The Embedded I/O Company

TEWS TECHNOLOGIES

TPMC550-SW-95

QNX6 - Neutrino Device Driver

8/4 Channel 12 Bit DAC PMC

User Manual

Issue 1.1 Version 1.0.0

November 2003

TEWS TECHNOLOGIES GmbH		**TEWS TECHNOLOGIES LLC**	
Am Bahnhof 7	25469 Halstenbek / Germany	1 E. Liberty Street, Sixth Floor	Reno, Nevada 89504 / USA
Phone: +49-(0)4101-4058-0	Fax: +49-(0)4101-4058-19	Phone: +1 (775) 686 6077	Fax: +1 (775) 686 6024
e-mail: info@tews.com	www.tews.com	e-mail: usasales@tews.com	www.tews.com

TPMC550-SW-95

8/4 Channel 12 Bit DAC PMC

QNX6-Neutrino Device Driver

Issue	Description	Date
1.0	First Issue	July 16, 2003
1.1	Introduction corrected	November 10, 2003

Table of Content

1 Introduction

The TPMC550-SW-95 QNX6-Neutrino device driver allows the operation of a TPMC550 - 8(4) Channel 12 Bit DAC PMC on QNX6-Neutrino operating systems with Intel or Intel-compatible x86 CPUs.

The TPMC550 device driver is basically implemented as a user installable Resource Manager. The standard file (I/O) functions (open, close and devctl) provide the basic interface for opening and closing a file descriptor and for performing device I/O and control operations.

To start an I/O request the client process has to send an appropriate message, which contains a function code and optional parameter, to the server process. After the I/O operation has finished the server process replies the request message with a completion status and optional process data to caller.

The TPMC550 device driver includes the following functions:

- writing a new DAC output value
- reading module configuration
- configuring, starting, stopping and using sequencer mode

2 Installation

The software is delivered on a PC formatted 3½" HD diskette, stored in a gzipped tar-archive named *TPMC550-SW-95.tar.gz*. This manual is located on the disk too, named *TPMC550-SW-95.pdf*.

Following files are stored in the tar-archive:

/driver/tpmc550.c	Driver source code
/driver/tpmc550.h	Definitions and data structures for driver and application
/driver/tpmc550def.h	Device driver include
/driver/node.c	Queue management source code
/driver/node.h	Queue management definitions
/example/example.c	Example application

For installation copy the tar-archive into the /usr/src directory and unpack it (e.g. `tar –xvzf TPMC550-SW-95.tar.gz`). After that the necessary directory structure for the automatic build and the source files are available underneath the new directory called *tpmc550*.

It is absolutely important to extract the TPMC550 tar archive in the /usr/src directory. Otherwise the automatic build with make will fail.

2.1 Build the device driver

Change to the /usr/src/tpmc550/driver directory

Execute the Makefile:

```
# make install
```

After successful completion the driver binary (tpmc550) will be installed in the /bin directory.

2.2 Build the example application

Change to the /usr/src/tpmc550/example directory

Execute the Makefile:

```
# make install
```

After successful completion the example binary (tp550exam) will be installed in the /bin directory.

2.3 Start the driver process

To start the TPMC550 device driver respective you have to enter the process name with optional parameter from the command shell or in the startup script.

```
tpmc550 [-v] &
```

The TPMC550 Resource Manager registers created devices in the Neutrinos pathname space under following names.

```
/dev/tpmc550_0
/dev/tpmc550_1
…
/dev/tpmc550_x
```

This pathname must be used in the application program to open a path to the desired TPMC550 device.

```
fd = open(“/dev/tpmc550_0”, O_RDWR);
```

For debugging you can start the TPMC550 Resource Manager with the –v option. Now the Resource Manager will print versatile information about TPMC550 configuration and command execution on the terminal window.

```
tpmc550 –v &
```

3 Device Input/Output functions

This chapter describes the interface to the device driver I/O system.

3.1 open()

NAME

open() - open a file descriptor

SYNOPSIS

#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

int open (const char *pathname, int flags)

DESCRIPTION

The *open* function creates and returns a new file descriptor for the TPMC550 named by pathname. The flags argument controls how the file is to be opened. TPMC550 devices must be opened O_RDWR.

EXAMPLE

```
int fd;

fd = open("/dev/tpmc550_0", O_RDWR);
```

RETURNS

The normal return value from open is a non-negative integer file descriptor. In the case of an error, a value of –1 is returned. The global variable errno contains the detailed error code.

ERRORS

Returns only Neutrino specific error codes, see Neutrino Library Reference.

