PicoCOM1

Device-Driver

Version 1.13 Date: 11.04.2011

F & S Elektronik Systeme GmbH
Untere Waldplätze 23
D-70569 Stuttgart

Table Of Contents

1 Device Driver 2
1.1 Driver for Digital I/O 2
1.2 Analog-IN Driver 10
1.3 Display Driver 14
1.4 NANDFMD-Driver 16
1.5 SERIAL Driver 17
1.6 Ethernet Driver 19
1.7 SD/MMC Driver 20
1.8 Audio Driver 20
1.9 I²C Driver 22
1.10 SPI Driver 23
1.11 CAN Driver 24

2 Modules and Utilities 28
2.1 NDCUCFG utility 28
2.2 Module NETUI 32
2.3 Extending the Search Path 33

3 Index 34

1 Device Driver

1.1 Driver for Digital I/O

PicoCOM1 has 43 programmable I/O lines. You have to use these driver to configure and access the I/O lines.

Installation of the driver is done by setting some registry values under the following registry key:

```
[HKLM\Drivers\BuiltIn\DIGITALIO]
```

Required settings:

Key	Value	Comment
“Prefix”	“DIO”	This required value specifies the driver’s device file name prefix. It is a three-character identifier, such as COM.
“Dll”	“DIGIO.dll”	Name of the DLL with the Driver
“Order”	Dword:97	This value specifies the load order for the driver. If two drivers have the same load order value, the drivers load in the order that they occur in the registry.
“Index”	Dword:1	This value specifies the device index, a value from 0 through 9.
“Ioctl”	Dword:4	Call post-initialisation function.
“Port”	Dword:n	0,1,2,3,4 or 5
“UseAsIOA“	Dword:n	1 = The corresponding pin

“UseAsIOB“		is used as general purpose I/O. One bit for each I/O pin.
“DataDirA” “DataDirB”	Dword:n	Data Direction. 0 = The corresponding pin is an input. 1 = The corresponding pin is an output. One bit for each I/O pin.
“DataInitA” “DataInitB”	Dword:n	Default value of the output pin after driver initialization.
“IRQCFG0”	Hex:00,00,00, 00,00,00	Interrupt configuration 0 0 = The corresponding pin is not configured to signal a raising edge. 1 = The corresponding pin is configured to signal a raising edge.
“IRQCFG1”	Hex:00,00,00, 00,00,00	Interrupt configuration 1 0 = The corresponding pin is not configured to signal a falling edge. 1 = The corresponding pin is configured to signal a falling edge.
“FriendlyName”	Digital I/O driver for Pico-COM1	

The driver is realised as a block device driver. The interface functions are `CreateFile()`, `ReadFile()`, `WriteFile()`, `SetFilePointer()` and `DeviceIoControl()`.

PORT 0								
BIT	7	6	5	4	3	2	1	0
PIN	24	23	18	17	16	15	14	13
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
UseAsIOA Bit	7	6	5	4	3	2	1	0
DataDirA Bit	7	6	5	4	3	2	1	0
DataInitA Bit	7	6	5	4	3	2	1	0

PORT 1								
BIT	7	6	5	4	3	2	1	0
PIN	39	38	37	36	35	34	33	32
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
UseAsIOA Bit	15	14	13	12	11	10	9	8
DataDirA Bit	15	14	13	12	11	10	9	8
DataInitA Bit	15	14	13	12	11	10	9	8

PORT 2								
BIT	7	6	5	4	3	2	1	0
PIN	48	47	46	45	44	43	41	40
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
UseAsIOA Bit	23	22	21	20	19	18	17	16
DataDirA Bit	23	22	21	20	19	18	17	16
DataInitA Bit	23	22	21	20	19	18	17	16

PORT 3								
BIT	7	6	5	4	3	2	1	0
PIN	60	59	58	57	56	55	50	49
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
UseAsIOA Bit	31	30	29	28	27	26	25	24
DataDirA Bit	31	30	29	28	27	26	25	24
DataInitA Bit	31	30	29	28	27	26	25	24

PORT 4								
BIT	7	6	5	4	3	2	1	0
PIN	70	69	68	67	66	65	64	63
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
UseAsIOB Bit	7	6	5	4	3	2	1	0
DataDirB Bit	7	6	5	4	3	2	1	0
DataInitB Bit	7	6	5	4	3	2	1	0

