedicated embedded computer system. Maybe the auto PC can be seen as a good example in how Windows CE integrates into embedded systems like a car radio, CD-ROM player, navigation system and maybe in the near future also with in-car based CAN devices.

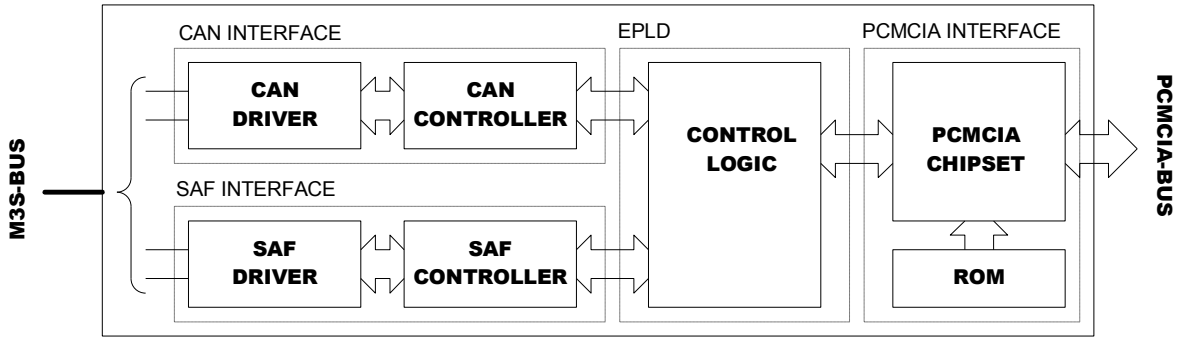
Within TNO-TPD we have been using the CAN bus in several applications since 1991, and Windows CE since 1997. Currently our main activities on CAN based Windows CE developments take place within the European TIDE-ICAN project and focus on the M3S communication standard (ISO 7176-17) for rehab devices, like wheelchairs, manipulators, environmental control units, joysticks, etc. Typical developments within this project include hardware interfacing, device drivers and applications programs.

Hardware interfacing

Within the project it was decided to use Windows CE based handheld PC's as the target platform. Since the commercially available handheld PC's didn't contain a M3S interface port and no M3S interface cards were available for the PCMCIA slots present on these devices, it was necessary to start the development of a PCMCIA (PC-CARD) based M3S interface card. However for other embedded Windows CE systems like PC-104 based computers, existing M3S and CAN interface cards could be viable solutions.

Based on experiences with other interface cards, the relatively low bus speed of the M3S bus (250 kbits/s) and the queuing capabilities of the SJA1000 CAN controller, it was decided not to put a microcontroller on the interface card. This then means that the processor on the handheld PC has to take care of retrieving the messages from the CAN controller in time.

The architecture of the M3S interface card is shown below.



Four main blocks can be identified:

- CAN interface
- SAF interface
- PCMCIA interface
- EPLD control logic

The CAN interface uses two standard components: a SJA1000 CAN controller and a 82C251 CAN driver. The SAF interface contains some analog circuitry necessary to access the additional safety lines present in the M3S bus. The PCMCIA interface takes care of the proper signal handling on the PCMCIA bus and it consists of the Zilog Z1601720 chip-set together with a serial ROM holding the configuration information of the card. The EPLD control logic takes care of interfacing the CAN and SAF blocks to the PCMCIA block and consists of an in-circuit re-programmable Altera EP7160S.

From the PCMCIA side the M3S interface card behaves as a dual-function card: a CAN interface and a SAF interface. All CAN controller registers are mapped directly into the memory space, while for the SAF interface a command, status, interrupt enable and interrupt status register are available in the memory space. The PCMCIA card interrupt is derived from both the CAN and SAF interrupt sources.

