Digital I/O with eComStation via the USB Interface Box meM-PIO

by **Uwe Hinz**, 2008
uhdrelb@t-online.de

Introduction

- The number of USB devices out there is huge !
  For OS/2 or eCS only a small number of them is usable.
  A lot of USB drivers are still missing...
  In the commercially available OS/2 - DAC software SCADA ONSPEC 6.1 [4][5] some Digital I/O may be supported... It has not been checked yet !

- This presentation is a successive one of „**Digital I/O with eCommStation via the USB-Interface meM-PIO**“, held at DWS2008 in Düsseldorf.

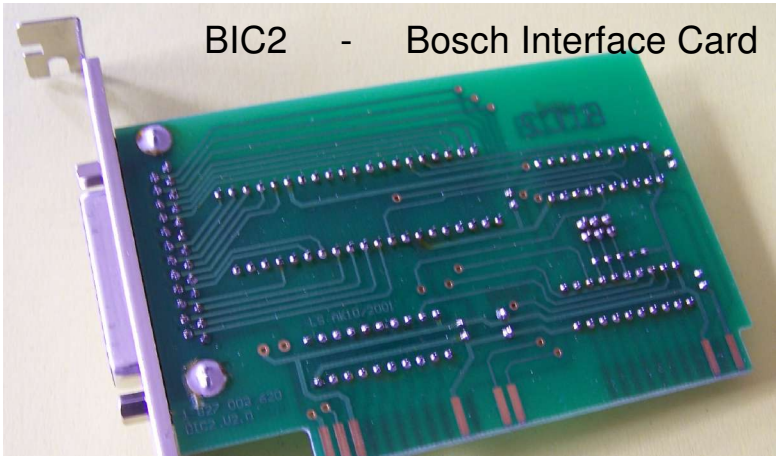
- It still puts the focus on DAC software and hardware for OS/2 or eCS, but this time with **meM-PIO** and a digital thermometer example.

Digital I/O with eComStation via the USB Interface Box meM-PIO

by Uwe Hinz, 2008

What had to be replaced by meM-PIO ?

Why **meM-PIO** instead of a normal ISA or PCI Add-On Card ?



- Traditional Add-On Cards do not work on eCS or OS/2 without their drivers. They are not available any more. **The big problem !**
- ISA motherboards are almost off the table.
- Developing cannot be done with modern laptop computers.

meM-PIO - by BMCM GmbH München (Munich)

On the other hand:

meM-PIO can be used with the legendary **Wim Brul Driver [1]**

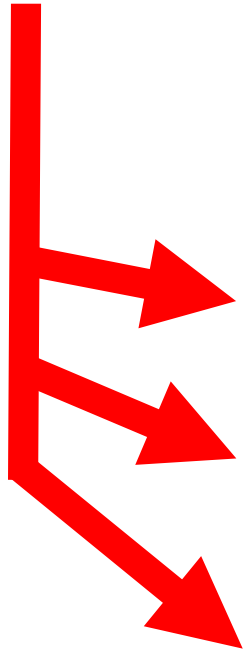
'usbecd.sys' . **This driver is General Purpose !**

Digital I/O with eComStation via the USB Interface Box meM-PIO

by Uwe Hinz, 2008

Wim Brul's device driver 'USBecd.sys' downloadable from hobbes

/pub/os2/system/drivers/misc



usb_uhci_fix01.zip	USB host controller driver (VIA and Intel chipsets) [More info]	1999/11/03	Compressed archive, 15.70Kb
usbecd00.zip	USB 1.1 Expanded Control Driver [More info]	2005/03/18	Compressed archive, 7.32Kb
usbecd10.zip	USB 1.1 Expanded Control Driver [More info]	2005/12/12	Compressed archive, 8.47Kb
usbecd10s.zip	USB 1.1 Expanded Control Driver - source/samples [More info]	2005/12/12	Compressed archive, 63.76Kb

Digital I/O with eComStation via the USB Interface Box meM-PIO

by Uwe Hinz, 2008

List of files in 'usbecd10.zip'

```
The volume label in drive C is VOLUME 3.
The Volume Serial Number is 2933:BC14.
Directory of C:\Desktop\work\USBecdDL\usbecd10

26.05.07 23.18          <DIR>      0 ----  .
26.05.07 23.18          <DIR>      0 ----  ..
11.04.05 10.31           1.875     0 a---  usbecd.sys
 6.12.05 14.25          18.668     0 a---  usbecd.txt
 6.12.05 19.14           2.062     0 a---  usbread.cmd
 6.12.05 19.13           9.635     0 a---  usbwrite.cmd
       6 file(s)      32.240 bytes used
                   1.499.275 K bytes free
```

the driver

the readme

the examples

by Uwe Hinz, 2008

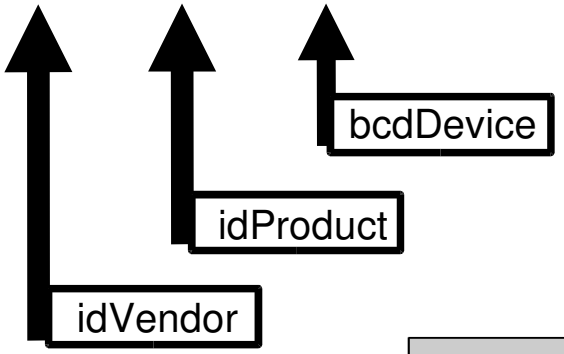
Wim Brul's USB device driver and eCS

Warpstock Europe 2008
Nov. 14-16, Düsseldorf, Germany

The way to use the driver goes here:

Put in **config.sys**, a set of three numbers complete the D parameter.
The three hex numbers make the particular device identifiable.

```
DEVICE=C:\USBDRVS\USBECD.SYS /D:0000:0000:0000 /N:$$$$$$$$ /S /V
```



