

**TIP114-SW-95**  
**QNX4 Device Driver**  
10 Channel Absolute Encoder Interface (SSI)  
on SBS PCI40A Carrier

Version 1.0  
Issue 1.0  
August 20, 2001

TEWS DATENTECHNIK GmbH  
Am Bahnhof 7  
D-25469 Halstenbek  
Germany  
Tel.: +49 (0)4101 4058-0  
Fax.: +49 (0)4101 4058-19  
Email: [Support@tews.com](mailto:Support@tews.com)  
WEB: <http://www.tews.com>

# **TIP114-SW-95**

## **10 Channel Absolute Encoder interface (SSI) on SBS PCI40A Carrier QNX4 Device Driver**

This document contains information, which is proprietary to TEWS DATENTECHNIK GmbH. Any reproduction without written permission is forbidden.

TEWS DATENTECHNIK GmbH has made any effort to ensure that this manual is accurate and complete. However TEWS DATENTECHNIK GmbH reserves the right to change the product described in this document at any time without notice.

This product has been designed to operate with IndustryPack® compatible carriers. Connection to incompatible hardware is likely to cause serious damage.

TEWS DATENTECHNIK GmbH is not liable for any damage arising out of the application or use of the device described herein.

©2001 by TEWS DATENTECHNIK GmbH

IndustryPack is a registered trademark of GreenSpring Computers, Inc

<b>Issue</b>	<b>Description</b>	<b>Date</b>
1.0	First Issue	August 20, 2001

# Table of Contents

<b>1</b>	<b>INTRODUCTION .....</b>	<b>4</b>
<b>2</b>	<b>INSTALLATION.....</b>	<b>5</b>
2.1	Build the device driver .....	5
2.2	Start the driver process .....	5
<b>3</b>	<b>CALLING DRIVER FUNCTIONS.....</b>	<b>6</b>
3.1	qnx_name_locate() .....	6
3.2	Send() .....	8
<b>4</b>	<b>USING THE DRIVER FUNCTIONS.....</b>	<b>10</b>
4.1	Send and Reply buffer.....	10
4.1.1	TIP114_IN_STRUCT .....	10
4.1.2	TIP114_OUT_STRUCT .....	11
4.2	Driver functions.....	12
4.2.1	TIP114_READ .....	12
4.2.2	TIP114_SIMREAD .....	13
4.2.3	TIP114_CONFSET .....	14
4.2.4	TIP114_CONFGET .....	16
<b>5</b>	<b>APPENDIX .....</b>	<b>18</b>
5.1	Defined Values.....	18
5.1.1	Function Codes.....	18
5.1.2	Flags.....	18
5.2	Status and Error Codes.....	18

# **1 Introduction**

The TIP114-SW-95 QNX4 device driver software allows the operation of a TIP114 10 Channel Absolute Encoder Interface IP with QNX4 operating systems with Intel and Intel-compatible x86 CPU and a PCI40A compatible carrier.

Supported features:

- Read absolute value from a specified channel
- Configure and read configuration of channel
- Synchronous read from all (enabled) channels

## 2 Installation

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

Following files are located on the diskette:

tip114.c	Driver source code
isr.c	Interrupt function source code
tip114.h	Driver and application include file
tip114def.h	Driver include file
tpxxhwdep.c	Hardware dependent functions
tpxxhwdep.h	Include file for hardware dependent functions
makefile	Makefile to compile the driver task
example/*	Example application

For installation copy these files into a desired target directory.

### 2.1 Build the device driver

1. Change to the target directory
2. Execute the Makefile

```
make
```

### 2.2 Start the driver process

For starting the driver process, you have to call the executable file 'tip114':

```
./tip114
```

Now the driver process will start a search for the first TIP114 module and will use this. If you want to use a second TIP114 you have to specify this when starting the process. The flag `-I<index>` will specify the index of the module. If you want to start a process using the 3<sup>rd</sup> module, you have to call:

```
./tip114 -I2
```

The process will be started with a default priority of 19, if you want to use an other priority you have to specify this when starting the process with the flag `-P<prio>`.

## 3 Calling Driver Functions

Your application must use the *Send()* function to communicate with the device driver process. Before you can use the *Send()* function you have to locate the process ID of the driver process, this will be done with the *qnx\_name\_locate()* function.

### 3.1 *qnx\_name\_locate()*

#### Synopsis:

```
pid_t qnx_name_locate
(
    nid_t          nid,
    char           *name,
    unsigned       size,
    unsigned       *copies
)
```

#### Description:

The *qnx\_name\_locate()* function locates a process that has registered itself under the given name using the *qnx\_name\_attach()* function.

The argument **nid** should always be 0, **name** specifies the process name of the driver process. **size** shall be set to 0 and the number of **copies** shall be set NULL.

The first thing the application has to do, is to locate the name of the device driver process, because we need the process ID to make the *Send()* calls.