Device driver development

Windows CE supports two types of device drivers. Built-in device drivers are for devices which are built into a given Windows CE platform, like keyboard and touch-screen. Installable device drivers are for peripheral devices which can be connected to a Windows CE platform, like PCMCIA cards and USB devices. Since the M3S interface is based upon the PCMCIA standard, the driver was written as an installable device driver via a dynamically linked library (DLL). Actually two independent driver parts were created. One-part handles the accesses to the CAN bus interface, while the other part handles the accesses to the SAF bus interface. On top of this device driver, two sets of access functions were created to facilitate the calling of the low level Win32 API functions.

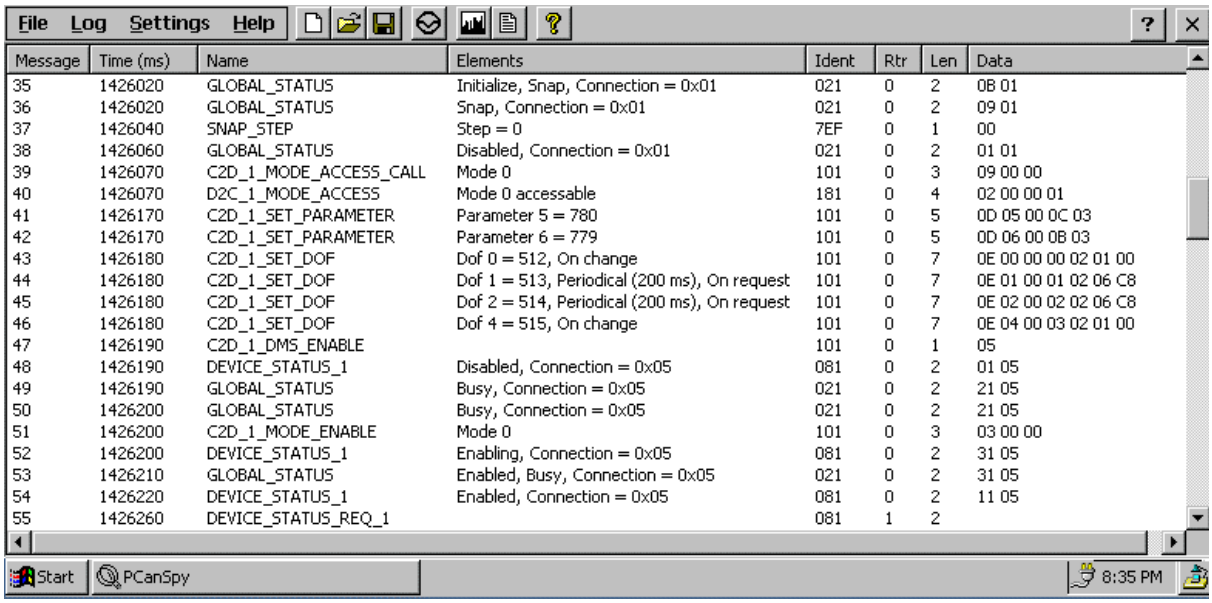
The CAN part uses an interrupt driven mechanism to receive and transmit messages. To prevent overruns when an application program runs slow, all messages are buffered in software FIFO queues located between the application program and the CAN controller.

The SAF part normally performs polled I/O accesses, but the interrupt facility allows an application to be signalled when a state change occurs on one or more of the SAF bus related signals.

