The Embedded I/O Company

TEWS TECHNOLOGIES

TIP700-SW-95

QNX-Neutrino Device Driver

16 Digital Outputs

Version 1.0.x

User Manual

Issue 1.0.0

October 2004

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

TIP700-SW-95

16 Digital Outputs

QNX-Neutrino Device Driver

Issue	Description	Date
1.0	First Issue	October 18, 2004

Table of Content

1 Introduction

The TIP700-SW-95 QNX-Neutrino device driver allows the operation of a TIP700 16 Digital Outputs IP on QNX-Neutrino operating systems and requires the TEWS QNX-Neutrino IPAC Carrier Driver Software (CARRIER-SW-95).

The TIP700 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:

- Writing digital output value
- Enabling and disabling output watchdog

2 Installation

The TIP700-SW-95 directory on the distribution media contains the following files:

TIP700-SW-95.pdf	This manual in PDF format
TIP700-SW-95.tar	Archive with driver source code

The archive TIP700-SW-95.tar contains the following files and directories:

Driver/tip700.c	Driver source code
Driver/tip700.h	Driver interface definitions and data structures (public)
Driver/tip700def.h	Device driver include file (private)
Driver/Makefile	Script for make utility to build and install the driver
Example/example.c	Example application
Example/Makefile	Script for make utility to build and install the example app.

In order to perform an installation, first login as *root* and copy the TAR archive to the */usr/src* directory and then extract all files and directories.

Before building a new device driver, the TEWS TECHNOLOGIES IPAC carrier driver must be installed properly, because this driver includes the header files *ipac_*.h*, which is part of the IPAC carrier driver distribution. Please refer to the IPAC carrier driver user manual in the directory path */CARRIER-SW-95* on the distribution media.

It's absolute important to extract the TIP700-SW-95.tar in the /usr/src directory otherwise the automatic build with make will fail.

2.1 Build the device driver

Change to the */usr/src/tip700/driver* directory

Execute the Makefile

```
# make install
```

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

2.2 Build the example application

Change to the /usr/src/tip700/example directory

Execute the Makefile

```
# make install
```

After successful completion the example binary (*t700exam*) will be installed in the */bin* directory.

2.3 Start the driver process

To start the TIP700 Resource Manager (Device Driver) you only have to start the TEWS TECHNOLOGIES IPAC Carrier Driver. The Carrier Driver automatically detects installed TEWS IPs and dynamically loads the concerning modul(s).

The TIP700 Resource Manager registers a device for each TIP700 in the QNX-Neutrinos pathname space under following name, where n specifies the used IPAC slot number (please refer to the IPAC Carrier Driver Manual):

`/dev/tip700_`*n*

This pathname must be used in the application program to open a path to the desired TIP700 device.

For debugging you can start the IPAC Carrier Driver with the –V (verbose) option. Now the Resource Manager will print versatile information about TIP700 configuration and command execution on the terminal window. For further details about debugging see IPAC Carrier Driver Manual.

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 TIP700 named by *pathname.* The flags argument controls how the file is to be opened. TIP700 devices must be opened *O_RDWR*.

EXAMPLE

```
int  fd;

fd = open(“/dev/tip700_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 TIP700 driver and should be set to NULL.

The following devctl command codes are defined in *tip700.h* :

Value	Meaning
DCMD_TIP700_WRITE	Write digital output value
DCMD_TIP700_ENABLE_WD	Enable the output watchdog
DCMD_TIP700_DISABLE_WD	Disable the output watchdog

See behind for more detailed information on each control code.

To use these TIP700 specific control codes the header file tip700.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

ENOTTY	Inappropriate I/O control operation. This error code is returned if the requested devctl function is unknown. Please check the argument *dcmd*.

Other function dependant error codes will be described for each devctl code separately. Note, the TIP700 driver always returns standard QNX error codes.

SEE ALSO

Library Reference - devctl()

3.3.1 DCMD_TIP700_WRITE

NAME

DCMD_TIP700_WRITE – Write digital output value

DESCRIPTION

This function attempts to write to the output registers of the TIP700 associated with the file descriptor *filedes.* A pointer to an unsigned short is passed by *data_ptr*. The argument size should always be sizeof(unsigned short) and is passed by *n_bytes*.

Bit 0 of the unsigned short value corresponds to output line 1, Bit 1 corresponds to output line 2 and so on.

EXAMPLE

```
unsigned short outval = 0xF1A3;
...
/*
**  Send request to the device driver
*/
result = devctl(fd, DCMD_TIP700_WRITE, &outVal, sizeof(outVal), NULL);

/*
**  Check the result of the last device I/O operation
*/
if( result == EOK)
{
  printf("\nWrite value successful\n");
}
else
{
  printf("devctl failed (Error = %d) : %s\n", result, strerror(result));
}
```

ERRORS

No errors.

SEE ALSO

Library Reference - devctl()

3.3.2 DCMD_TIP700_ENABLE_WD

NAME

DCMD_TIP700_ENABLE_WD - Enable output watchdog

DESCRIPTION

This devctl function enables the output watchdog facility of the TIP700. If the output watchdog is not retriggered within approximately 120 milliseconds the state of the output register will be set to inactive. The watchdog is triggered implicitly by a write to the output register.

The initialization state (driver startup) of the output watchdog is disabled.

EXAMPLE

```
/*
**  Send request to the device driver
*/
result = devctl(fd, DCMD_TIP700_ENABLE_WD, NULL, 0, NULL);

/*
**  Check the result of the last device I/O operation
*/
if( result == EOK)
{
  printf("\nEnable watchdog successful\n");
}
else
{
  printf("devctl failed (Error = %d) : %s\n", result, strerror(result));
}
```

ERRORS

No errors.

SEE ALSO

Library Reference - devctl()

3.3.3 DCMD_TIP700_DISABLE_WD

NAME

DCMD_TIP700_DISABLE_WD - Disable output watchdog

DESCRIPTION

This ioctl function disables the output watchdog facility of the TIP700.

The initialization state (driver startup) of the output watchdog is disabled.

EXAMPLE

```
/*
**  Send request to the device driver
*/
result = devctl(fd, DCMD_TIP700_DISABLE_WD, NULL, 0, NULL);

/*
**  Check the result of the last device I/O operation
*/
if( result == EOK)
{
  printf("\nDisable watchdog successful\n");
}
else
{
  printf("devctl failed (Error = %d) : %s\n", result, strerror(result));
}
```

ERRORS

No errors.

SEE ALSO

Library Reference - devctl()