PORT 5								
BIT	7	6	5	4	3	2	1	0
PIN						76	75	74
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
UseAsIOB Bit	15	14	13	12	11	10	9	8
DataDirB Bit	15	14	13	12	11	10	9	8
DataInitB Bit	15	14	13	12	11	10	9	8

IRQCFG1	IRQCFG0	Comment
0	0	Interrupt disabled
0	1	Raising edge enabled
1	0	Falling edge enabled
1	1	Raising and falling edge enabled

IRQCFG0 and IRQCFG1:

Port1								Port0								Port
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	BitPos
39	38	37	36	35	34	33	32	24	23	18	17	16	15	14	13	IO-Pin

Port3								Port2								Port
31	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	BitPos
60	59	58	57	56	55	50	49	48	47	46	45	44	43	41	40	IO-Pin

Port5								Port4								Port
47	**46**	**45**	**44**	**43**	**42**	**41**	**40**	**39**	**38**	**37**	**36**	**35**	**34**	**33**	**32**	BitPos
					76	75	74	70	69	68	67	66	65	64	63	IO-Pin

Programming Example (native code)

1.) Open one digital port

```
HANDLE hDIO;
hDIO = CreateFile( _T("DIO1:"),
            GENERIC_WRITE, 0,
            NULL, OPEN_EXISTING,
            FILE_ATTRIBUTE_NORMAL, NULL );
if( hDIO == INVALID_HANDLE_VALUE )
{
```

```
      MessageBox( NULL, TEXT("WinMain():
                  CreateFile() failed"),
                  TEXT("Err! - DIO-Test"),
                  MB_OK | MB_ICONEXCLAMATION);
      return(FALSE);
}
```

2.) Write data to the port

```
unsigned char data = 0xAA;
DWORD dwBytesWrite = 1;
WriteFile( hDIO, &data, dwBytesWrite,
           &dwBytesWrite, NULL );
if( dwBytesWrite != 1 )
{
      // Error
}
```

3.) Change port

```
LONG lDistance = 1;
SetFilePointer( hDIO, lDistance, NULL,
                FILE_BEGIN);
```

4.) Using Interrupts

```
WAITIRQ cWaitIrq;
cWaitIrq.usBitPos  = 34;   //Port 4, Bit 2
cWaitIrq.dwTimeOut = 2000; //Wait for 2sec for
                           //Interrupt
/* If the corresponding pin is configured as
raising edge *xor* falling edge this value has
to be false. If the corresponding pin is
configured for both edges this value has to be
false to receive the falling edge and true to
receive the raising edge. */
```

```
cWaitIrq.bType = FALSE;
```

4.1) Request Interrupt

```
/* Request a sysintr */
if(! DeviceIoControl(hDIO,IOCTL_DIO_REQUEST_IRQ
                     &cWaitIrq.usBitPos,
                     sizeof(unsigned short),
                    NULL, 0,NULL, NULL))
{
      //Error. Can not request for interrupt.
}
```

4.2) Wait for Interrupt

```
/* Wait for a sysintr */
DWORD dwWaitRes = -1;   //Value that indicates
                        //if event or timeout
                        //occurred. That match
                        //the return value of
                        //WaitForSingleObject.
if(! DeviceIoControl( hDIO,IOCTL_DIO_WAIT_IRQ,
                      &cWaitIrq,
                       sizeof(WAITIRQ),
                      &dwWaitRes, sizeof(DWORD),
                      NULL, NULL ))
{
      //Error. Can not wait for interrupt.
}
```

4.3) Reset Interrupt

```
/* Call InterruptDone on a sysintr */
if(! DeviceIoControl(hDIO,IOCTL_DIO_INTDONE_IRQ,
                     &cWaitIrq.usBitPos,
                     sizeof(unsigned short),
                     NULL, 0, NULL, NULL ))
{
      //Error. Can not reset interrupt.
}
```

4.4) Release Interrupt

```
/* Release a sysintr */
if(! DeviceIoControl(hDIO,IOCTL_DIO_RELEASE_IRQ,
                     &cWaitIrq.usBitPos,
                     sizeof(unsigned short),
                     NULL, 0, NULL, NULL ))
{
      //Error. Can not release interrupt.
}
```

1.2 Analog-IN Driver

PicoCOM1 features 3 analog inputs. The selection of a channel can be done with the registry key *Channel* or dynamically with the `SetFilePointer()` function.
To access all channels separately using different file handles, one driver instance for each channel can be created in registry. Just copy the existing registry entry and adapt the *Index* and the *Channel* value.