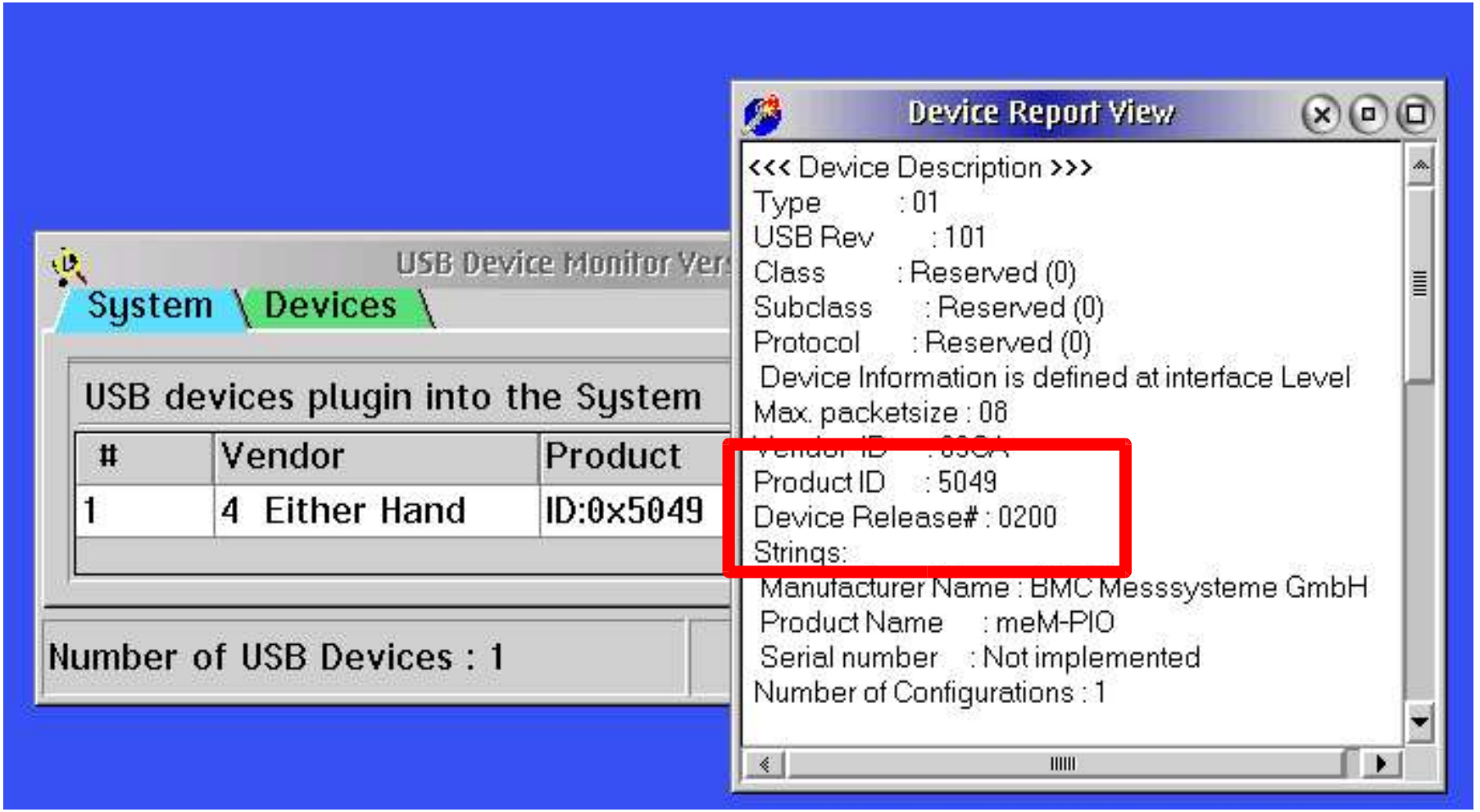
The three numbers can be checked with the eCS USB monitor easily

All parameters explained in **usbecd.txt**

Digital I/O with eComStation via the USB Interface Box meM-PIO

by Uwe Hinz, 2008

USB Device Monitor screenshot for meM-PIO



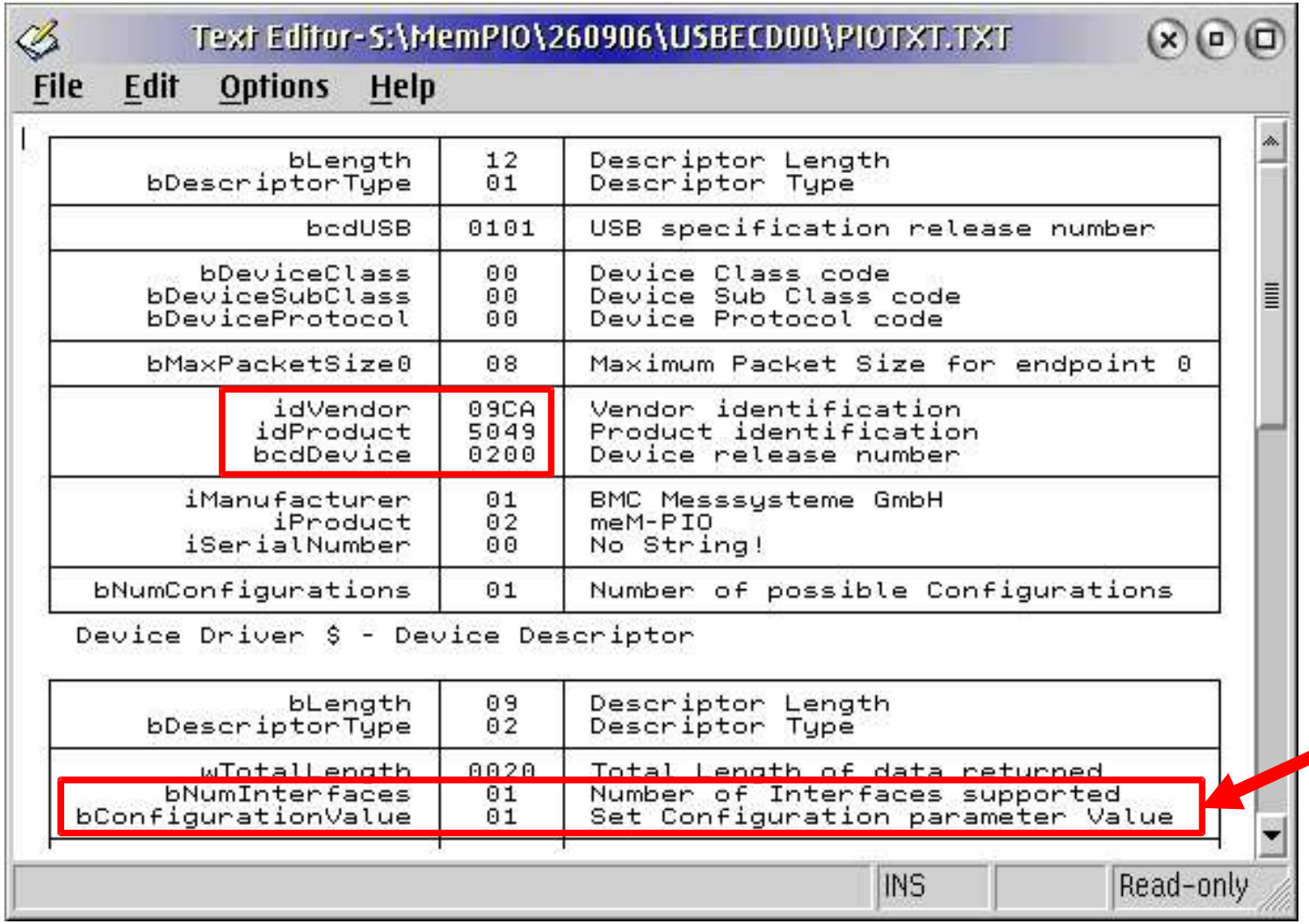

```
DEVICE=USBECD.SYS /D:09CA:5049:0200 /N:$ /S /V
```

Caution! Pitfall! The hex numbers in the D parameter must be upper case.

Digital I/O with eComStation via the USB Interface Box meM-PIO

by Uwe Hinz, 2008

Wim Brul's example - output of the REXX script 'usbwrite.cmd'



The USB-device has got how many interfaces ? How many configurations ? - Listed here !

by Uwe Hinz, 2008

The next level

After the Wim Brul examples run, the next question is:

- How do I know what I have to send to my USB device to make it work ?
- What can I expect, my USB device will send me back ?

If the manufacturer does not publish any appropriate specification, an OS different from eCS and a test application have to be used in order to observe the data exchange with **meM-PIO**

by Uwe Hinz, 2008

The USB analyser hardware

An USB analyser had to be ordered !

Market investigation
lead to the

Ellisys 110 .

