The Embedded I/O Company

TEWS TECHNOLOGIES

TPMC118-SW-95

QNX6-Neutrino Device Driver

6 Channel Motion Control

User Manual

Issue 1.0 Version 1.0.0

December 2002

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

TEWS TECHNOLOGIES LLC
1 E. Liberty Street, Sixth Floor
Reno, Nevada 89504 / USA
Phone: +1 (775) 686 6077
Fax: +1 (775) 686 6024
e-mail: usasales@tews.com
www.tews.com

TPMC118-SW-95

6 Channel Motion Control

QNX6-Neutrino Device Driver

Issue	Description	Date
1.0	First Issue	December 8, 2002

Table of Content

1 Introduction

The TPMC118-SW-95 QNX6-Neutrino device driver allows the operation of a TPMC118 6 channel motion control PMC on QNX6-Neutrino operating systems. The driver only supports the standard variant TPMC118-10.

The TPMC118 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.

Supported features:

- Reading value of encoder channel
- Reading synchronized channels
- Adding and removing channels from synchronous mode
- Configuring channel encoder resolution and reference/index mode
- Reading digital inputs
- Wait for event on digital input
- Setting analog output

For understanding features and modes, it is recommended to read the TPMC118 Users Manual.

2 Installation

The software is delivered on a PC formatted 3½" HD diskette.

Following driver specific files are located on the diskette:

/driver/tpmc118.c	Driver source code
/driver/tpmc118.h	Driver interface definitions and data structures
/driver/tpmc118def.h	Device driver include
/driver/node.h	Queue management definitions
/driver/node.c	Queue management source code
/example/example.c	Example application
TPMC118-SW-95.pdf	This Manual in PDF format

For installation create a new directory (e.g. *.../tpmc118*) in the */usr/src* directory and copy the complete */driver* and */example* directories (with sub-directories and all files) from the distribution diskette into the new created project directory.

It is absolute important to create the tpmc118 project directory in the /usr/src directory otherwise the automatic build with make will fail.

2.1 Build the device driver

1. Change to the */usr/src/tpmc118/driver* directory
2. Execute the Makefile

```
# make install
```

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

Build the example application

3. Change to the */usr/src/tpcm118/example* directory
4. Execute the Makefile

```
# make install
```

After successful completion the example binary (tp118exam) will be installed in the */bin* directory.

2.2 Start the driver process

To start the TPMC18 device driver respective the TPMC118 resource manager you have to enter the process name with optional parameters from the command shell or in the startup script.

```
tpmc118 &
```

This will start the TPMC118 resource manager with default configuration that means the driver start scanning the PCI-bus for TPCM118 modules and creates devices for each TPMC118.

The TPMC118 Resource Manager registers created devices in the Neutrinos pathname space under following names.

```
/dev/tpmc118_0
/dev/tpmc118_1
...
/dev/tpcm118_x
```

This pathname must be used in the application program to open a path to the desired TPMC118 device.

```
fd = open("/dev/tpmc118_0", O_RDWR);
```

Start the TPMC118 Resource Manager with the –v option for debugging. Now the Resource Manager will print versatile information about TPMC118 configuration and command execution on the terminal window.

```
tpmc118 –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 TPMC118 named by pathname. The flags argument controls how the file is to be opened. TPMC118 devices must be opened O_RDWR.

EXAMPLE

```
int fd;

fd = open(“/dev/tpmc118_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 TPMC118 driver and should be set to NULL.

The following devctl command codes are defined in *tpmc118.h*:

Value	Description
DCMD_TP118_READENC	Read value from specified encoder channel
DCMD_TP118_READINP	Read value from digital inputs
DCMD_TP118_SETDAC	Set new analog output value of specified channel
DCMD_TP118_SETRES	Set encoder resolution of specified channel
DCMD_TP118_SETPRELD	Set new encoder preload value
DCMD_TP118_SETMODE	Set encoder reference/index mode
DCMD_TP118_SYNCON	Add the specified encoder channel to synchronous mode
DCMD_TP118_SYNCOFF	Remove the specified encoder channel from the synchronous mode
DCMD_TP118_SYNCREAD	Read the synchronized encoder channels
DCMD_TP118_SETINTR	Wait for a specified event on digital inputs

See behind for more detailed information on each control code.

To use these TPMC118 specific control codes the header file TPMC118.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 TPMC118 driver always returns standard QNX error codes.

SEE ALSO

Library Reference - devctl()

3.3.1 DCMD_TP118_READENC

NAME

DCMD_TP118_READENC – Read encoder value

DESCRIPTION

This devctl function reads the value of a specified encoder channel. A pointer to the callers message buffer (*TP118_READENC*) and the size of this structure is passed by the parameters *data_ptr* and *n_bytes* to the device.