Installation of the driver is done by setting some registry values under the following registry key:

`[HKLM\Drivers\BuiltIn\ANALOGIN]`

Required settings:

Key	Value	Comment
"Prefix"	“AIN“	This required value specifies the driver’s device file name prefix. It is a three-character identifier, such as COM.
“Dll“	“PC2_ANALOGIN.DLL“	Name of the DLL with the driver
“Order“	Dword:1	This value specifies the load order for the driver. If two drivers have the same load order value, the drivers load in the order that they occur in the registry.
“Index“	Dword:1	This value specifies the device index, a value from 0 through 9.
“Flags”	Dword:0	4: Disabled from loading

Key	Value	Comment
“Ioctl“	Dword:4	Call post-initialisation function.
“Channel“	Dword:n	Number of the analogue channel. See Table Channel. *Default: 0*
“Timeout”	Dword:50	Timeout waiting for a sample to be completed. *Default: 50*
“FriendlyName“	“ Analog input driver for PicoCOM1“	
“Debug”	Dword:0|4	Set to 4 to get list of registry settings at serial debug port. *Default: 0*

Table 1: Analog-IN registry settings.

The driver is realised as a block device driver. The interface functions are `CreateFile()` and `ReadFile()`. After opening the channel you can call `ReadFile()` to read one value from the port. The type of the pointer for `ReadFile()` must be of size WORD. To sample more than one value a buffer (array) of several WORDS can be passed to `ReadFile()`.

Programming example:

```
HANDLE hAIN;

/* open analog-in driver */
hAIN = CreateFileW(L"AIN1:",
GENERIC_READ|GENERIC_WRITE, 0, NULL,
OPEN_EXISTING, 0, NULL);
if (INVALID_HANDLE_VALUE != hAIN)
{
```

```
    WORD wValue = 0;
    DWORD dwBytesRead;
    BOOL bNoError = TRUE;

    for(int i=0; i<3 && bNoError; i++)
    {
        /* select channel */
        SetFilePointer(hAIN, i, NULL,
                       FILE_BEGIN);

        /* sample analog value 10 times */
        for(int n=10; n>0; n--)
        {
            if (ReadFile(hAIN, &wValue, 1,
                         &dwBytesRead, NULL))
            {
                RETAILMSG(1,
(L"AIN value ch%d: %d\r\n", i, wValue));
            }
            else
            {
                RETAILMSG(1,
(L"Reading from analog in failed (LE: %d)\r\n",
                  GetLastError()));
            }

           Sleep(2);
        } /* read loop */

    } /* channel loop */

    CloseHandle(hAIN);
}
else
{
```

```
    RETAILMSG(1,
        (L"Can not open 'AIN1:' (LE: %d)\r\n",
         GetLastError()));
}
```

1.3 Display Driver

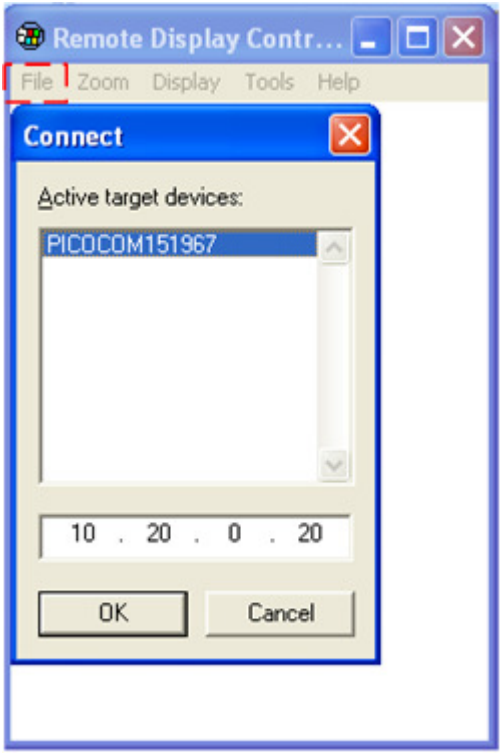
PicoCOM1 is naturally designed to come without any display. However, by default it has a remote display installed as the Windows CE GDI display driver. It is also a Windows CE LC-Display driver available to connect monochrome graphic displays which can be driven by a serial SPI connection.