Application program development

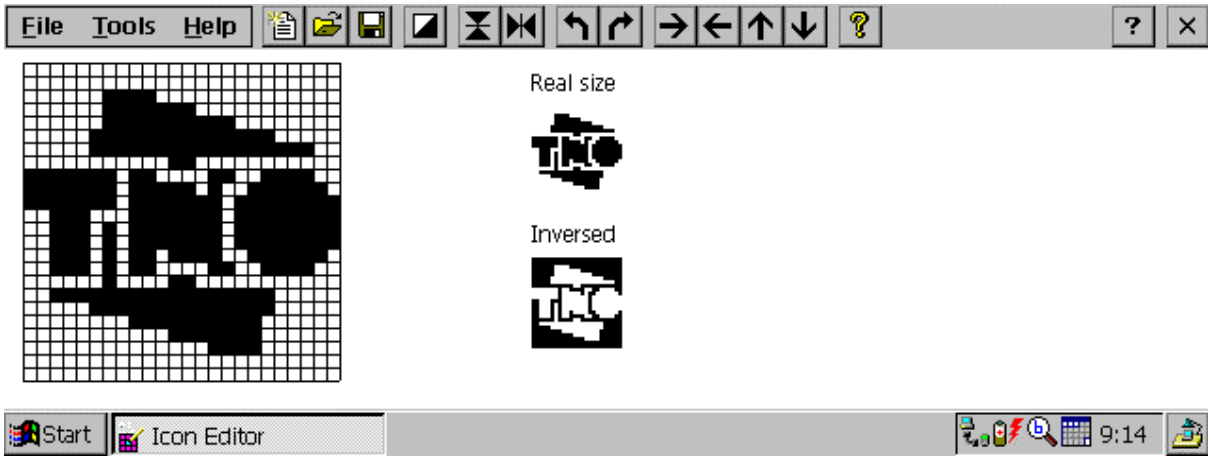
Several application programs are foreseen within the project. Some can be seen as maintenance tools for engineers, for example to spy on the M3S bus. Others can be seen as configuration tools for therapists, for example to change parameter settings and the graphical appearance of menus within the M3S system. However most applications fall in the category of M3S device programs.

A typical maintenance tool developed within the project is PCANSPY (Can Spy for Windows CE). This tool basically shows all messages appearing on the M3S bus. However smart filtering options are available, in which only certain groups of messages or messages for certain devices are shown. Furthermore the tool can show the meaning of the data within the messages according to the M3S protocol and it can give statistical information on the usage of the CAN bus. For logging purposes the tool allows recorded messages to be saved into several file formats and later on also to be retrieved and shown again.