The *TP118_READENC* structure has the following layout:

```
typedef struct
{
        int                     channel;
        long                    value;
} TP118_READENC, *PTP118_READENC;
```

channel

This argument specifies the encoder channel that shall be read. Valid values are 1 up to 6.

value

This parameter will return the actual value of the encoder. Before reading encoder values the channels shall be configured.

EXAMPLE

```
int           fd;
int           result;
TP118_READENC encBuf;

...

/*
** read encoder value of channel 1
*/
encBuf.channel = 1;

result = devctl(   fd,
                   DCMD_TP118_READENC,
                   &encBuf,
                   sizeof(encBuf),
                   NULL);
if (result == EOK)
{
     /* successful read */
}

...
```

ERRORS

EINVAL	Invalid argument. This error code is returned if either the size of the message buffer is too small, or the specified parameter is out of range.

SEE ALSO

Library Reference - devctl()

3.3.2 DCMD_TP118_READINP

NAME

DCMD_TP118_READINP – Read digital input values

DESCRIPTION

This devctl function reads the values of the digital input channels. A pointer to the callers message buffer (*TP118_READINP*) and the size of this structure is passed by the parameters *data_ptr* and *n_bytes* to the device.

The *TP118_READINP* structure has the following layout:

```
typedef struct
{
        long                    value;
} TP118_READINP, *PTP118_READINP;
```

value

This parameter will return the actual value of the digital input register. Only bit 0 up to 5 are valid. Bit 0 is advised to IN1, bit 1 is advised IN2 and so on.

EXAMPLE

```
int             fd;
int             result;
TP118_READINP   digBuf;

...

/*
** read digital input value
*/
result = devctl(    fd,
                    DCMD_TP118_READINP,
                    &digBuf,
                    sizeof(digBuf),
                    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_TP118_SETDAC

NAME

DCMD_TP118_SETDAC – Set new DAC output

DESCRIPTION

This devctl function sets a new value to specified analog output channel. A pointer to the callers message buffer (*TP118_SETDAC*) and the size of this structure is passed by the parameters *data_ptr* and *n_bytes* to the device.

The *TP118_SETDAC* structure has the following layout:

```
typedef struct
{
    int             channel;
    long            value;
} TP118_SETDAC, *PTP118_SETDAC;
```

channel

This argument specifies the analog output channel that shall be written. Valid values are 1 up to 6.

value

This parameter specifies the new analog output value. Valid values are between -32768 (-10V) and 32767 (~+10V).

EXAMPLE

```
int           fd;
int           result;
TP118_SETDAC  dacBuf;

...

/*
** set analog output of channel 4 to 0x4000 (+5V)
*/
dacBuf.channel= 4;
dacBuf.value  = 0x4000;

result = devctl(   fd,
                   DCMD_TP118_SETDAC,
                   &dacBuf,
                   sizeof(dacBuf),
                   NULL);
if (result == EOK)
{
     /* successful set */
}

...
```

ERRORS

EINVAL	Invalid argument. This error code is returned if either the size of the message buffer is too small, or the specified parameter is out of range.

SEE ALSO

Library Reference - devctl()

3.3.4 DCMD_TP118_SETRES

NAME

DCMD_TP118_SETRES – Set new encoder channel resolution

DESCRIPTION

This devctl function changes the resolution of the specified encoder channel. A pointer to the callers message buffer (*TP118_SETMODE*) and the size of this structure is passed by the parameters *data_ptr* and *n_bytes* to the device.

The *TP118_SETMODE* structure has the following layout:

```
typedef struct
{
    int             channel;
    unsigned char   mode;
} TP118_SETMODE, *PTP118_SETMODE;
```

channel

This argument specifies the encoder channel which configuration shall be changed. Valid values are 1 up to 6.

mode

This parameter specifies the new encoder resolution. Valid values are:

Value	Description
TP118_ENCCNT_OFF	The encoder channel is disabled
TP118_ENCCNT_X1	The encoder counts in simple resolution
TP118_ENCCNT_X2	The encoder counts in double resolution
TP118_ENCCNT_X4	The encoder counts in quadruple resolution

EXAMPLE

```
int           fd;
int           result;
TP118_SETMODE modeBuf;

...