The TIP114 driver processes will be installed as "tip114/<index>". <index> specifies the index number of the TIP114 (see 2.2 Start the driver process). For example the process ID of the 3<sup>rd</sup> TIP114 will be "tip114/2".

#### Returns:

The *qnx\_name\_locate()* function returns a process id if successful; otherwise it returns (-1), and *errno* is set.

#### Errors:

ESRCH                      There is no process that has attached this name.

See also the description of *errno* in the Watcom C Library Reference

#### See also:

Watcom C Library Reference (*qnx\_name\_locate*)

#### Includes:

<sys/name.h>

**Example:**

```
pid_t    tip114_procid;

...

tip114_procid = qnx_name_locate( 0,
                                "tip114/0",
                                0,
                                NULL);

if (tip114_procid != -1)
{
    /* device driver process found */
}
else
{
    /* device driver process not found */
}

...
```

## 3.2 Send()

### Synopsis:

```
int Send
(
    pid_t      pid,
    void      *smsg,
    void      *rmsg,
    unsigned   *snbytes,
    unsigned   *rnbytes
)
```

### Description:

The kernel function *Send()* sends a message pointed to by **smsg** to the process identified by **pid**. Any reply is placed in the message buffer pointed to by **rmsg**. The size of the sent message is **snbytes**, while the size of the reply message is truncated to a maximum of **rnbytes**.

The TIP114 device driver process always waits for the same message buffer type (*TIP114\_IN\_STRUCT*) and always will receive the same reply buffer (*TIP114\_OUT\_STRUCT*). The different functions will use different parts of a union within these structures. The structures will be described in the device driver functions. (see 4 Using the Driver Functions)

### Returns:

The *Send()* function returns a zero on success; On error, it returns (-1), and *errno* is set.

### Errors:

EAGAIN	No more Process Manager to Network Manager queue packets available.
EFAULT	In order to complete the message exchange the current process would have incurred a segment violation. Your buffer(s) may be invalid or too small.
EHOSTUNREACH	The destination node is not in the net mapping, or a physical I/O error occurred trying to communicate to the node.
EINTR	Call interrupted by signal
EINVAL	The virtual circuit buffer cannot be grown due to an invalid message length.
ENOMEM	The virtual circuit buffer cannot be grown because no memory is available.
ESRCH	The process pid does not exist.

See also the description of *errno* in the Watcom C Library Reference

### See also:

Watcom C Library Reference (Send)

### Includes:

<sys/kernel.h>



## Example:

```
/*
** Read the actual value of channel 2
*/
pid_t          tip114_procid;
TIP114_IN_STRUCT in_par;
TIP114_OUT_STRUCT out_par;
int           retval;

...

in_par.command =      TIP114_READ;
in_par.u.chanNo =    2;
retval =      Send( tip114_procid,
                   &in_par,
                   &out_par,
                   sizeof(in_par),
                   sizeof(out_par));
if (retval == 0)
{
    if(out_par.cmdStat == 0)
    {
        /* Send successful completed */
        printf( "Value: %Xh\n",
               out_par.u.value);
    }
    else
    {
        /* Read command failed */
    }
}
else
{
    /* Send failed */
}

...
```

# 4 Using the Driver Functions

## 4.1 Send and Reply buffer

This chapter describes the use of the TIP114 driver functions. These driver functions are using the *Send()* function (described in the chapter 3.2 *Send()*). All the driver function calls are using the same send and reply buffer.

The send buffer has the structure of *TIP114\_IN\_STRUCT* and the reply buffer has the structure of the *TIP114\_OUT\_STRUCT*. These structures will be described below.

### 4.1.1 TIP114\_IN\_STRUCT

```
typedef struct
{
    int          command;          /* I/O command          */
    union
    {
        int      chanNo;          /* channel number       */
        struct
        {
            int   chanNo;          /* channel number       */
            int   clock;          /* clock rate in us     */
            int   databits;      /* number of data bits  */
            int   flags;          /* use definitions      */
        } config;
    } u;
} TIP114_IN_STRUCT;
```

#### Arguments:

- command** This argument specifies the driver function that shall be executed. The values of the driver functions and the use of the send buffer will be described below in the chapter “Driver functions”.
- u** This union shares the buffer depending on the specified function. The following arguments are all part of the union and must be used as a part of it.
- chanNo** This value specifies the channel Number where the value should be read from. This union selection will be used by *TIP114\_READ* and *TIP114\_CONFIGGET*.
- config** This union selection is a structure which has four values that are used for the *TIP114\_CONFIGSET* function (4.2.3 *TIP114\_CONFIGSET*).