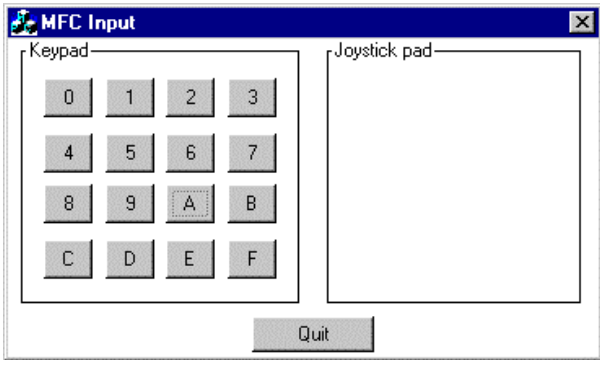


Message	Time (ms)	Name	Elements	Ident	Rtr	Len	Data
35	1426020	GLOBAL_STATUS	Initialize, Snap, Connection = 0x01	021	0	2	0B 01
36	1426020	GLOBAL_STATUS	Snap, Connection = 0x01	021	0	2	09 01
37	1426040	SNAP_STEP	Step = 0	7EF	0	1	00
38	1426060	GLOBAL_STATUS	Disabled, Connection = 0x01	021	0	2	01 01
39	1426070	C2D_1_MODE_ACCESS_CALL	Mode 0	101	0	3	09 00 00
40	1426070	D2C_1_MODE_ACCESS	Mode 0 accessable	181	0	4	02 00 00 01
41	1426170	C2D_1_SET_PARAMETER	Parameter 5 = 780	101	0	5	0D 05 00 0C 03
42	1426170	C2D_1_SET_PARAMETER	Parameter 6 = 779	101	0	5	0D 06 00 0B 03
43	1426180	C2D_1_SET_DOF	Dof 0 = 512, On change	101	0	7	0E 00 00 00 02 01 00
44	1426180	C2D_1_SET_DOF	Dof 1 = 513, Periodical (200 ms), On request	101	0	7	0E 01 00 01 02 06 C8
45	1426180	C2D_1_SET_DOF	Dof 2 = 514, Periodical (200 ms), On request	101	0	7	0E 02 00 02 02 06 C8
46	1426180	C2D_1_SET_DOF	Dof 4 = 515, On change	101	0	7	0E 04 00 03 02 01 00
47	1426190	C2D_1_DMS_ENABLE		101	0	1	05
48	1426190	DEVICE_STATUS_1	Disabled, Connection = 0x05	081	0	2	01 05
49	1426190	GLOBAL_STATUS	Busy, Connection = 0x05	021	0	2	21 05
50	1426200	GLOBAL_STATUS	Busy, Connection = 0x05	021	0	2	21 05
51	1426200	C2D_1_MODE_ENABLE	Mode 0	101	0	3	03 00 00
52	1426200	DEVICE_STATUS_1	Enabling, Connection = 0x05	081	0	2	31 05
53	1426210	GLOBAL_STATUS	Enabled, Busy, Connection = 0x05	021	0	2	31 05
54	1426220	DEVICE_STATUS_1	Enabled, Connection = 0x05	081	0	2	11 05
55	1426260	DEVICE_STATUS_REQ_1		081	1	2	

A typical configuration tool developed within the project is PICNEDIT (Icon Editor for Windows CE). This tool allows therapists to change the graphical appearance of menu items, which are presented to the user from a display within a M3S system.

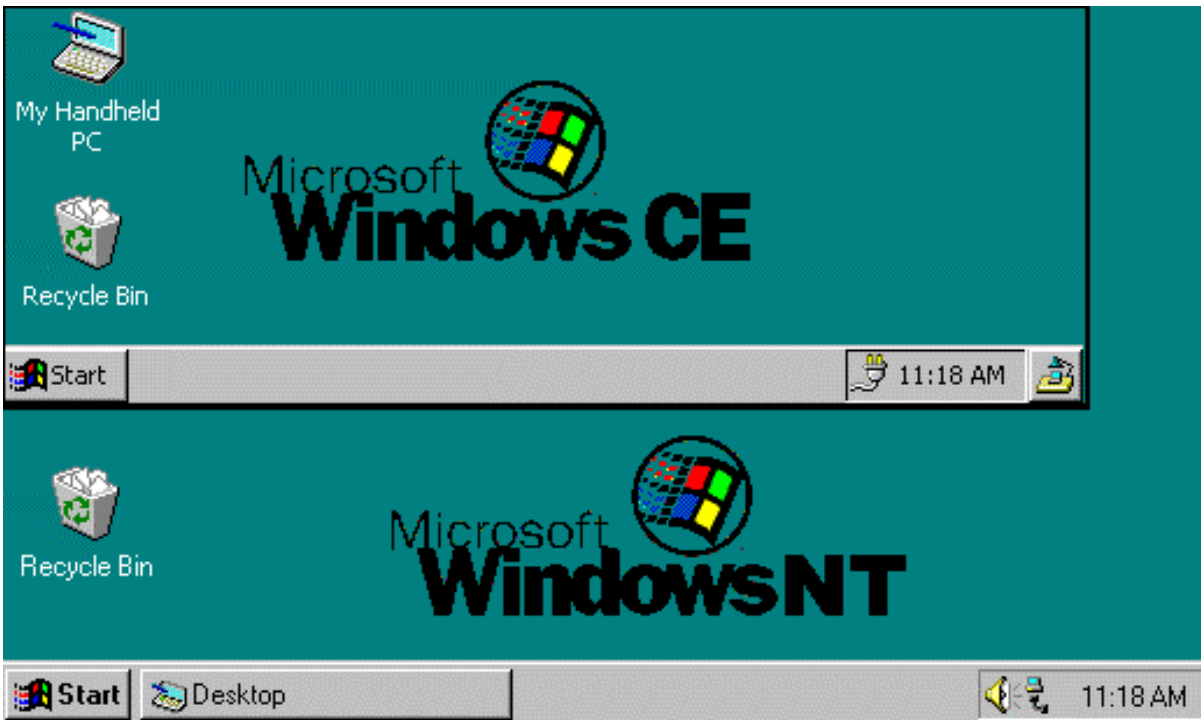


Several M3S device programs are foreseen within the project, these include graphical display device with menu selection, touch screen-input device, infrared environmental control device, telephone control device, etc.