Including the analyser
programme
Ellisys Visual USB

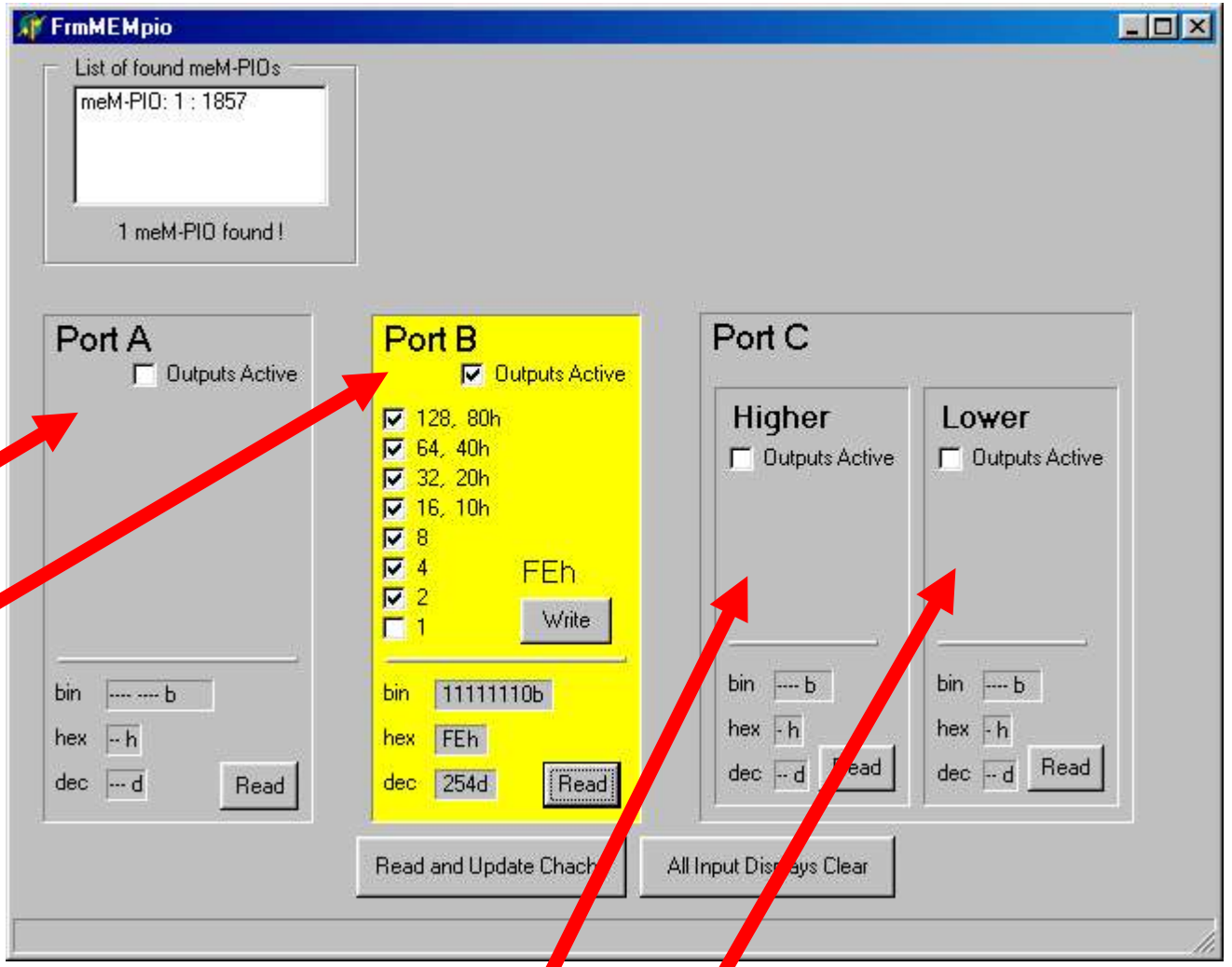
for Win2000, it promised
professional support.

Digital I/O with eComStation via the USB Interface Box meM-PIO

by Uwe Hinz, 2008

The meM-PIO test application in DELPHI

Win98 controlling **meM-PIO** and using a **DELPHI test application**, that helps to observe its initialisation and data exchange.



Port 1, Input Mode

Port 2, Output Mode

Port 3 (High and Low), Input Mode

Port A, Port B, Port C are equivalent to Port 1, Port 2, Port 3

Warpstock Europe XS 2008 Nov. 14-16, Düsseldorf, Germany

Digital I/O with eComStation via the USB Interface Box meM-PIO

by Uwe Hinz, 2008

Watching the data exchange

No additional wires or settings are needed.
meM-PIO data exchange will be investigated with Ellisys 110

Ellisys 110

[2]

USB cable

PC (any OS)

development takes place here

USB cable

USB cable

The analyser application

Ellisys Visual USB

(Win2000, XP)

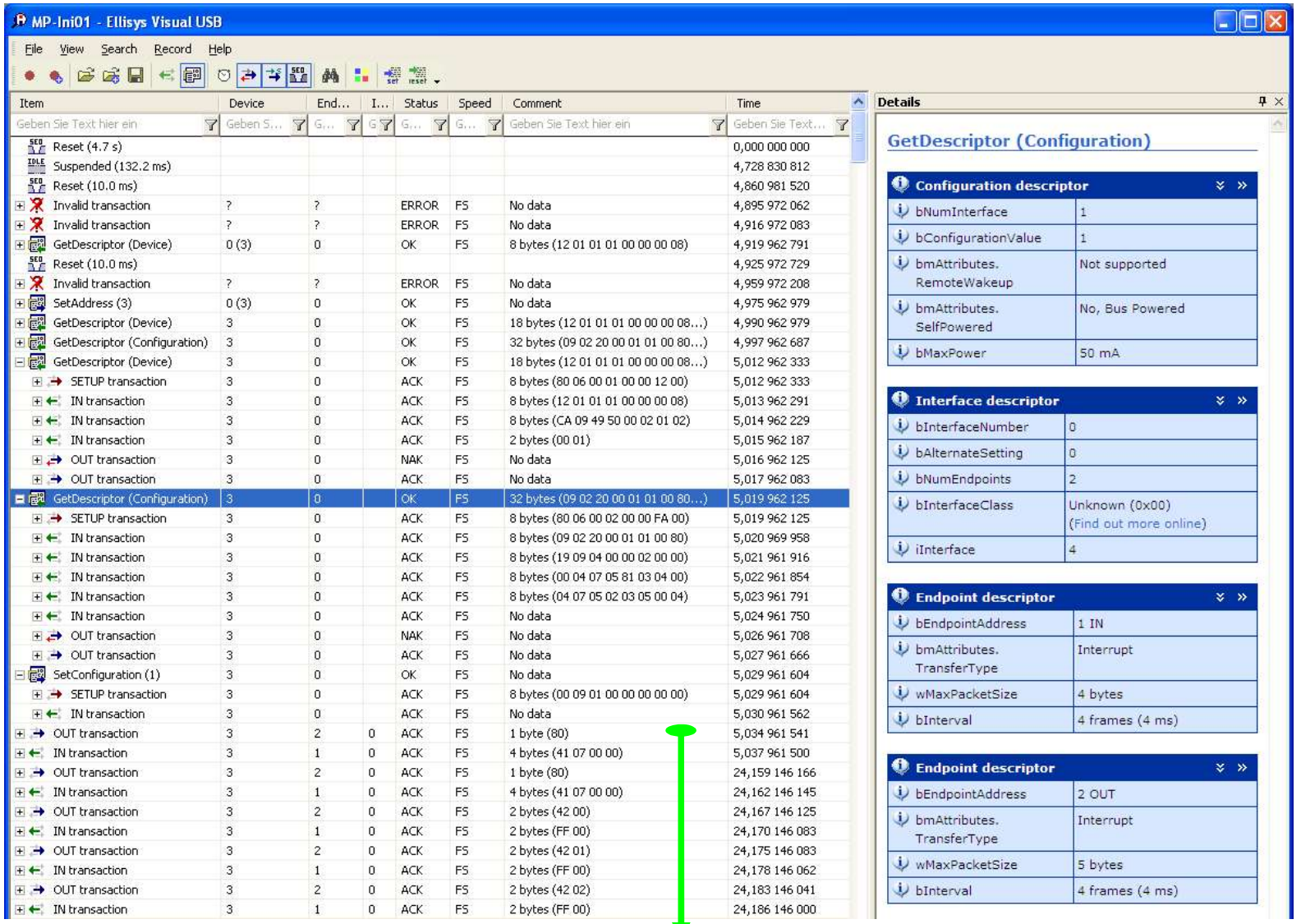
Digital I/O with eComStation via the USB Interface Box meM-PIO

by Uwe Hinz, 2008

Data exchange between the DELPHI test application and meM-PIO

1. Joining the Bus

2. Initialising meM-PIO



3. Data exchange

For details see [3].