The driver can be found in registry under:

```
[HKLM\System\GDI\Drivers]
```

To connect to the remote display of PicoCOM1 a host program on your development PC is needed. You can download the host program CERHOST.EXE from http://www.picocom.de.

Please make sure that you have configured the network interface of the PicoCOM1 and you are able to establish a connection to the PicoCOM1 from your development PC. Then start CERHOST.EXE and select 'connect' from File-Menu. You should get an output similar to the figure below.



Note: In some rare situations CERHOST can not display the target name. If you can not see the device in the 'Active target devices' section, please make a left-mouse-click into the left upper corner of the list box. If your PicoCOM1 sends the broadcast, you should see the Ip-Address of the target as result of your selection by the left-mouse-click.

1.4 NANDFMD-Driver

```
[HKLM\Drivers\BuiltIn\NANDFMD]
```

Required settings:

Key	Value	Comment
"Prefix"	“DSK“	This required value specifies the driv-er’s device file name prefix. It is a three-character identifier, such as COM.
“Dll“	“pc1_NandFlash.dll“	name of the DLL with the driver
“Order“	Dword:0	This value specifies the load order for the driver. If two drivers have the same load order value, the drivers load in the order that they occur in the registry.
“Index“	Dword:1	This value specifies the device index, a value from 0 through 9.
“FriendlyName“	"PicoCOM1 Nand Flash Driver“	
“Profile“	“FFSDISK“	Drive name

1.5 SERIAL Driver

This driver is needed to access the serial interfaces `COM1:`, `COM2:` and `COM3:`.

The registry keys for the driver are:

```
[HKLM\Drivers\BuiltIn\SERIAL1]
[HKLM\Drivers\BuiltIn\SERIAL2]
[HKLM\Drivers\BuiltIn\SERIAL3]
```

Optional settings:

Key	Value	Comment
"Priority256"	Dword:101	Priority for serial receive/transmit thread. Default: 101
"RS485"	Dword:1 Dword:0	Enable RS485 mode for COM1: Default: 1

RS485 Mode

On PicoCOM1 you can toggle COM1: between RS232 and RS485. To do this, you have to add the registry value RS485 and set it to 1.

Programming Example

1) Open one serial port

```
HANDLE hCOM = CreateFile( L”COM2:”,
                          GENERIC_WRITE|
                          GENERIC_READ,0,NULL,
OPEN_EXISTING,
                          FILE_ATTRIBUTE_NORMAL,
NULL );
if( hCOMA == INVALID_HANDLE_VALUE )
      /* Error handling */
```

2) Write to serial port

```
DWORD dwBytesWrite      = 0;
BYTE  byData            = 0xAA;

int res = WriteFile( hCOM, &byData,
                     1,&dwBytesWrite,
                     NULL );
if(res == 0 || dwBytesWrite != 1)
      /* Error handling */
```

3) Read from serial port

```
ReadFile(hCOM, byData, 1, &dwBytesWrite,
         NULL);
if(res == 0 || dwBytesWrite != 1)
      /* Error handling */
```

4) Closing one serial port

```
if(hCOM != INVALID_HANDLE_VALUE)
      CloseHandle(hCOM);
```

1.6 Ethernet Driver

The Ethernet-Interface on the PicoCOM1 features a small set of additional configurations:

Key	Value	Comment
“LEDConfig”	Dword:0…8	Specifies the use of the LED 0: Link OK 1: RX or TX Activity 2: TX Activity 3: RX Activity 4: Collision 5: 100 Base-TX mode 6: 10 Base-T mode 7: Full Duplex 8: Link OK / Blink on RX-TX Activity *Default: 8*
“TransmitGain”	Dword:0…3	Sets the transmit output amplitude 0: 0dB 1: 0.4dB 2: 0.8 dB 3: 1.2 dB *Default: 1*
“Speed”	Dword: 0 | 10 | 100	Link speed in Mbit/s *Default: 0 (disabled)*
“FullDuplex”	Dword: 0…1	Enable Full-Duplex mode *Default: 1*

Please note that it is required to define the “Speed” and the “FullDuplex” value to disable autonegotiation.