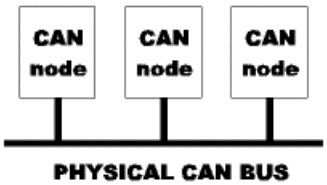


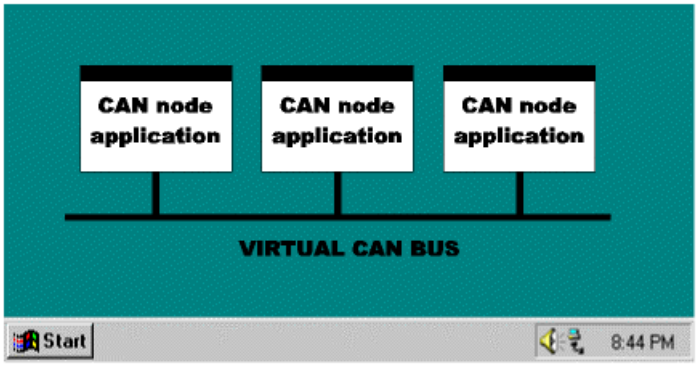
CAN Development Environment

Development of application programs for Windows CE normally takes place on a Windows 95/98/NT computer. Software writing can be done with special Windows CE toolkits for Visual C++, Visual Basic and Visual J++. Debugging of programs from these tools can be done remotely via a serial cable connection to the target platform or on a Windows NT computer using a Windows CE emulator.



Since our hardware and software developments took place in parallel, it was important to be able to debug CAN applications and higher-level protocol issues while the lower level hardware was not available at that time. Therefore an extension to TNO's CAN Development Environment (CDE) was made, in which a CAN system communicating over a physical CAN bus connections is replaced by parallel running Windows applications communicating over a virtual CAN bus.



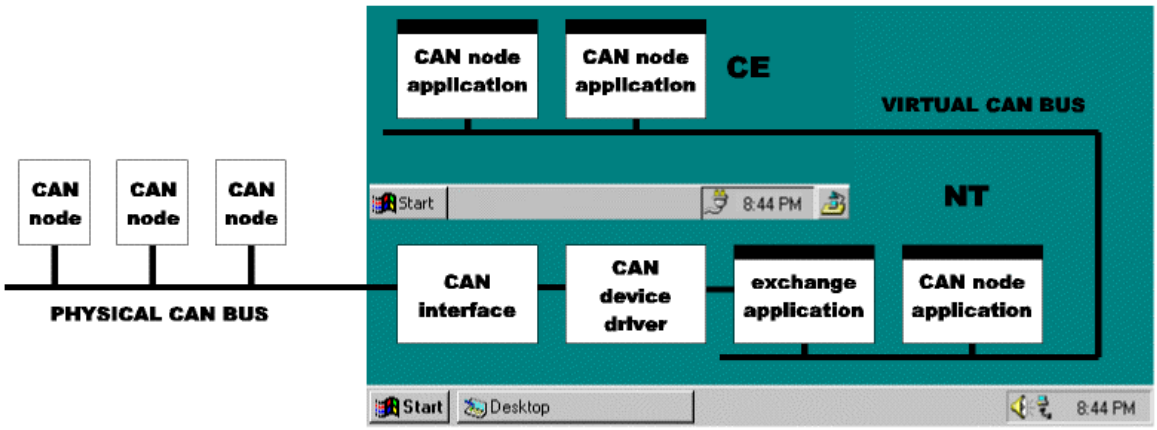


Because of the strict layering in the software of the CDE, it was rather easy to port the software modules normally taking care of message exchange via a physical CAN controller, to a mechanism in which the messages are distributed completely in software to other Window applications connected to the virtual bus.