by Uwe Hinz, 2008

Three types of Setup Packets can be identified

```
- Three types of Setup Packets:

GetDescriptor(Device)
          80 06 00 01 00 00 12 00

GetDescriptor(Config)
          80 06 00 02 00 00 20 00

SetConfiguration(1)
          00 09 01 00 00 00 00 00
```

by Uwe Hinz, 2008

Discovered meM-PIO commands and answers

Assigned Name	Command	Answer
WakeUp	80	41 07 00 00
InitPort	42 pp	FF 00
SetPortDir	34 pp dd 00	dd
WritePort	14 pp ww 00	ww 00
ReadPort	22 pp	rr 00

Port Number: pp = (0x00, 0x01, 0x02) 00:Port1,01:Port2 02:Port3
Port Direction: dd = (0x00, 0x0f, 0xf0, 0xff) F:OUT, 0:IN
Data to be written: ww = (0x00...0xff)
Data read: rr = (0x00...0xff)

by Uwe Hinz, 2008

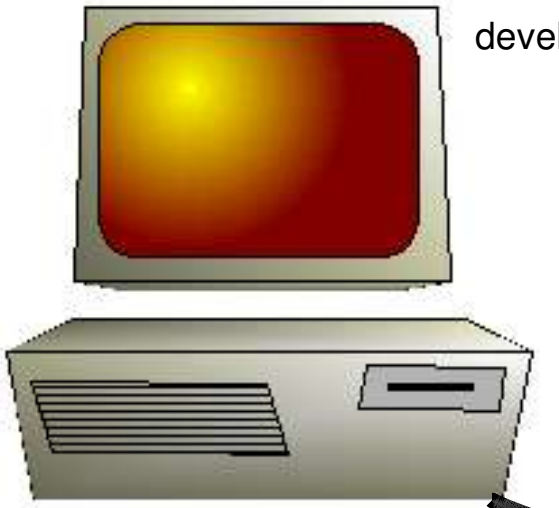
Further Developing with eCS after the data exchange is known

After data exchange between the OS and eCS is analysed and the meanings of messages are clear, development can go ahead with less equipment.

meM-PIO

PC (eCS)

development takes place here



USB port

USB cable

by Uwe Hinz, 2008

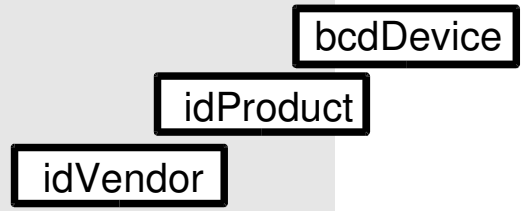
The setup part of the data exchange

```
- All meM-PIO Setup Requests and their answers

  GetDescriptor(Device)
      SETUP transaction: 80 06 00 01 00 00 12 00
         IN transaction: 12 01 01 01 00 00 00 08
         IN transaction: CA 09 49 50 00 02 01 02
         IN transaction: 00 01
               ;Device Descriptor 18 (12h) bytes answered.

  GetDescriptor(Config)
      SETUP transaction: 80 06 00 02 00 00 20 00
         IN transaction: 09 02 20 00 01 01 00 80
         IN transaction: 19 09 04 00 00 02 00 00
         IN transaction: 00 04 07 05 81 03 04 00
         IN transaction: 04 07 05 02 03 05 00 04
               ;Configuration Descriptor 32 (20h) bytes answered.

  SetConfiguration(1)
      SETUP transaction: 00 09 01 00 00 00 00 00
         IN transaction:  -  -  -  -  -  -  -  -
               ;No answer after configuration is set.
```



Digital I/O with eComStation via the USB Interface Box meM-PIO

by Uwe Hinz, 2008

What is in the 8 bytes

The meaning of every bit and every byte determines the variety of communication that is going on on the Universal Serial Bus.

One line of REXX code in Wim Brul s examples. The exact USB significance [3] added as comments.

```
  oiBuffer = X2C(80 06 00 01 00 00 12 00)
/*                |  |  |     |     ^^-^^------------------ wLength   */
/*                |  |  |     |     0012H(18):  Maximum Block Size.   */
/*                |  |  |     |        p. 189, 9.4.3 Get Descriptor   */
/*                |  |  |     ^^-^^------------------------ wIndex    */
/*                |  |  |     0000H:  Zero by definition.             */
/*                |  |  |        p. 189, 9.4.3 Get Descriptor         */
/*                |  |  ^^-^^------------------------------ wValue    */
/*                |  |  0100H = DEVICE:  Descriptor type              */
/*                |  |     p. 187, Table 9-5, 1st row                 */
/*                |  ^^---------------------------------- bRequest    */
/*                |  06 = GETDESCRIPTOR                               */
/*                |   p. 187, Table 9-4, 7th row                      */
/*               ^^-------------------------------- bmRequestType    */
/*               80h -> 1  00  00000b                                */
/*                      |  |   ^                                     */
/*                      |  ^   0 = Device                            */
/*                      ^  0 = Standard                              */
/*                      1 = Device-to-host                           */
/*                      p. 183, Table 9-2, 1st row                   */
```

by Uwe Hinz, 2008

How the descriptors are requested from the USB Device

```
/*-------------------------------------------------------*/
/* DeviceDescriptorReadOut                               */
sPayLoad = '................'
oiBuffer = X2C(80 06 00 01 00 00 12 00) || sPayLoad
```

Variable oiBuffer

80 06 00 01 00 00 12 00 2E 2E 2E 2E 2E 2E 2E 2E 2E 2E 2E 2E 2E 2E 2E 2E 2E 2E

CALL WriteAndReadProcedure

oiBuffer before CALL

80 06 00 01 00 00 12 00 12 01 10 01 00 00 00 08 03 04 01 60 00 04 01 02 03 01

...

oiBuffer after CALL

```
/* -------------------------------------------------------*/
/*                                                        */
   WriteAndReadProcedure:

      RC = CHAROUT(ddName,oiBuffer)
           ...
      RETURN
```

Digital I/O with eComStation via the USB Interface Box meM-PIO

by Uwe Hinz, 2008

The difference between Setup Packets and Parameter Packets

Warpstock Europe 2008 Nov. 14-16, Düsseldorf, Germany

Setup Packets can be observed on the Universal Serial Bus. They travel from the OS to the USB Device.

