TDRV002-SW-95

QNX6-Neutrino Device Driver

Multiple Channel Serial Interface

Version 1.0.x

User Manual

Issue 1.0.3

October 2010

TEWS TECHNOLOGIES GmbH
Am Bahnhof 7 25469 Halstenbek, Germany
Phone: +49 (0) 4101 4058 0 Fax: +49 (0) 4101 4058 19
e-mail: info@tews.com www.tews.com

TDRV002-SW-95

Multiple Channel Serial Interface

QNX6-Neutrino Treiber

Supported Modules:

- TPMC371
- TPMC372
- TPMC375
- TPMC376
- TPMC460
- TPMC461
- TPMC462
- TPMC463
- TPMC465
- TPMC466
- TPMC467
- TCP460
- TCP461
- TCP462
- TCP463
- TCP465
- TCP466
- TCP467

Issue	Description	Date
1.0.0	First Issue	September 27, 2005
1.0.1	TPMC467/TCP467 support added, modified file list	February 2, 2007
1.0.2	Modification for devc-devices and QNX6.3.x with SPx	February 23, 2007
1.0.3	Description of installation corrected, “Avoiding Data Loss” added, Address TEWS LLC removed	October 7, 2010

Table of Contents

1 Introduction

The TDRV002-SW-95 QNX6-Neutrino device driver is a full-duplex serial device driver which allows the operation of TDRV002 serial devices on Intel x86 based QNX6-Neutrino operating systems.

The TDRV002-SW-95 device driver is based on the standard QNX6 8250 serial communication manager. Due to this way of implementation the driver interface and function is compatible to the standard QNX6 serial device manager.

All standard utility programs for configuration (e.g. stty) and maintaining terminal interfaces could be used in the same manner.

TDRV002 will be used in the follow representative for all this modules and module names.

Supported Modules:

TPMC371	8 Channel Serial Interface
TPMC372	4 Channel Serial Interface
TPMC375	8 Channel Serial Interface (Programmable Interface)
TPMC376	4 Channel Serial Interface (Programmable Interface)
TPMC460	16 Channel Serial Interface
TPMC461	8 Channel Serial Interface
TPMC462	4 Channel Serial Interface
TPMC463	4 Channel Serial Interface
TPMC465	8 Channel Serial Interface (Programmable Interface)
TPMC466	4 Channel Serial Interface (Programmable Interface)
TPMC467	4 Channel Serial Interface (Programmable Interface)
TCP460	16 Channel Serial Interface
TCP461	8 Channel Serial Interface
TCP462	4 Channel Serial Interface
TCP463	4 Channel Serial Interface
TCP465	8 Channel Serial Interface (Programmable Interface)
TCP466	4 Channel Serial Interface (Programmable Interface)
TCP467	4 Channel Serial Interface (Programmable Interface)

Additional supported features:

- Receive and transmit FIFO trigger levels are configurable with driver start up
- Configuration of programmable interface with driver startup. (Only if hardware supports programmable interface).

2 Installation

The directory TDRV002-SW-95 on the distribution media contains the following files:

TDRV002-SW-95-SRC.tar.gz	Driver source archive
TDRV002-SW-95-1.0.3.pdf	This manual in PDF format
Release.txt	Release information
ChangeLog.txt	Release history

The TAR archive TDRV002-SW-95-SRC.tar.gz contains the following files and directories:

tdrv002/driver/nto/x86/o/Makefile
tdrv002/driver/nto/x86/Makefile
tdrv002/driver/nto/Makefile
tdrv002/driver/common.mk
tdrv002/driver/common.mk-nops
tdrv002/driver/common.mk-nopm
tdrv002/driver/externs.c
tdrv002/driver/externs.h
tdrv002/driver/init.c
tdrv002/driver/intr.c
tdrv002/driver/main.c
tdrv002/driver/Makefile
tdrv002/driver/options.c
tdrv002/driver/proto.h
tdrv002/driver/tedit.c
tdrv002/driver/tdrv002.h
tdrv002/driver/tto.c
tdrv002/driver/tdrv002config.txt
tdrv002/example/nto/x86/o/Makefile
tdrv002/example/nto/x86/Makefile
tdrv002/example/nto/Makefile
tdrv002/example/common.mk
tdrv002/example/example.c
tdrv002/example/Makefile

In order to perform the installation copy the tar-archive into the /usr/src directory and unpack it (`e.g. tar –xzvf TDR002-SW-95-SRC.tar.gz`). After that the necessary directory structure for the automatic build and the source files are available underneath the new directory called tdrv002.

Its absolute important to create the tdrv002 project directory in the /usr/src directory otherwise the automatic build with make will fail.

For building the device driver it is necessary that the QNX serial DDK is installed. (Installer: "/QNX Realtime Platform/Software Development/Device Driver Kits/ Character (Serial) DDK targeting x86").