/*
** set up channel 2 in quadruple mode
*/
modeBuf.channel    = 2;
modeBuf.mode       = TP118_ENCCNT_X4;

result = devctl(   fd,
                   DCMD_TP118_SETRES,
                   &modeBuf,
                   sizeof(modeBuf),
                   NULL);
if (result == EOK)
{
     /* successful set */
}

...
```

ERRORS

EINVAL	Invalid argument. This error code is returned if either the size of the message buffer is too small, or the specified parameter is out of range.

SEE ALSO

Library Reference - devctl()

3.3.5 DCMD_TP118_SETPRELD

NAME

DCMD_TP118_SETPRELD – Set new encoder preload value

DESCRIPTION

This devctl function changes the preload value of the specified encoder channel. A pointer to the callers message buffer (*TP118_SETPRLD*) and the size of this structure is passed by the parameters *data_ptr* and *n_bytes* to the device.

The *TP118_SETPRLD* structure has the following layout:

```
typedef struct
{
    int             channel;
    unsigned long   value;
    unsigned char   flags;
} TP118_SETPRLD, *PTP118_SETPRLD;
```

channel

This argument specifies the encoder channel which configuration shall be changed. Valid values are 1 up to 6.

value

This parameter specifies the new preload value.

flags

This parameter specifies if the preload shall be executed immediately (only non referenced mode). Valid values are:

Value	Description
TP118_FL_IMMPRLD	The preload will be executed immediately if allowed.
TP118_FL_NORMPRLD	The value will be loaded into the preload register.

EXAMPLE

```
int           fd;
int           result;
TP118_SETPRLD preldBuf;

...

/*
** set new preload value (0x12345678) for channel 5 immediately
*/
preldBuf.channel   = 5;
preldBuf.value     = 0x12345678;
preldBuf.flags     = TP118_FL_IMMPRLD;

result = devctl(   fd,
                   DCMD_TP118_SETPRELD,
                   &preldBuf,
                   sizeof(preldBuf),
                   NULL);
if (result == EOK)
{
     /* successful set */
}

...
```

ERRORS

EINVAL	Invalid argument. This error code is returned if either the size of the message buffer is too small, or the specified parameter is out of range.

SEE ALSO

Library Reference - devctl()

3.3.6 DCMD_TP118_SETMODE

NAME

DCMD_TP118_SETMODE – Set new encoder channel reference/index mode

DESCRIPTION

This devctl function changes the reference/index mode of the specified encoder channel. A pointer to the callers message buffer (*TP118_SETMODE*) and the size of this structure is passed by the parameters *data_ptr* and *n_bytes* to the device.

The *TP118_SETMODE* structure has the following layout:

```
typedef struct
{
    int             channel;
    unsigned char   mode;
} TP118_SETMODE, *PTP118_SETMODE;
```

channel

This argument specifies the encoder channel which configuration shall be changed. Valid values are 1 up to 6.

mode

This parameter specifies the new encoder resolution. Valid values are:

Value	Description
TP118_ENC_NONREF	The encoder will work in non reference mode
TP118_ENC_REF	The encoder will work in reference mode
TP118_ENC_AUTOREF	The encoder will work in auto reference mode
TP118_ENC_INDEX	The encoder will work in index mode

EXAMPLE

```
int           fd;
int           result;
TP118_SETMODE modeBuf;

...

/*
** set up channel 2 in index mode
*/
modeBuf.channel    = 2;
modeBuf.mode       = TP118_ENC_INDEX;

result = devctl(   fd,
                   DCMD_TP118_SETMODE,
                   &modeBuf,
                   sizeof(modeBuf),
                   NULL);
if (result == EOK)
{
     /* successful set */
}

...
```

ERRORS

EINVAL	Invalid argument. This error code is returned if either the size of the message buffer is too small, or the specified parameter is out of range.

SEE ALSO

Library Reference - devctl()

3.3.7 DCMD_TP118_SYNCON

NAME

DCMD_TP118_SYNCON – Add the specified channel to synchronous channel set

DESCRIPTION

This devctl function adds the specified encoder channel to the synchronized channel set. A pointer to the callers message buffer (*TP118_SYNC*) and the size of this structure is passed by the parameters *data_ptr* and *n_bytes* to the device.

The *TP118_SYNC* structure has the following layout:

```
typedef struct
{
    int                 channel;
} TP118_SYNC, *PTP118_SYNC;
```

channel

This argument specifies the encoder channel that shall be used in synchronized mode. Valid values are 1 up to 6.

EXAMPLE

```
int           fd;
int           result;
TP118_SYNC    syncBuf;

...

/*
** add channel 3 to synchronized mode
*/
syncBuf.channel    = 3;

result = devctl(   fd,
                   DCMD_TP118_SYNCON,
                   &syncBuf,
                   sizeof(syncBuf),
                   NULL);
if (result == EOK)
{
     /* successful set */
}

...
```

ERRORS

EINVAL	Invalid argument. This error code is returned if either the size of the message buffer is too small, or the specified parameter is out of range.

SEE ALSO

Library Reference - devctl()

3.3.8 DCMD_TP118_SYNCOFF

NAME

DCMD_TP118_SYNCOFF – Remove the specified channel from synchronous channel set

DESCRIPTION

This devctl function removes the specified encoder channel from the synchronized channel set. A pointer to the callers message buffer (*TP118_SYNC*) and the size of this structure is passed by the parameters *data_ptr* and *n_bytes* to the device.

The *TP118_SYNC* structure has the following layout:

```
typedef struct
{
      int                     channel;
} TP118_SYNC, *PTP118_SYNC;
```

channel

This argument specifies the encoder channel that shall be removed from synchronized mode. Valid values are 1 up to 6.

EXAMPLE

```
int           fd;
int           result;
TP118_SYNC    syncBuf;

...