1.7 SD/MMC Driver

SD slot on PicoCOM1 is able to access SD and MMC storage cards. SDIO cards are not supported. Options and registry settings for SD driver are available in registry key:

```
[HKEY_LOCAL_MACHINE\Drivers\sdmem]
```

Registry settings:

Key	Value	Comment
"Clock"	Dword:5000000	Clock on the SD slot. Should be set to 5Mhz or 15Mhz.
"DeadTime"	Dword:1000	Polling interval for card detection.
"SingleBlockWrites"	Dword:0	This option allows to disable multiple block write command when set to 1. Multiple block write commands cause some problem with some SD cards.

1.8 Audio Driver

Audio driver for PicoCOM1 is implemented as wavdev2 driver and can be configured under the following registry key:

```
[HKEY_LOCAL_MACHINE\Drivers\BuiltIn\Audio]
```

Note: Due to **compatibility purposes** the **mixer interface** of audio driver **has changed in driver major release version 2** (V2.x) . Following table will describe the mixer values used in this new version. If you intend to used the "old" audio mixer, which still

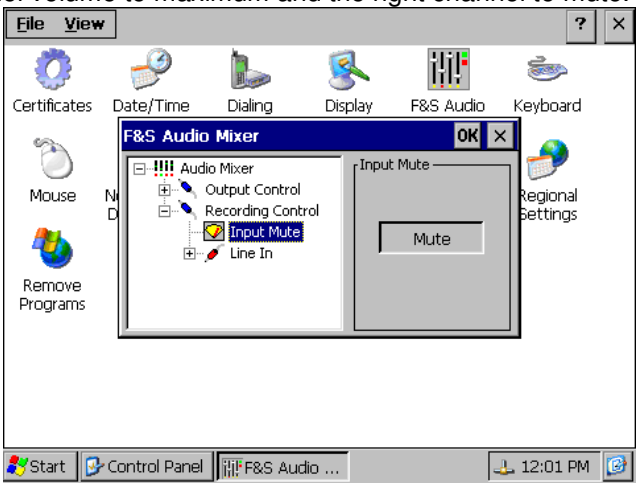
is available in all kernel images, please contact our support team to get detailed information.

Possible settings:

Key	Value	Comment
“Prefix”	“WAV”	This required value specifies the driver’s device file name prefix. It is a three-character identifier, such as COM.
“DLL”	“pc1_wavedev.dll”	name of the driver file
“Index”	Dword:1	This value specifies the device index, a value from 0 through 9.
“MasterOutMute”	Dword:0/1	Mute all audio output channels. *Default: 0*
“MasterOutVol”	Dword: 0 – 0xFFFFFFFF	Main volume for all Output-channels. *Default: 0xFFFFFFFF*
“BypassMute”	Dword:0/1	Mute Line-In bypass. *Default: 1*
“HeadphoneVol”	Dword: 0 – 0xFFFFFFFF	Volume for head-phone channel. *Default: 0xDFF2DFF2*
“MasterInVol”	Dword: 0 – 0xFFFFFFFF	Main volume for all Input-channels. *Default: 0x0*
“MasterInMute“	Dword:0/1	Mute Line-In *Default: 1*
“LineInVol “	Dword: 0 – 0xFFFFFFFF	Volume for Line-In channel. *Default: 0x0*

Additionally the audio-line can be configured using the F&S Audio Mixer utility, remaining in the control panel. Any mixer changes automatically adapt the registry settings. To store the current configuration permanently you just have to save the registry.

Remark: All volume settings separate into left and right channel value. The first 2 bytes of the 4 byte value are controlling the volume of the left channel. The second 2 bytes control the right channel volume. A value of 0xFFFF0000 for example sets the left channel volume to maximum and the right channel to mute.



1.9 I²C Driver

Note: Not included in the current kernel release !

1.10 SPI Driver

Note: Not included in the current kernel release !

1.11 CAN Driver

This chapter only describes the configuration of the driver. The usage of the driver including examples is described in the document “PicoCOM1_CanInterface_eng.pdf”.