`80 06 xx xx xx xx xx xx`

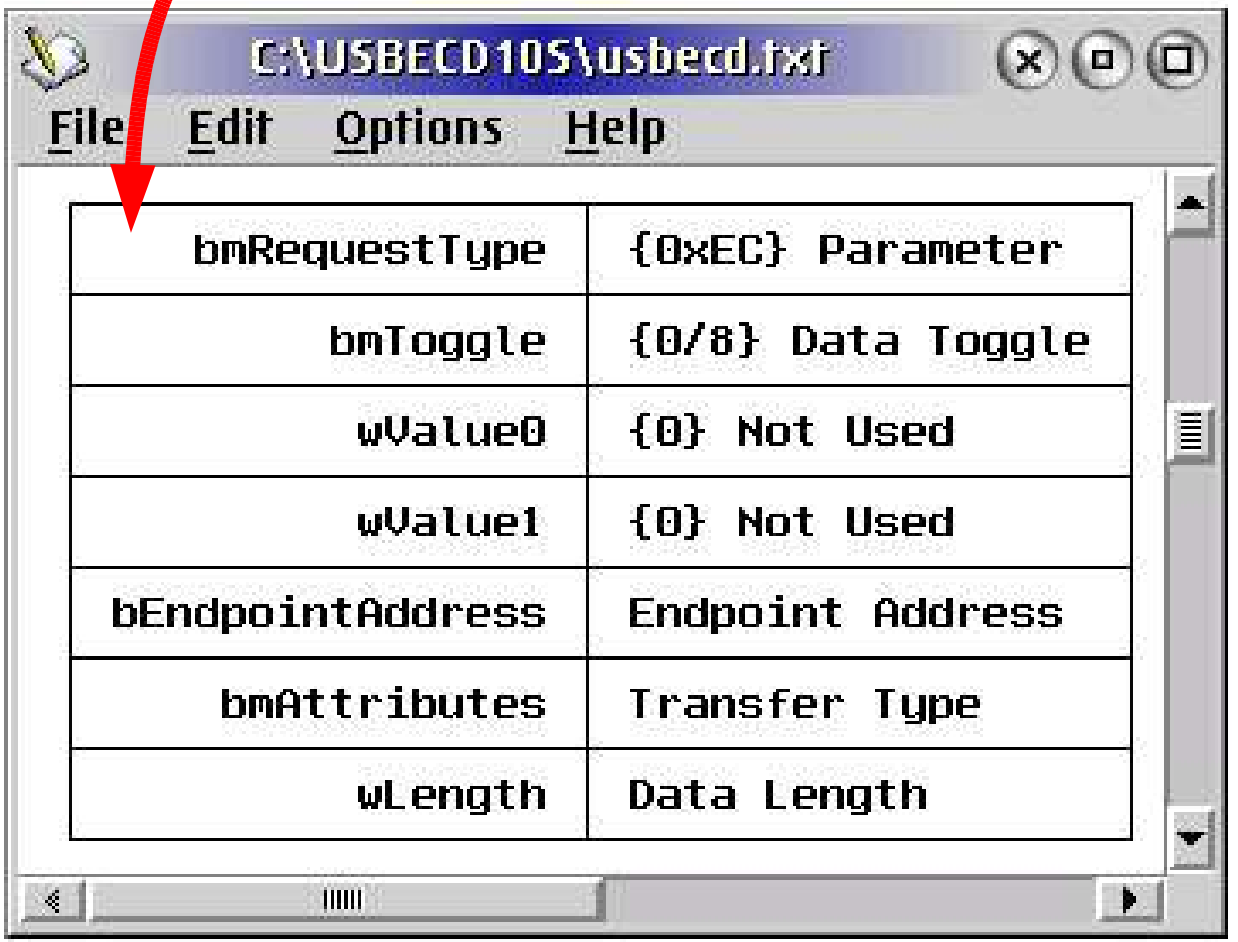
for setup

Parameter Packets are to carry the data. They do not appear on the bus. Just the data they carry do so.

`EC 0x 00 00 xx xx xx xx`

for data exchange

The USB driver wants **0xEC** in the request type byte to do data exchange instead of setup.



by Uwe Hinz, 2008

Basic form of Parameter Packets and how to complete it

- Basic type of Parameter Packets :

For sending or getting data

EC 0x 00 00 xx xx xx xx

xx is the unknown information.
It must be extracted from the
the configuration descriptor.
See next pages....

by Uwe Hinz, 2008

Where the missing Endpoint information is located

```
- All meM-PIO Setup Requests and their answers

  GetDescriptor(Device)
      SETUP transaction: 80 06 00 01 00 00 12 00
         IN transaction: 12 01 01 01 00 00 00 08
         IN transaction: CA 09 49 50 00 02 01 02
         IN transaction: 00 01
               ;Device Descriptor 18 (12h) bytes answered.

  GetDescriptor(Config)
      SETUP transaction: 80 06 00 02 00 00 20 00
         IN transaction: 09 02 20 00 01 01 00 80
         IN transaction: 19 09 04 00 00 02 00 00
         IN transaction: 00 04 07 05 81 03 04 00
         IN transaction: 04 07 05 02 03 05 00 04
               ;Configuration Descriptor 32 (20h) bytes answered.

  SetConfiguration(1)
      SETUP transaction: 00 09 01 00 00 00 00 00
         IN transaction:  -  -  -  -  -  -  -  -
               ;The only configuration set, no answer.
```

81 bEndpointAddress

03 bAttribute

02 bEndpointAddress

03 bAttribute

by Uwe Hinz, 2008

Two types of Parameter Packets are needed


```
- Two types of Parameter Packets :


 Getting data from the device
           EC 0x 00 00 81 03 xx xx

 Sending data to the device
           EC 0x 00 00 02 03 xx xx
```

Endpoint addresses (0x81, 0x02) and transfer type (0x03) are completed.
The remaining xx indicates the length of the trailing data. The maximum length is passed in the configuration descriptor too.
See next pages....

by Uwe Hinz, 2008

Where the missing Maximum Data Buffer Length is located

```
- All meM-PIO Setup Requests and their answers

  GetDescriptor(Device)
      SETUP transaction: 80 06 00 01 00 00 12 00
         IN transaction: 12 01 01 01 00 00 00 08
         IN transaction: CA 09 49 50 00 02 01 02
         IN transaction: 00 01
               ;Device Descriptor 18 (12h) bytes answered.

  GetDescriptor(Config)
      SETUP transaction: 80 06 00 02 00 00 00 20
         IN transaction: 09 02 20 00 01 01 00 80
         IN transaction: 19 09 04 00 00 02 00 00
         IN transaction: 00 04 07 05 81 03 04 00
         IN transaction: 04 07 05 02 03 05 00 04
               ;Configuration Descriptor 32 (20h) bytes answered.

  SetConfiguration(1)
      SETUP transaction: 00 09 01 00 00 00 00 00
         IN transaction:  -  -  -  -  -  -  -  -
               ;The only configuration set, no answer.
```

0x0004 wLength

0x0005 wLength

by Uwe Hinz, 2008

Parameter Packets ready for meM-PIO

- Parameter packets with maximum data buffer lengths inserted:
  (length of trailing data buffer)

Sending data to the device
EC 0x 00 00 02 03 05 00

Getting data from the device
EC 0x 00 00 81 03 04 00

Parameter packets in usable form. Trailing data are not mentioned here.
See next two pages that show complete parameter packets for sending and reading...