SEE ALSO

Library Reference - open()

3.2 close()

NAME

close() – close a file descriptor

SYNOPSIS

#include <unistd.h>

int close (int filedes)

DESCRIPTION

The close function closes the file descriptor *filedes*.

EXAMPLE

```
int fd;

...

if (close(fd) != 0)
{
    /* handle close error conditions */
}
```

RETURNS

The normal return value from close is 0. In the case of an error, a value of –1 is returned. The global variable *errno* contains the detailed error code.

ERRORS

Returns only Neutrino specific error code, see Neutrino Library Reference.

SEE ALSO

Library Reference - close()

3.3 devctl()

NAME

devctl() – device control functions

SYNOPSIS

```
#include <sys/types.h>
#include <unistd.h>
#include <devctl.h>

int devctl
(
        int         filedes,
        int         dcmd,
        void        *data_ptr,
        size_t      n_bytes,
        int         *dev_info_ptr
)
```

DESCRIPTION

The devctl function sends a control code directly to a device, specified by *filedes*, causing the corresponding device to perform the requested operation.

The argument *dcmd* specifies the control code for the operation.

The arguments *data_ptr* and *n_bytes* depends on the command and will be described for each command in detail later in this chapter. Usually *data_ptr* points to a buffer that passes data between the user task and the driver and *n_bytes* defines the size of this buffer.

The argument *dev_info_ptr* is unused for the TPMC550 driver and should be set to NULL.

The following devctl command codes are defined in *tpmc550.h*:

Value	**Description**
DCMD_TPMC550_WRITE	Write a new DAC output value
DCMD_TPMC550_GET_CONFIG	Get the module configuration
DCMD_TPMC550_SEQ_START	Start sequencer mode
DCMD_TPMC550_SEQ_STOP	Stop sequencer mode
DCMD_TPMC550_SEQ_CONF	Configure channel for sequencer mode
DCMD_TPMC550_SEQ_FILL	Fill output data buffer for/in sequencer mode
DCMD_TPMC550_RESET	Perform a module reset

See behind for more detailed information on each control code.

To use these TPMC550 specific control codes the header file TPMC550.h must be included in the application.

RETURNS

On success, EOK is returned. In the case of an error, the appropriate error code is returned by the function (not in errno!).

ERRORS

Returns only Neutrino specific error code, see Neutrino Library Reference.

Other function dependent error codes will be described for each devoctl code separately. Note, the TPMC550 driver always returns standard QNX error codes.

SEE ALSO

Library Reference - devctl()

3.3.1 DCMD_TPMC550_WRITE

NAME

DCMD_TPMC550_WRITE – Write a new DAC output value

DESCRIPTION

This function writes a new DAC output value to a specified channel and returns immediately to the caller. For this function a *TPMC550_IN_WRITE_STRUCT* structure must be initialized with appropriate data.

```
typedef struct
{
        int                 channel;        // channel number
        long                value;          // new output value
        long                flags;          // output flags
} TPMC550_IN_WRITE_STRUCT;
```

channel

This argument specifies the channel number. Valid channel numbers are 1 up to 8 for TPMC550-10/-20 and 1 up to 4 for TPMC550-11/-21.

value

This argument specifies the new output value. The valid output value ranges depend on the module configuration. For channels configured as unipolar outputs (0V - 10V) the valid values are between 0 and +4095. For channels configured as bipolar outputs (+/-10V) the valid values are between -2048 and +2047. Values out of range will be set to the next valid value.

flags

This value is an ORed value of the following flags:

Value	**Description**
TP550_CORRECTION	If this flag is set, the output value will be corrected with the factory stored correction data.
TP550_LATCHED	If this flag is set, the output value will be written into the channels output register, the conversion will not be started until the *TP550_SIMCONV* flag is set. This flag will be used for simultaneous output of all channels.
TP550_SIMCONV	If this flag is set a simultaneous conversion of all channels will be initiated. The last written value will be converted.

EXAMPLE

```
int                           fd;
int                           result;
TPMC550_IN_WRITE_STRUCT       writeBuf;

...

/*
** write value to channel 1
*/
writeBuf.channel = 1;
writeBuf.value   = 0x042;
writeBuf.flags   = TP550_CORRECTION;

result = devctl(   fd,
                   DCMD_TPMC550_WRITE,
                   &writeBuf,
                   sizeof(writeBuf),
                   NULL);
if (result == EOK)
{
     /* write successful */
}

...
```

ERRORS

ECHRNG	Specified channel not supported by attached module.
EBUSY	The channel can not be used, because the module is configured in sequencer mode.
ETIME	The module does not complete the busy cycle.