All driver settings are defined under the following registry key:

```
[HKLM\Drivers\BuiltIn\CAN1]
```

Possible settings:

Key	Value	Comment
Prefix	“CID“	This required value specifies the driver’s device file name prefix. It is a three-character identifier, such as COM.
Dll	“CANDRV.DLL“	Name of the driver DLL.
Order	Dword:20	This value specifies the load order for the driver. If two drivers have the same load order, they use the order that they occur in the registry.
Index	Dword:1	This value specifies the index x in the device name CIDx: (x = 0..9).
Ioctl	Dword:4	Call post-initialisation function.
DeviceArrayIndex	Dword:0	Number of the hardware port you want to access. 0: Port at connector J9 (starterkit) ***Note:*** *This value should not be changed*

FriendlyName	“CAN driver for PicoCOM“	Description as shown in some info dialogs
UseTxIRQ	Dword: 1	Use send buffer 0: No send buffer; wait until transmission is possible when sending 1: Use send buffer; return immediately when sending and create “transmitted” event when actually done.
SendBufferSize	Dword:100	Number of messages in each send buffer (since V2.x)
EventQueueSize	Dword: 200	Number of possible event entries in each event queue
Debug	Dword: 0	Activate additional debug output ***Note:*** *This value usually does not need to be changed*
Priority256	Dword: 103	Priority for CAN service thread
Baudrate	Dword: 1000000	Default baudrate
CanMode2B	Dword: 0	Default CAN bus mode: 0: CAN2.0A (only standard frames) 1: CAN2.0B (standard and/or extended frames)
Format	Dword: 0	Default frame format: 0: depending on mode (standard in CAN2.0A, extended in CAN2.0B) 1: always standard 2. always extended

Virtualize	Dword: 0	Virtual CAN loop-back: 0: The local host never sees transmitted frames 1: The local host also sees and maybe accepts transmitted frames
AcceptanceCode	Dword: 0	Code value for default acceptance filter (since V2.x)
AcceptanceMask	Dword: 0	Mask value for default acceptance filter (since V2.x)
MaskActive	Dword: 0	Acceptance filter mask logic (see below, since V2.x)
Align	Dword: 0	ID and acceptance filter alignment (see below, since V2.x)
IRQ	Dword: 143	Default IRQ for CAN controller ***Note:*** *This value should not be changed*

MaskActive

The MaskActive entry defines which bits of the acceptance mask denote to require a match of the message ID bit with the acceptance code bit and which message ID bits are always accepted.

MaskActive setting	Acceptance Mask bit	ID bit of arriving message
0	0	Must match acceptance code bit
0	1	Always accepted
1	0	Always accepted

1	1	Must match acceptance code bit

Align

The Align entry tells how the CAN message IDs and acceptance filter masks/codes are aligned within the 32-bit DWORD value. Align=0 is the same setting as in the V1.x drivers.

In addition to the 11 standard ID bits, an acceptance filter for standard frames may also cover up to the first two data bytes of the message itself, which allows for easier implementation of high level protocols like DeviceNet or CanOpen. These data bytes will always be masked in bits 15..0 of the mask/code. An acceptance filter for extended frames can only cover the 29-bit extended ID and no additional data bytes.

Align	Standard-Frame-ID	Extended-Frame-ID	Standard-Frame-Filter	Extended-Frame-Filter
0	Bits 10..0	Bits 28..0	ID: Bits 31..21 Data: Bits 15..0	ID: Bits 31..3 no Data
1	Bits 10..0	Bits 10..0	ID: Bits 10..0 no Data	ID: Bits 10..0 no Data
2	Bits 28..18	Bits 28..0	ID: Bits 28..18 Data: Bits 15..0	ID: Bits 28..0 no Data
3	Bits 31..21	Bits 31..3	ID: Bits 31..21 Data: Bits 15..0	ID: Bits 31..3 no Data

2 Modules and Utilities

2.1 NDCUCFG utility

This utility is always included in the WindowsCE image and enables the customer to access the registry from the command line and to call some additional helper functions.

Ndcucfg.exe can be started over serial line, telnet or within a command window. By default, ndcucfg.exe is started from a Launch/Depend configuration in

```
[HKEY_LOCAL_MACHINE\Init]
```

and receives commands over serial line COM3:. If you want to change the serial line you can find settings of *ndcucfg.exe* under the following registry key:

```
[HKEY_LOCAL_MACHINE\System\NDCUCFG]
```

Possible settings:

Key	Value	Comment
"Port"	“COM3:”	NDCUFG is automatically started during boot because of a entry in HKLM\INIT. With this value you can specify on which serial line ndcucfg uses for communication.
"BatchFile“	String	The commands in the file will be exe-

		cuted during start of ndcucfg.exe.