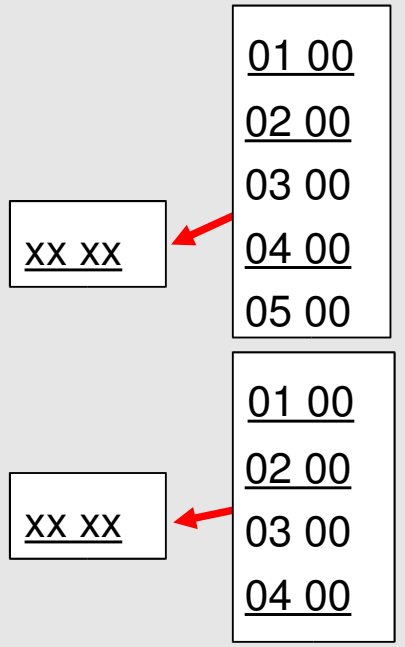
by Uwe Hinz, 2008

Data Buffer Length can vary

- Parameter packets with variable data buffer lengths inserted:
  (length of trailing data buffer)

Sending data to the device
EC 0x 00 00 02 03 xx xx

Getting data from the device
EC 0x 00 00 81 03 xx xx



Data Buffer Length can be variable.
Even commands and their immediate answers can have different lengths.
For example: Sent **0x80** will be answered **0x41 0x07 0x00 0x00**.
See next two pages...

by Uwe Hinz, 2008

Parameter Packet and data put together for sending

```
/*-------------------------------------------------------------------------*/
/* DataFromDeviceToDevice
sTrailingData = X2C(80)    /* 5 bytes maximum allowed */
oiBuffer = X2C(EC 08 00 00 02 03 01 00) || sTrailingData
```

0x80 to be sent

EC 08 00 00 02 03 01 00 80

Variable: oiBuffer

```
CALL WriteAndReadProcedure
```

DataToggleBit before CALL

```
...
```

EC 00 00 00 02 03 01 00 80

DataToggleBit after CALL

```
/* ------------------------------------------------------*/
/*                                                       */
   WriteAndReadProcedure:

      RC = CHAROUT(ddName,oiBuffer)
           ...
      RETURN
```

Caution! 0x80 is not the answer! See next page...

by Uwe Hinz, 2008

. . . and for reading the answer

```
/*-------------------------------------------------------------
/* DataFromDeviceToDevice
sTrailingData = '....'       /* 4 bytes maximum allowed */
oiBuffer = X2C(EC 00 00 00 81 03 04 00) || sTrailingData
```

EC 00 00 00 81 03 04 00 2E 2E 2E 2E

```
CALL WriteAndReadProcedure
...
```

EC 08 00 00 81 03 04 00 41 07 00 00

```
/* ------------------------------------------------------*/
/*                                                       */
   WriteAndReadProcedure:

      RC = CHAROUT(ddName,oiBuffer)
           ...
      RETURN
```

Waiting bytes to be read
(How many? It must be known before the length of the trailing data is set.)

Variable: oiBuffer

DataToggleBit before CALL

DataToggleBit after CALL

0x41 0x07 0x00 0x00 read

The four last bytes in **oiBuffer** have to be extracted.
Then data can be used !

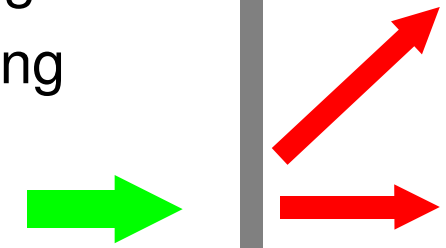
by Uwe Hinz, 2008

From Phase 1 to Phase 2

Phase 1

- Collecting knowlege about meM-PIO using Wim Brul's driver

- Testing the knowlege with a number of small REXX scripts



Phase 2

- Creating a Test Application with C and Markus Montkowski's USBCALLS [7] ?

- Creating a Test Application from the small REXX scripts ?

- Creating a new meM-PIO driver ?

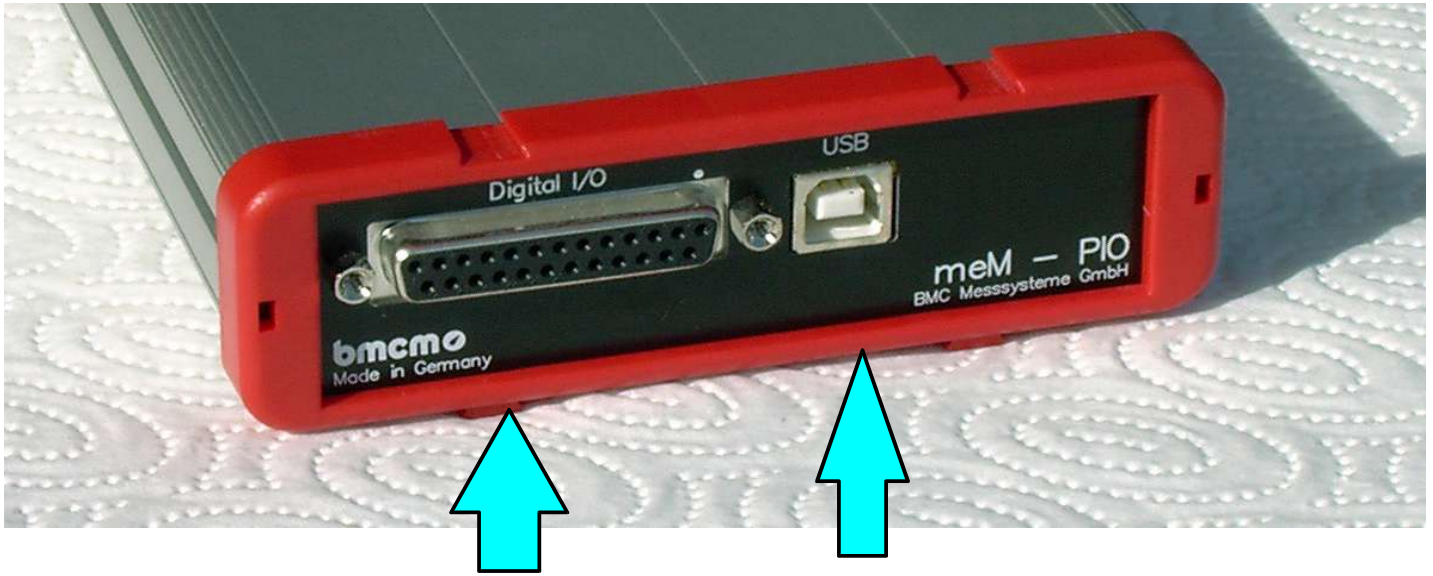
Digital I/O with eComStation via the USB Interface Box meM-PIO

by Uwe Hinz, 2008

meM-PIO and its front panel

meM-PIO [2] is an ideal USB Interface Box.

That particular device was bought in 2004. Its design year was probably 2002.



8 + 8 + 4 + 4 = 24 TTL compatible I/O lines.
Pin assignment identical to **BIC2** !