For Serial DDKs not using the pm library, please use common.mk-nopm instead of common.mk build file. In detail, for QNX system releases before 6.3.0 copy common.mk-nopm to common.mk and start the build process.

For Serial DDKs not using the ps library, please use common.mk-nops instead of common.mk build file. In detail, for QNX system releases 6.3.x without a Service Pack copy common.mk-nops to common.mk and start the build process.

2.1 Build the device driver

1. Change to the /usr/src/tdrv002/driver directory
2. Execute the Makefile

```
# make install
```

After successful completion the driver binary will be installed in the */bin* directory.

2.2 Start the Driver Process

To start the TDRV002 device driver respective the TDRV002 serial communications manager you have to enter the process name with optional parameter from the command shell or in the startup script.

```
# devc-tdrv002 [options] &
```

OPTIONS

-b number	Initial baud rate (default 9600).
-C size	The size of the canonical buffer in bytes (default 1024).
-E	Start in raw mode (the default). Software flow control is disabled by default.
-e	Start in edit mode (default raw). Software flow control is enabled by default.
-F	Disable hardware flow control (default to hardware flow control enabled).
-f	Enable hardware flow control (default).
-I number	The size of the interrupt input buffer in bytes (default 8192).
-O number	The size of the interrupt output buffer in bytes (default 8192).
-S|*s*	Disable / enable software flow control. The default depends on the mode: in raw mode (*-E*, the default), its disabled; in edited mode (*-e*), it's enabled. The order in which you specify the *–E* or *–e*, and *–S* or *–s* options matters:

Options	**Mode**	**Software flow control**
-e	Edited	Enabled
-S –e	Edited	Enabled
-e –S	Edited	Disabled
-E	Raw	Disabled
-s –E	Raw	Disabled
-E –s	Raw	Enabled

-u number	Append number to the device name prefix (/dev/ser). The default is 3, which mean the first TDRV002 device is /dev/ser3; additional device are given increasing numbers.

-v	Print out debug information.
-L filename	Specifies the filename of the configuration file which defines the setup for FIFO trigger levels and programmable interfaces. This file is necessary for modules supporting progammable interfaces. (A detailed description of the configuration file can be found in the chapter *Configuration File*.)

Most of the options above are standard options for serial communications manager. Please refer also to related QNX6 documentation if necessary.

DESCRIPTION

The devc-tdrv002 manager is based on the standard QNX6 devc-ser8250 serial communications manager and can support any number of serial ports and TDRV002 modules.

The devc-tdrv002 manager searches the entire PCI-bus for TDRV002 devices and creates devices for each serial channel. The first device created depends on the **–u** option. If the **–u** option is omitted the first TDRV002 serial device is */dev/ser3*. If a TPMC461 (8 channel) and a TPMC462 (4 channel) are used, the devices */dev/ser3, /dev/ser4, …/dev/ser10* will be created for the TPMC461, */dev/ser11 … /dev/ser14* will be created for the TPMC462.

The order of device creation of the devices on different modules depends on the PCI deviceID. (Rising order: TPMC371, TPMC372, …, TPMC460, …, TCP460, …, TCP467)

Usually the device names /dev/ser1 and /dev/ser2 are assigned to the default PC serial ports, therefore the TDRV002 devices can start with /dev/ser3 (default). If there are additional onboard serial devices you have to start with a higher device number for the TDRV002 devices by defining an appropriate number with the **–u** option (please check also the */dev* directory).

A read request by default returns when at least 1 character is available. To increase efficiency, you can control three parameters to control when a read is satisfied:

Time	Return after a specified amount of time has elapsed (c_cc[VTIME]).
Min	Return when this number of characters is in the input buffer (c_cc[VMIN]).
Char	Return if the forwarding character is in the input buffer (c_cc[VEND]).

These parameters, and others, are set using library routines (see *tcgetattr(), txsetattr(), readcond()* and *TimerTimeout()* in the Library Reference).

The following fields and flags are supported in the *termios* structure.

Field	**Supported fields and flags**
c_cc	All characters
c_iflag	BRKINT ICRNL IGNBRK IXON
c_oflag	OPOST
c_cflag	CLOCAL CSIZE CSTOPB PARENB PARODD
c_lflag	ECHO ECHOE ECHOK ECHONL ICANON IEXTEN ISIG NOFLSH

EXAMPLES

Start the device driver with default parameters (first created device is */dev/ser3*, 9600 baud, see also options above…):

```
# devc-tdrv002 -F &
```

Start the device driver with default parameters and change baud rate to 38400

```
# devc-tdrv002 -F –b 38400 &
```

Start the device driver with default parameters. The first created device is */dev/ser5*.

```
# devc-tdrv002 -F –u 5 &
```

Start the device driver with default parameters and configuration information from ./tdrv002config.txt.

```
# devc-tdrv002 –L tdrv002config.txt &
```

2.3 Configuration File

This chapter describes the syntax used in the configuration file.