SEE ALSO

Library Reference - devctl()

3.3.2 DCMD_TPMC550_GET_CONFIG

NAME

DCMD_TPMC550_GET_CONFIG – Get the module configuration

DESCRIPTION

This function reads the module configuration of the TPMC550 and returns immediately to the caller. This function uses a pointer to the TPMC550_OUT_CONFIG_STRUCT where the configuration values are filled in by the driver.

```
typedef struct
{
        int                     channels;       // number of valid channels
        signed char             offsetCorr[8];  // offset correction value
        signed char             gainCorr[8];    // gain correction value
        char                    bipolMode[8];   // bipolar mode enabled?
} TPMC550_OUT_CONFIG_STRUCT;
```

channels

This value returns the number of channels supported by the module. The value will be 8 for TPMC550-10/-20 and 4 for TPMC550-11/-21.

offsetCorr[]

This array returns the offset correction data for the channels. The correction of channel 1 will be returned with index 0, channel 2 with index 1 and so on. The data is only valid for existing channels.

gainCorr[]

This array returns the gain correction data for the channels. The correction of channel 1 will be returned with index 0, channel 2 with index 1 and so on. The data is only valid for existing channels.

bipolMode[]

This array returns if a channel configured for uni- or bipolar output. If bipolMode[*n*] is *TRUE*, the channel is configured in bipolar mode (+/-10V output). If bipolMode[*n*] is *FALSE*, the channel is configured in bipolar mode (0 - 10V output). The value for channel 1 will be returned with index 0, channel 2 with index 1 and so on. The values are only valid for existing channels.

EXAMPLE

```
int                            fd;
int                            result;
TPMC550_OUT_CONFIG_STRUCT      cfgBuf;

...

/*
** read module configuration
*/
encBuf.channel = 1;

result = devctl(   fd,
                   MD_TPMC550_GET_CONFIG,
                   &cfgBuf,
                   sizeof(cfgBuf),
                   NULL);
if (result == EOK)
{
     /* successful read */
}

...
```

ERRORS

EINVAL	Invalid argument. This error code is returned if the size of the message buffer is too small.

SEE ALSO

Library Reference - devctl()

3.3.3 DCMD_TPMC550_SEQ_START

NAME

DCMD_TPMC550_SEQ_START – Start sequencer mode

DESCRIPTION

This function sets up and starts the sequencer mode, the function returns immediately to the caller. For this function the structure *TPMC550_IN_SEQSTART_STRUCT* must be initialized with appropriate data. The channels must be configured for sequencer mode before this call is made.

```
typedef struct
{
        unsigned short          seqTime;        // Sequencer Time
        unsigned long           flags;          // TP550_SEQRUNAROUND | TP550_LATCHED
} TPMC550_IN_SEQSTART_STRUCT;
```

seqTime

This argument specifies the sequencer cycle time. This argument will not be used if the *TP550_SEQRUNAROUND* is set.

flags

This value is an ORed value of the following flags:

Value	**Description**
TP550_SEQRUNAROUND	If this flag is set, the sequencer starts in continuous mode. The conversion will be made as fast as possible.
TP550_LATCHED	If this flag is set, the sequencer will output all channels synchronously.

EXAMPLE

```
int                          fd;
int                          result;
TPMC550_IN_SEQSTART_STRUCT   seqStartBuf;

...

/*
** start sequencer mode
*/
seqStartBuf.seqTime = 50;
seqStartBuf.flags   = TP550_LATCHED;

result = devctl(   fd,
                   TPMC550_IN_SEQSTART_STRUCT,
                   &seqStartBuf,
                   sizeof(seqStartBuf),
                   NULL);
if (result == EOK)
{
     /* sequencer mode started */
}

...
```

ERRORS

EBUSY	The sequencer mode is already active.

SEE ALSO

Library Reference - devctl()

3.3.4 DCMD_TPMC550_SEQ_STOP

NAME

DCMD_TPMC550_SEQ_STOP – Stop sequencer mode

DESCRIPTION

This function stops the sequencer mode and returns to conventional mode. The function returns immediately to the caller.

EXAMPLE

```
int             fd;
int             result;

...

/*
** stop sequencer mode
*/

result = devctl(   fd,
                   DCMD_TPMC550_SEQ_STOP,
                   NULL,
                   0,
                   NULL);
if (result == EOK)
{
     /* sequencer successfully stopped */
}

...
```

ERRORS

ETIME	Sequencer-Stop command has been written to the specific register, but the sequencer refused to stop.