Powered by USB only

Digital I/O with eComStation via the USB Interface Box meM-PIO

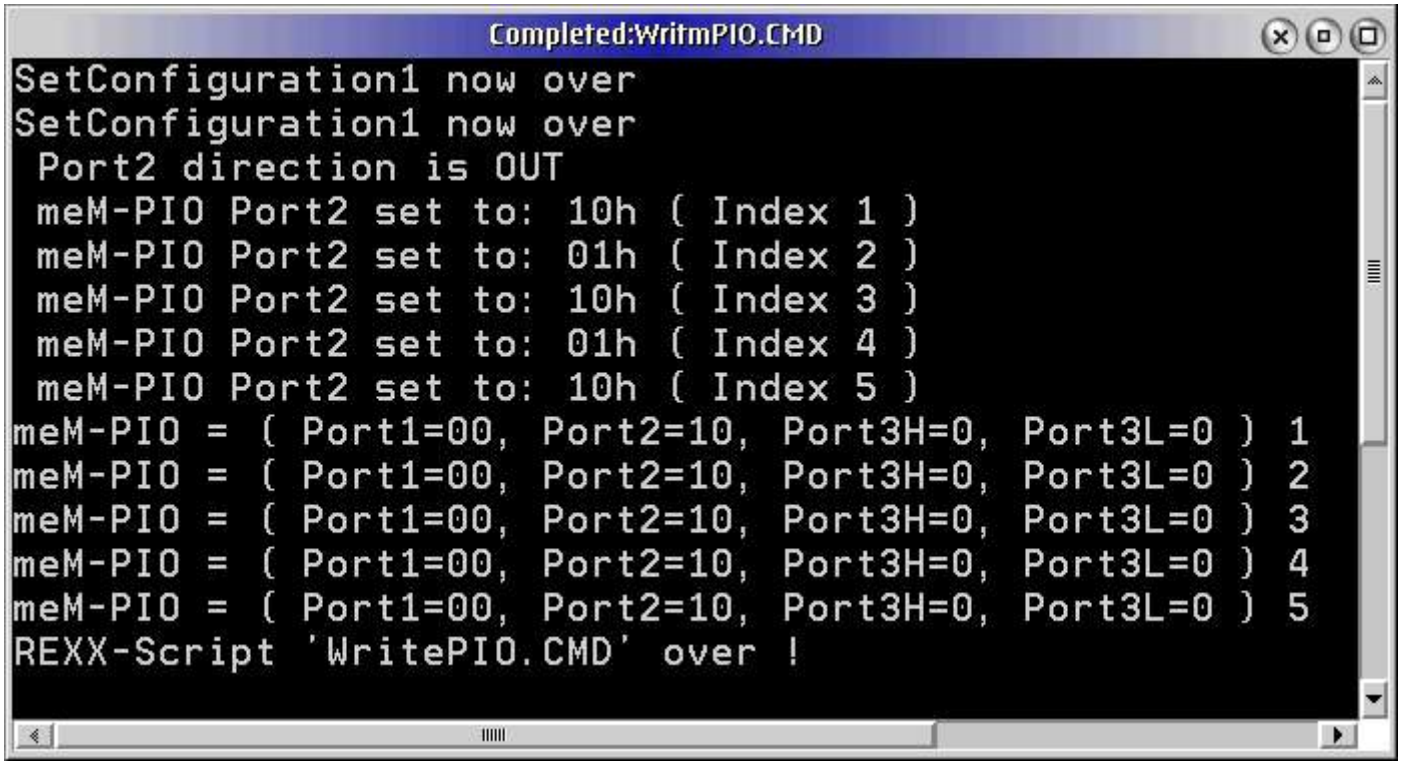
by Uwe Hinz, 2008

The REXX test application 'WritmPIO.cmd' for meM-PIO

Sending Setup Packets

Output via Port2

Reading all 3 ports



With a set of test plugs the reactions of meM-PIO can be examined.

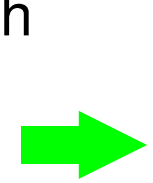
One test plug can show the correct output via **Port2** with **LED**s, another one can demonstrate the input via **Port1** with a **switch**, and a third test plug can loopback **Port3L** to **Port3H**.

by Uwe Hinz, 2008

From Phase 2 to Phase 3

Phase 2

- Test Applications with REXX created !



Phase 3

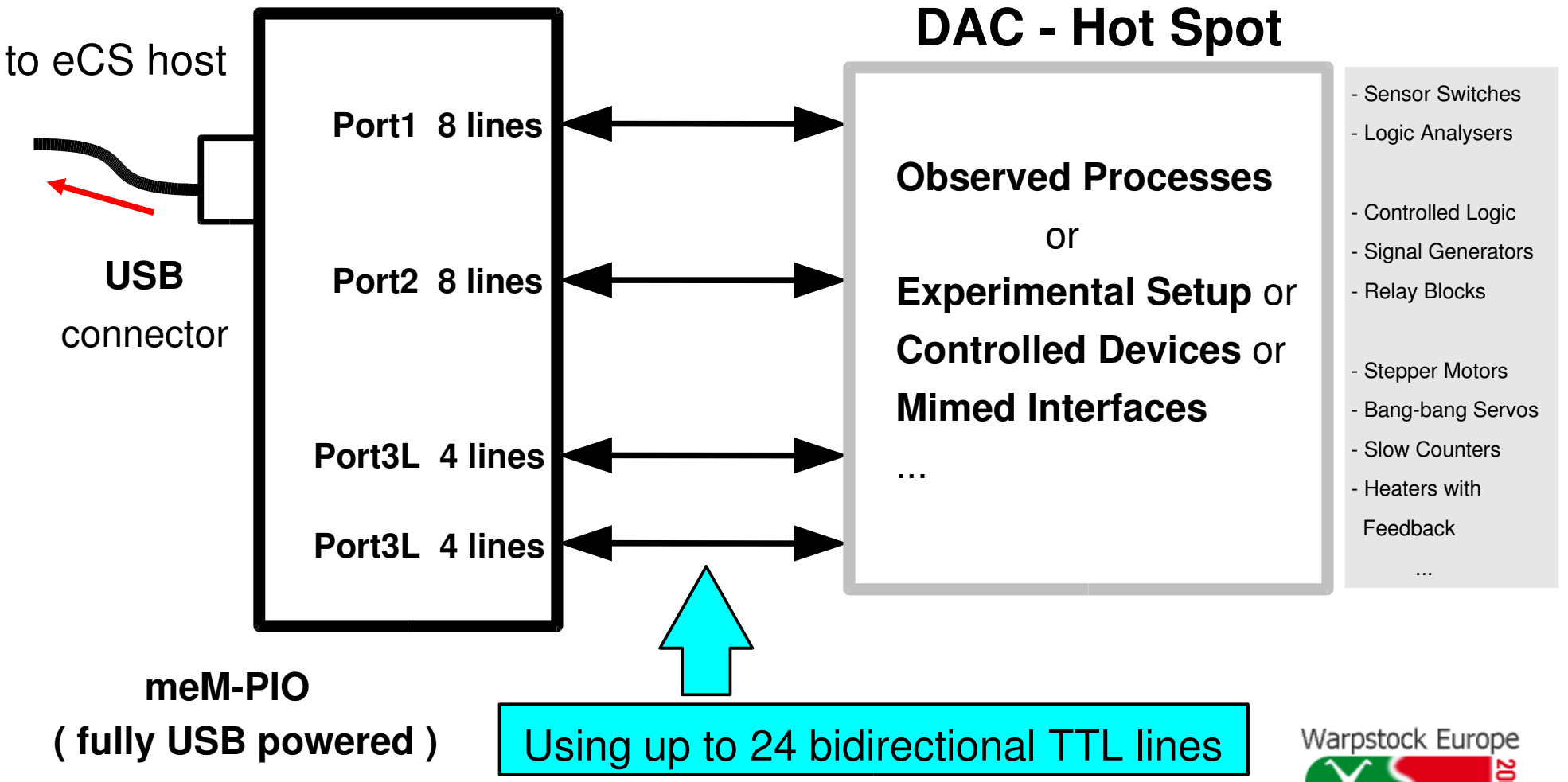
Turning the collected knowledge into a Device Control Application

' DTM20xxA.cmd '

Digital I/O with eComStation via the USB Interface Box meM-PIO

by Uwe Hinz, 2008

Hypothetic use of meM-PIO



by Uwe Hinz, 2008

DTM 2010 - The Device, that can be connected to meM-PIO

Warpstock Europe 2008
Nov. 14-16, Düsseldorf, Germany

This historic digital thermometer was manufactured before 1989 by ' VEB Thermometerwerk Geraberg '.