List of commands (not complete):

- *display mode set <mode>*
  Changes the display mode to the given number.
- *display mode get*
  Retrieves the display mode.
- *display rotate get*
  Retrieves the display rotation angle.
- *display rotate set <n>*
  Changes the display rotation to the given angle.
- *reg open*
  opens the root key under HKLM
- *reg open <key>*
  opens the specified key under HKLM(open)
- *reg opencu <key>*
  opens the specified key under HKCU(opencu)
- *reg enum*
  displays a list of all keys and values under the current location
- *reg set value <name> dword <value>*
- *reg set value <name> string <value>*
- *reg set value <name> multi <value1>;<value2>*
- *reg set value <name> hex <value>,<value>,<value>*
  sets/creates the value with name <name> to the value <value>
- *reg create key <name>*
  Creates the specified sub-key and opens it.
- *reg del value <name>*
  Delete the specified value from registry.
- *reg del key <name>*
  Delete the specified key from registry.
- *reg save*
  Saves the registry in flash memory, so that modifications

are available after reset.
- *fat format <volume>*
  Formats the volume with name <volume>.
- *contrast +*
  Increase contrast voltage of LCD (small steps)
- *contrast ++*
  Increase contrast voltage of LCD (large steps)
- *contrast -*
  Decrease contrast voltage of LCD (small steps)
- *contrast --*
  Decrease contrast voltage of LCD (large steps)
- *contrast get*
  Returns the current contrast voltage of LCD.
- *contrast set <n>*
  Sets the contrast voltage of LCD. The value is the high time for the PWM circuit.
- *backlight on*
  Switch on backlight of LCD
- *backlight off*
  Switch off backlight of LCD
- *touch calibrate*
  Shows the calibration screen for the touch panel.
- *sip on*
  Shows the input panel window.
- *sip off*
  Hides the input panel window.
- *reboot*
  Reboots the device.
- *cert import cert <store> <file>*
  Import certificate with filename <file> into certificate store <store>. Values for <store> MY, CA or ROOT
- *cert import pkey <store> <file>*
  Import private key from file into certificate store MY, CA or ROOT
- *cert enum*
  List all certificates from store MY, CA and ROOT

- *cert delete <store> <store name>*
  Delete certificate
- *user create <name> <password>*
  Creates new use with password
- *user delete <name>*
  Delete user
- *user enum*
  List all users
- *REM <comment>*
  Records comments (remarks) in a batch file.
- *ECHO <message>*
  Displays messages.
- *start <file name> <parameter>*
  Creates a new process and its primary thread.
- *ndcucfg -B<file name>*
  runs <file name> as batch process.

2.2 Module NETUI

This module implements the user interface for the Network access. This module is used if a network resource is accessed which needs a user and password. By setting the described parameters, it is possible to avoid the normally shown dialog box.

The value can be found under key:

```
[HKLM\System\NETUI]
```

Parameter:

Key	Value	Comment
"AutoLogon"	Dword:0|1	Set this value to 1 to use the registry values UserName and Password for network access.
"UserName“	String	
"Password“	String	

Note: Using these option causes a security risk as the password will be stored in plain text.

2.3 Extending the Search Path

It's possible to extend the default path that the kernel uses to locate executable files. The necessary entry can be found under registry key:

```
HKEY_LOCAL_MACHINE\Loader
```

Possible settings:

Key	Value	Comment
"SystemPath"	Multi:"\\ffsdisk\\"	To extend the path you must add values to the value.

The SystemPath value has a maximum length of MAX_PATH characters, which includes the terminating NULL. Any path specified by the OEM is the last path to be when looking for a EXE. This registry value is only read during system boot.

3 Index

Driver
  Audio 20
  CAN 23
  Digital I/O 2
  Display 14
  Ethernet 19
  Flash file system 16
  SD/MMC 20
  SERIAL 17
  SPI 22
  UART 17
Extending the Search Path 32
Module NETUI 31
Utility
  NDCUCFG.EXE 27