This then allowed us to run a CAN based system on a single Windows computer and debug the applications. To investigate more in some higher-level protocol issues, we simply re-compiled our existing CAN spy and test programs with the virtual bus access module (in stead of a physical bus access module). We were then able to monitor ongoing CAN communication on the virtual bus, inject specific messages in a virtual CAN bus system and at the same time (using the Visual debuggers) see how the applications handle the protocol.

Initially this virtual CAN bus environment was written for Windows 95/98/NT computers but because Windows CE supports the same Win32 API functionality it was also possible to run the same virtual CAN bus mechanism on a Windows CE computer. Since Windows NT and the Windows CE emulator (running within Windows NT) share the same virtual bus, it was automatically possible to exchange virtual CAN messages between applications running under Windows NT and applications being debugged in the Windows CE emulator.

Later on the CDE environment was extended with a mechanism to connect the virtual CAN bus to a physical CAN bus outside the PC, via a CAN interface card with a CAN device driver and special application program taking care of the message exchange between these two busses. This then really allowed us to debug complex CAN systems in which parts of the system are run in a debugging environment, while other parts of the system are run on true target hardware (so called 'hardware in the loop' simulations).



The virtual CAN bus environment does not provide true system timing accuracy, since all virtual CAN nodes run as multi-tasking applications on the same processor. For the same reason access to the virtual CAN bus does not support arbitration, but is handled purely on a first-come first-serve basis. Also no disturbances on the virtual CAN bus are provided now, but these can be very easily added.

Since the CAN nodes are run as applications on a Windows platform it can mean that specific I/O functions for a certain CAN node may not be available. In those cases the module from the Hardware Abstraction Layer in the CDE is replaced by the identical module from the Hardware Emulation Layer, which then emulates the behaviour of the true I/O functions. This module can be a simple piece of source code, but also an interface to a model of a complete I/O system made with a tool like Matlab (so called 'software in the loop' simulations).

Conclusions

Windows CE is currently attracting lots of attention from different application fields. In the very near future CAN will be surely one of them. Within the TIDE-ICAN project, TNO-TPD worked

(together with some other project partners) on CAN related developments in the areas of hardware interfacing, device drivers and applications programs. Intermediate results of this project already show that Windows CE offer good possibilities for CAN based applications. The extension of TNO's CAN Development Environment with a virtual CAN bus, offered a big speed advantage in the development time of applications and makes hardware/software in the loop simulations of CAN systems much easier.

Literature

M3S dissemination office; *M3S home-page*; http://www.tno.nl/m3s/

Microsoft; *Windows CE home-page*; http://www.microsoft.com/windowsce

Nelisse, M.W.; *M3S, A general-purpose integrated and modular architecture for the rehabilitation environment*; Proceedings of the 2nd international CAN Conference; London, UK, October 3-4 1995; p. 10-2 - 10-9

Nelisse, M.W.*; CAN Development Environment; Proceedings of the 3rd international CAN Conference*; Paris, FRA, October 1-2 1996; p. 14-12 - 14-19

Nelisse, M.W. & J.A. van Woerden; *ICAN Integrated Communication and control for All Needs*; Improving the Quality of Life for the European Citizen; Proceedings of the 3rd TIDE Congress, Helsinki, FIN, June 23-25 1998, Assistive technology research series, vol. 4, p. 296-30

Correspondence

TNO Institute of Applied Physics
Attn. Martin Nelisse
PO Box 155
2600 AD Delft
The Netherlands

Phone: +31 15 2692323
Telefax: +31 15 2692111
E-mail: nelisse@tpd.tno.nl
Web: http://www.tno.nl