SEE ALSO

Library Reference - devctl()

3.3.5 DCMD_TPMC550_SEQ_CONF

NAME

DCMD_TPMC550_SEQ_CONF – Configure channel for sequencer mode

DESCRIPTION

This function sets up or disables a channel for sequencer mode, the function returns immediately to the caller. For this function the structure *TPMC550_IN_IN_SEQCONF_STRUCT* must be initialized with appropriate data. This function must be called before the sequencer mode is started, changes while the mode is active are not possible. For changes the channel must first be removed from the sequencer and then configured again.

```
typedef struct
{
        int                     channel;        // Channel number
        int                     bufSize;        // Size of Sequencer Data Buffer
        unsigned long           flags;          // TP550_CORRECTION | TP550_SEQONESHOT
        unsigned long           timeFactor;     // Specify number of ignored cycles before a new
                                                // value will be converted
} TPMC550_IN_SEQCONF_STRUCT;
```

channel

This argument specifies the channel number. Valid channel numbers are 1 up to 8 for TPMC550-10/-20 and 1 up to 4 for TPMC550-11/-21.

bufSize

This argument specifies the buffer size (in values) which shall be allocated for data buffering.

flags

This value is an ORed value of the following flags:

Value	**Description**
TP550_CORRECTION	If this flag is set, the output value will be corrected with the factory stored correction data.
TP550_SEQONESHOT	If this flag is set, the sequencer will use the buffer as a FIFO. Each data value will be converted once. If every value has been used once, the channel will hold the last value. If new data is written the buffer is used again, until there is no unused data. If this flag is unset, the sequencer will restart with the first value after reaching the end of the buffer (Always the complete buffer is used also if not the complete buffer is filled with data). Data can be changed by overwriting the buffer. The first value will be written after the written one, if the end of the buffer is reached, the first value will be overwritten.

TP550_SEQREMOVE	If this flag is set, the channel will be removed from the sequencer configuration. Only the channel argument is valid. If this flags is not set, the channel will be added to the sequencer configuration.

timeFactor

This argument specifies how many cycles shall be skipped before a new value will be used. For example, the sequencer cycle time is 1 second, but the channel shall update every 10 seconds, the argument must be set to 9.

```
timeFactor = (channelRefreshTime / sequencerCycleTime) - 1
```

EXAMPLE

```
int                          fd;
int                          result;
TPMC550_IN_SEQCONF_STRUCT    seqCfgBuf;

...

/*
** configure channel 1 for sequencer mode
*/
seqCfgBuf.channel    = 1;
seqCfgbuf.bufSize    = 10;
seqCfgBuf.flags      = TP550_CORRECTION | TP550_SEQONESHOT;
seqCfgBuf.timeFactor = 0;                     /* no skipped cycles */

result = devctl(   fd,
                   TPMC550_IN_SEQCONF_STRUCT,
                   &seqCfgBuf,
                   sizeof(seqCfgBuf),
                   NULL);
if (result == EOK)
{
     /* configuration successful */
}

...
```

ERRORS

EBUSY	The sequencer mode is active. No configuration possible.
ECHRNG	Specified channel not supported by attached module.
EACCES	Channel is already configured. It must be removed first.
EINVAL	Invalid argument. This error code is returned if the size of the message buffer is too small.
ENOMEM	Not enough memory to allocate buffer of specified size.

SEE ALSO

Library Reference - devctl()

3.3.6 DCMD_TPMC550_SEQ_FILL

NAME

DCMD_TPMC550_SEQ_FILL – Fill output data buffer for/in sequencer mode

DESCRIPTION

This function fills a buffer of a channel, the function returns immediately to the caller. For this function the structure *TPMC550_IN_SEQFILL_STRUCT* must be initialized with appropriate data. The channels must be configured for sequencer mode before this call is made. This function can be called before and while the sequencer mode is active. Every channel used in sequencer mode shall be filled once before the sequencer is started.

```
typedef struct
{
    int             channel;        // Channel number
    int             bufSize;        // Size of Data Buffer (in long words)
    long            buffer[TPMC550_MAXBUFSIZE]; // databuffer
} TPMC550_IN_SEQFILL_STRUCT;
```

channel

This argument specifies the channel number. Valid channel numbers are 1 up to 8 for TPMC550-10/-20 and 1 up to 4 for TPMC550-11/-21.

bufSize

Specifies the number of data values specified in buffer.

buffer[]

This output values stored in the array will be written into the sequencer buffer of the specified channel. This buffer must always be smaller or equal to the configured sequencer buffer. The maximum size of this array, can be adapted by changing the value of the define *TPMC550_MAXBUFSIZE* in “tpmc550.h”.

The driver and the example must be recompiled and restarted after changing *TPMC550_MAXBUFSIZE*.