It is equiped with a reduced **IEEE 488 Interface** (8 Data Lines + 3 Handshake Lines, no Control Lines) [6] [8].

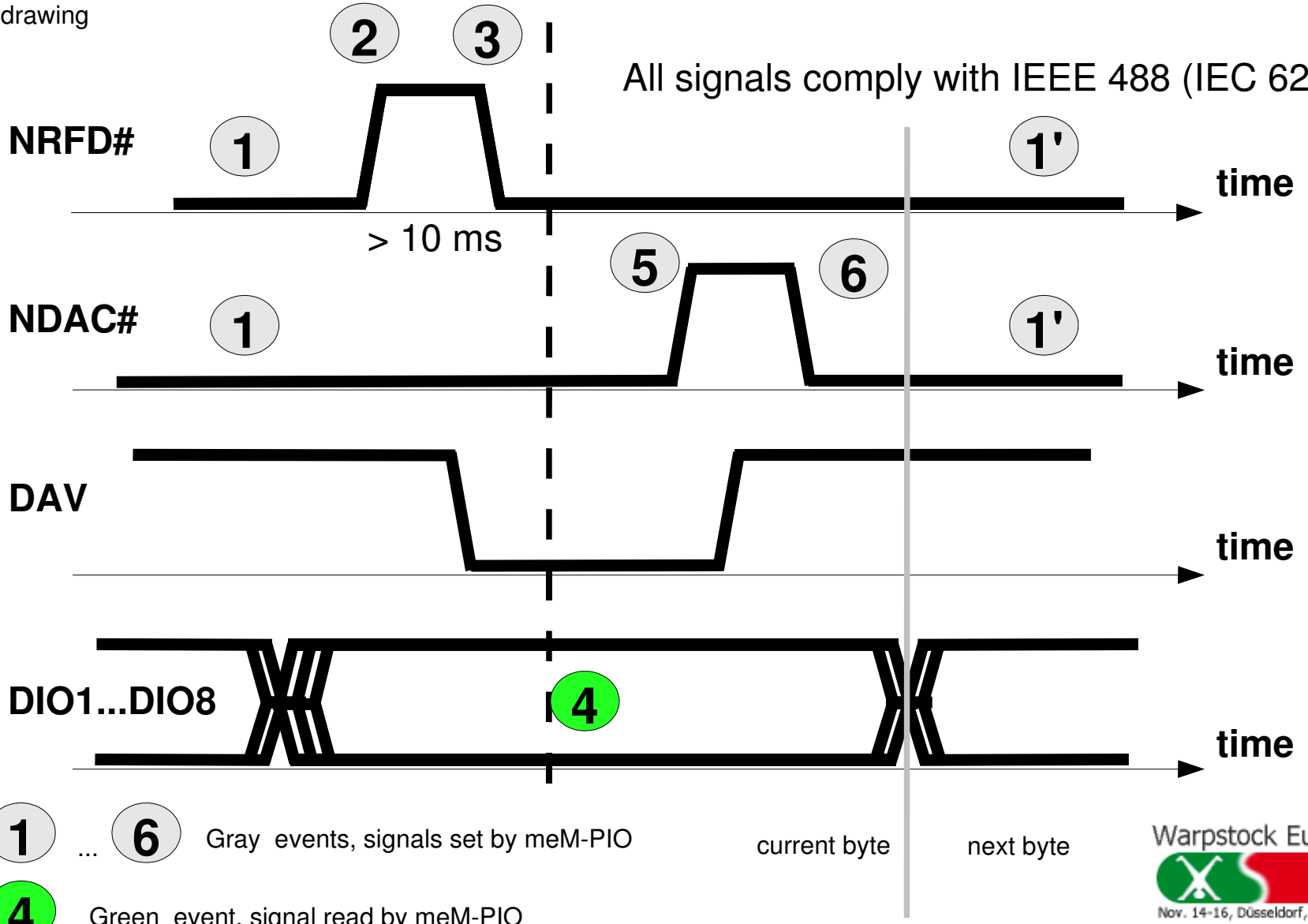
No extra electronics - just a self-made cable is needed in order to attache the device to **meM-PIO**.

Digital I/O with eComStation via the USB Interface Box meM-PIO

by Uwe Hinz, 2008

DTM 2010 signals

free scale drawing



Digital I/O with eComStation via the USB Interface Box meM-PIO

by Uwe Hinz, 2008

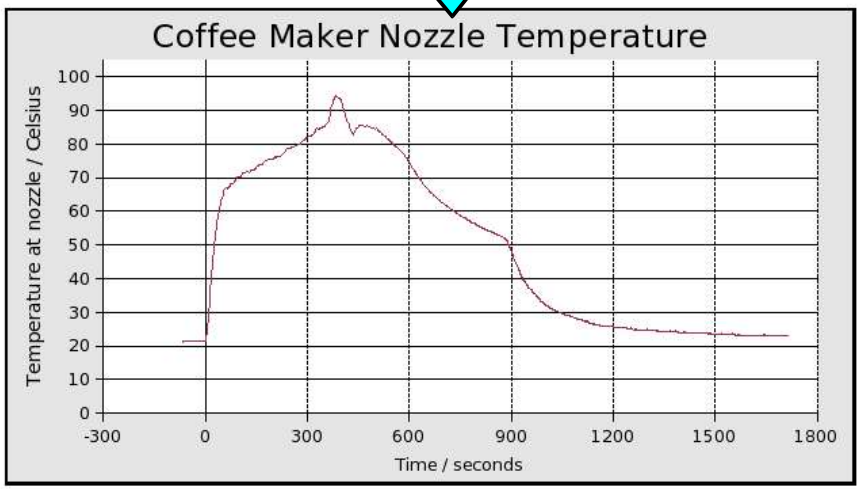
A Coffee Maker Example

Warpstock Europe 2008
Nov. 14-16, Düsseldorf, Germany

Thermometer ' DTM 2010 ' and
REXX script ' DTM20xxA.cmd '

Data file ' DTM0003.csv '

OpenOffice Calc file ' DTM0003.scx '



by Uwe Hinz, 2008

References 1

[1] http://home.hccnet.nl/w.m.brul/usbprobe/index.html

[2] http://www.bmc-messsysteme.de/ger/sitemap.html

[3] **Compaq, Intel, Microsoft, NEC** 'Universal Serial Bus Specification' Revision 1.1, September 23, 1998

[4] http://AutomationONSPEC.com/AOSbrochure.pdf

[5] http://automationonspec.com/

[6] http://www.techsoft.de/htbasic/tutgpibm.htm?tutgpib.htm/

References 2

[7] http://en.ecomstation.ru/projects/usbtools/download/usbcalls-20060807.zip , ftp://ftp.netlabs.org/pub/events/DWS2005/DWS2005_USBStack.pdf

[8] http://www.juengling-online.de/Data/DTM_2010.pdf

by Uwe Hinz, 2008

To Do List

- Speed measurement and extended tests
- Using a language different from REXX
- Doing the same project with USBCALLS
- Tests with more than one meM-PIO
- Adding a Serial Number check to bind an individual meM-PIO to a future application
- Extending error checks within the applications
- Getting a new meM-PIO and looking for the differences that occurred over time
- Connecting meM-PIO with a device different from thermometer DTM 2010
- Creating a project that is run by more than one person
- Turning Wim Brul's driver into a special meM-PIO driver
- and ...

Acknowledgement

I thank

All members of the OS/2 User Group Dresden,
Annemarie Koebel, my grand child, who took the DTM 2010 photos,

and

Heidi Fritzlar, my wife, for her help and understanding...