/*
** remove channel 3 from synchronized mode
*/
syncBuf.channel    = 3;

result = devctl(   fd,
                   DCMD_TP118_SYNCOFF,
                   &syncBuf,
                   sizeof(syncBuf),
                   NULL);
if (result == EOK)
{
     /* successful removed */
}

...
```

ERRORS

EINVAL	Invalid argument. This error code is returned if either the size of the message buffer is too small, or the specified parameter is out of range.

SEE ALSO

Library Reference - devctl()

3.3.9 DCMD_TP118_SYNCREAD

NAME

DCMD_TP118_SYNCREAD – Read all synchronized encoder channels

DESCRIPTION

This devctl function reads all synchronized encoder channels. A pointer to the callers message buffer (*TP118_SYNCREAD*) and the size of this structure is passed by the parameters *data_ptr* and *n_bytes* to the device.

The *TP118_SYNCREAD* structure has the following layout:

```
typedef struct
{
        long                    value[TP118_MAX_CHANS];
        unsigned char           valid[TP118_MAX_CHANS];
} TP118_SYNCREAD, *PTP118_SYNCREAD;
```

value[]

This array returns the encoder values of the synchronized channels, all channels that are not set to synchronized mode will be returned as 0. The array index specifies the assigned channel. Index 0 specifies channel 1, index 1 specifies channel 2 and so on.

valid[]

This array returns flags for each channel if it is set to synchronous mode or not. If value is set to *TRUE*, the channel is synchronized and the value is valid. If *FALSE* is returned, the channel is not synchronized and 0 will be returned for this channel. The array index specifies the assigned channel. Index 0 specifies channel 1, index 1 specifies channel 2 and so on.

EXAMPLE

```
int                 fd;
int                 result;
TP118_SYNCREAD      syncRdBuf;
Int                 i;

...

/*
** read synchronize channels
*/
result = devctl(    fd,
                    DCMD_TP118_SYNCREAD,
                    &syncRdBuf,
                    sizeof(syncRdBuf),
                    NULL);
if (result == EOK)
{
     /* successful set */
     for (i = 0; i < TP118_MAX_CHANS; i++)
     {
          if (syncRdBuf.valid[i])
          {
               printf("CH%d: %ld\n", i + 1, syncRdBuf.value[i];
          }
     }
}

...
```

ERRORS

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

SEE ALSO

Library Reference - devctl()

3.3.10 DCMD_TP118_SETINTR

NAME

DCMD_TP118_SETINTR – Wait for Event on the specified digital input channel

DESCRIPTION

This devctl function waits on a specified event on a specified digital input channel or for a specified time. A pointer to the callers message buffer (*TP118_INTR*) and the size of this structure is passed by the parameters *data_ptr* and *n_bytes* to the device.

The *TP118_INTR* structure has the following layout:

```
typedef struct
{
        int             channel;
        int             edge;
        int             timeout;        /* in seconds, -1 --> forever */
} TP118_INTR, *PTP118_INTR;
```

channel

This argument specifies the digital input channel to wait on. Valid values are 1 up to 6.

edge

This value specifies the transition to wait for. Valid values are:

Value	Description
TP118_EDGE_HI	The function will wait for a Low to High transition
TP118_EDGE_LO	The function will wait for a High to Low transition

timeout

This value specifies the time that shall be waited at maximum. The time is specified in seconds. If the specified time expires, the function will return with error.

EXAMPLE

```
int           fd;
int           result;
TP118_INTR    intrBuf;

...

/*
** wait on a high edge on channel 3, timeout shall occur after 15 seconds
*/
intrBuf.channel    = 3;
intrBuf.edge       = TP118_EDGE_HI;
intrBuf.timeout    = 15;

result = devctl(   fd,
                   DCMD_TP118_SETINTR,
                   &intrBuf,
                   sizeof(intrBuf),
                   NULL);
if (result == EOK)
{
     /* successful waited */
}

...
```

ERRORS

EINVAL	Invalid argument. This error code is returned if either the size of the message buffer is too small, or the specified parameter is out of range.
ETIMEDOUT	The function has waited the specified time without the specified event occurring.

SEE ALSO

Library Reference - devctl()