Each line starts with a prefix, a ‘*#*’ specifies a line with comment and a ‘*$*’ specifies a line with configuration data. Leading spaces will be ignored.

Behind the prefix ‘*#*’ all character will be ignored.

Behind the prefix ‘*$*’only valid characters are allowed. The line must have the following syntax:

```
$<mod>/<chan>-<P0><P1><P2><P3><P4><P5><P6><P7>-<Rx>/<Tx>
```

<mod>

Selects the module the configuration should be set to. *0* selects the 1st found module, *1* the 2nd and so on. A configuration that should be used for all modules can be specified with ‘*’.

<chan>

Selects the channel the configuration should be set to. *0* selects the 1st channel of the module, *1* the 2nd, and so on. A configuration that should be used for all channels of a specified module can be specified with ‘*’.

<P0>

This value specifies the setting of the *RS485/RS232#* configuration. A ‘*1*’ sets this bit and a ‘*0*’ resets the bit. (This value is ignored for non programmable interfaces)

<P1>

This value specifies the setting of the *HDPLX* configuration. A ‘*1*’ sets this bit and a ‘*0*’ resets the bit. (This value is ignored for non programmable interfaces)

<P2>

This value specifies the setting of the *RENA* configuration. A ‘*1*’ sets this bit and a ‘*0*’ resets the bit. (This value is ignored for non programmable interfaces)

<P3>

This value specifies the setting of the *RTERM* configuration. A ‘*1*’ sets this bit and a ‘*0*’ resets the bit. (This value is ignored for non programmable interfaces)

<P4>

This value specifies the setting of the *TTERM* configuration. A ‘*1*’ sets this bit and a ‘*0*’ resets the bit. (This value is ignored for non programmable interfaces)

<P5>

This value specifies the setting of the *SLEW LIMIT* configuration. A ‘*1*’ sets this bit and a ‘*0*’ resets the bit. (This value is ignored for non programmable interfaces)

<P6>

This value specifies the setting of the *SHDN* configuration. A ‘*1*’ sets this bit and a ‘*0*’ resets the bit. (This value is ignored for non programmable interfaces)

<P7>

This value specifies the setting of the *Auto RS485 Operation* configuration. A ‘*1*’ sets this bit and a ‘*0*’ resets the bit. (This value is ignored for non programmable interfaces)

<Rx>

Specifies the receive FIFO trigger level. The value must be between 1 and 63. (56 is a value that fits into most applications and systems)

<Tx>

Specifies the transmit FIFO trigger level. The value must be between 1 and 63. (8 is a value that fits into most applications and systems)

The configuration entries will always be scanned from the beginning of the file and the 1st matching configuration will be used. This allows the specification of general configurations and some special, that will be used for specifies channels.

Values for configuration parameters <P0>...<P7> are described in detail in the modules hardware user manual.

EXAMPLE

1) All channels shall get the same configuration:

```
# Setup all channel for RS232 with trigger levels of 56 for Rx and 8 for Tx
$*/*-00000000-56/8
```

2) Setup the first module different to the other modules

```
# Setup all channels of the 1st module for RS232, and all other channels for
# RS422 (RS485/RTERM) with trigger levels of 56 for Rx and 8 for Tx
$0/*-00000000-56/8
$*/*-10010000-56/8
```

3) Like 2) but 4th channel of the 1st module should also be RS422 (R485/RTERM)

```
$0/3-10010000-56/8
$0/*-00000000-56/8
$*/*-10010000-56/8
```

2.4 Avoiding Data Loss

If higher baud rates are used, or system load is high, it may be necessary to change the serial configurations. First FIFO trigger levels can be modified and second the size of the SW-buffers can be increased.

The receive trigger level specifies the number of characters that have to be in the FIFO to generate the interrupt. The remaining space in the FIFO specifies the time before a data overrun will occur and data gets lost. Therefore changing the configuration may be necessary if there is a high interrupt load on the system and the ISR may be delayed. The FIFO trigger level is defined in the configuration file. (See 2.3 Configuration File)

Example 1:

Configuration: receive trigger level is set to 56, 115200-8N1
8 Characters space in FIFO when interrupt occurs (FIFO-size [64] – trigger level [56])
➔ time until the FIFO must be read is at least ~0.69 ms ((8 * 10Bit) / 115200Baud)

Example 2:

Configuration: receive trigger level is set to 16, 115200-8N1
48 Characters space in FIFO when interrupt occurs (FIFO-size [64] – trigger level [16])
➔ time until the FIFO must be read is at least ~4.16 ms ((48 * 10Bit) / 115200Baud)

Changing the FIFO trigger level also changes the interrupt load. A decrease of the Rx FIFO trigger level will increase the number interrupt!

The second modification, changing sizes of SW-buffers is useful, if the interrupts can be handled in time, but there is still loss of data. The size of the SW-buffers can be specified when the driver is started. (See 2.2 Start the Driver Process)