EXAMPLE

```
int                         fd;
int                         result;
TPMC550_IN_SEQFILL_STRUCT   seqFillBuf;

...

/*
** fill sequencer buffer for channel 1
*/
seqFillBuf.channel   = 1;
seqFillBuf.bufSize   = 2;
seqFillBuf.buffer[0] = 0x042;
seqFillBuf.buffer[1] = 0xFFF;

result = devctl(   fd,
                   DCMD_TPMC550_SEQ_FILL,
                   &seqFillBuf,
                   sizeof(seqFillBuf),
                   NULL);
if (result == EOK)
{
     /* buffer successfully filled */
}

...
```

ERRORS

ECHRNG	Specified channel not supported by attached module.
ENOSPC	Sequence too big for allocated memory space.

SEE ALSO

Library Reference - devctl()

3.3.7 DCMD_TPMC550_RESET

NAME

DCMD_TPMC550_RESET – Perform a module reset

DESCRIPTION

This devctl function performs a reset of an attached TPMC550 module. The data and control registers are overwritten and the sequencer is stopped. No parameters have to be supplied.

EXAMPLE

```
int             fd;
int             result;

...

/*
** perform reset of TPMC550 module
*/

result = devctl(   fd,
                   DCMD_TPMC550_RESET,
                   NULL,
                   0,
                   NULL);
if (result == EOK)
{
     /* reset successful */
}

...
```

ERRORS

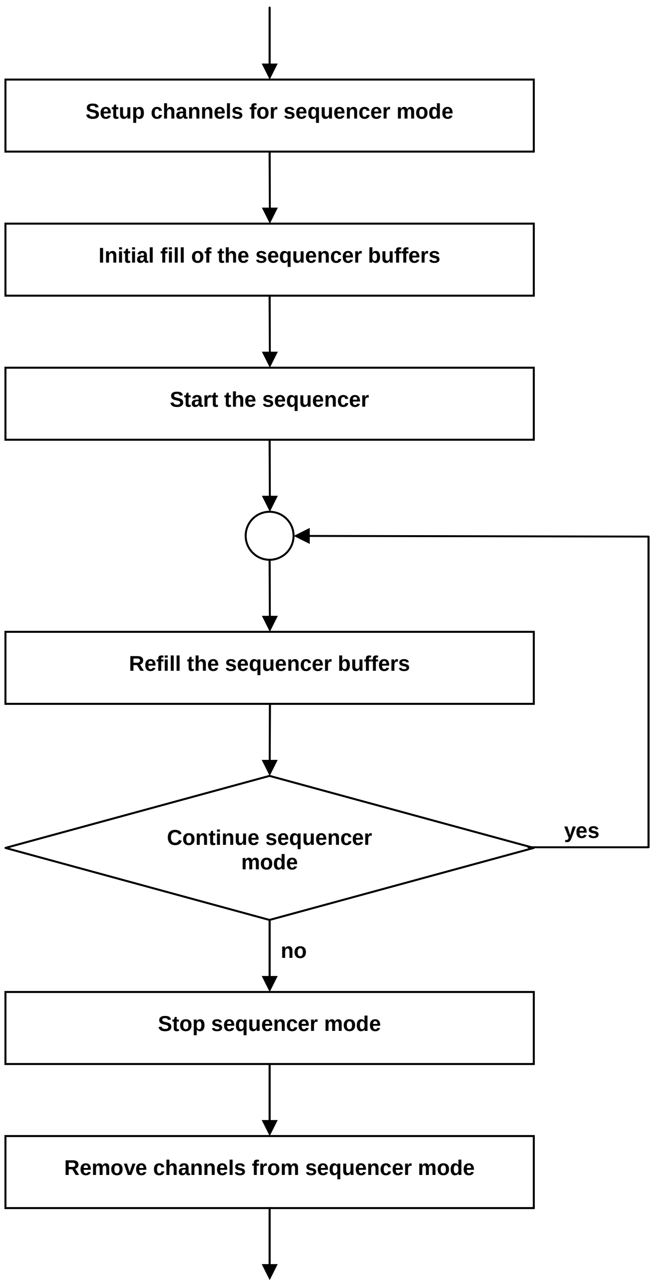
ETIME	A timeout happened during reset.

SEE ALSO

Library Reference - devctl()

4 Programming Hints

4.1 Using the sequencer mode


Setup channels for sequencer mode
Initial fill of the sequencer buffers
Start the sequencer
Refill the sequencer buffers
Continue sequencer mode
yes
no
Stop sequencer mode
Remove channels from sequencer mode