## 4.1.2 TIP114\_OUT\_STRUCT

```
typedef struct
{
    int          cmdStat;          /* I/O command status      */
    union
    {
        unsigned long value;      /* input value              */
        unsigned long svalue[10]; /* simultaneous values     */
        struct
        {
            int      chanNo;      /* channel number          */
            int      clock;       /* clock rate in us        */
            int      databits;    /* number of data bits     */
            int      flags;       /* use definitions         */
        } config;
    } u;
} TIP114_OUT_STRUCT;
```

### Arguments:

- cmdStat** This argument returns the function status or error code. These codes are described in the chapter “Status and Error codes”.
- u** This union shares the buffer depending on the specified function. The following arguments are all part of the union and must be used as a part of it.
- value** This union selection is used by *TIP114\_READ* command and returns actual value of the specified channel.
- svalue[]** This union selection is a array of values which will be used when calling the *TIP114\_SIMREAD* command. A value for all channels will be returned, the index identifies the channel number. Values returned for disabled channels are not valid.
- config** This union selection is a structure which returns the actual setup of the specified channel. This selection is used with the *TIP114\_CONFGET* command (4.2.4 TIP114\_CONFGET).

## 4.2 Driver functions

### 4.2.1 TIP114\_READ

This function reads the actual value of the specified channel. The union **u** is used with the selection **chanNo** with the Send buffer. **chanNo** specifies the channel which shall be read. The Reply buffer will be used with the selection **value** in the union **u**, the actual value of the specified channel will be placed in this selection.

#### Example:

```
/*
** Read the actual value from channel 2
*/
pid_t          tip114_procid;
TIP114_IN_STRUCT  in_par;
TIP114_OUT_STRUCT out_par;
int            retval;

...

in_par.command =      TIP114_READ;
in_par.u.chanNo =    2;
retval =      Send( tip114_procid,
                    &in_par,
                    &out_par,
                    sizeof(in_par),
                    sizeof(out_par));

if (retval == 0)
{
    if(out_par.cmdStat == 0)
    {
        /* Send successful completed */
        printf( "Value: %Xh\n",
                out_par.u.value);
    }
    else
    {
        /* Read command failed */
    }
}
else
{
    /* Send failed */
}

...
```

## 4.2.2 TIP114\_SIMREAD

This function reads the actual value of all 10 channels simultaneously. The union **u** is not used with the Send buffer. The Reply buffer will be used with the selection array **svalue** in the union **u**, the actual values of the channels will be placed in this array.

### Example:

```
/*
** Read the actual value from all channels
*/
pid_t          tip114_procid;
TIP114_IN_STRUCT  in_par;
TIP114_OUT_STRUCT out_par;
int           retval;

...

in_par.command = TIP114_SIMREAD;
retval = Send( tip114_procid,
               &in_par,
               &out_par,
               sizeof(in_par),
               sizeof(out_par));
if (retval == 0)
{
    if(out_par.cmdStat == 0)
    {
        /* Send successful completed */
        for (I = 0; I < 10; I++)
        {
            printf( "Value %d: %Xh\n",
                    x,
                    out_par.u.svalue[x]);
        }
    }
    else
    {
        /* Read command failed */
    }
}
else
{
    /* Send failed */
}

...
```

### 4.2.3 TIP114\_CONFSET

This function reads the actual value of the specified channel. The union **u** is used with the selection **conf** with the Send buffer. The values of **conf** are described below. The Reply buffer will not use the union **u**.

The structure **conf** includes the following four parameters. **chanNo** specifies the channel which shall be configured. **clock** specifies the clock rate which shall be used, this value is specified in steps of 1µs. Valid values are '1'..'15' and '0', if the channel shall be disabled. **databits** specifies the number of data bits that shall be used. Allowed values are '1'..'31' and '0' for 32 bits. The last parameter **flags** is a bit field which is ORed value of the defines from the following categories:

The parity mode is selected with:

<i>TIP114_FL_NOP</i>	no parity mode
<i>TIP114_FL_ODP</i>	odd parity mode
<i>TIP114_FL_EVP</i>	even parity mode

The zero flag mode is selected with:

<i>TIP114_FL_NOZERO</i>	don't use the zero flag
<i>TIP114_FL_ZERO</i>	use the zero flag

The gray code mode is selected with:

<i>TIP114_FL_NOGRAY</i>	The input will be interpreted binary
<i>TIP114_FL_GRAY</i>	The input will be interpreted as gray coded

(More details about configuration can be found in the TIP114 User manual.)

## Example:

```
/*
** configure channel 2 with the following parameters:
**     24 data bits
**     5 us clock rate
**     gray coded
**     no zero flag
**     no parity
*/
pid_t          tip114_procid;
TIP114_IN_STRUCT in_par;
TIP114_OUT_STRUCT out_par;
int            retval;

...

in_par.command      = TIP114_CONFSET;
in_par.u.conf.chanNo = 2;
in_par.u.conf.databits = 24;
in_par.u.conf.clock   = 5;
in_par.u.conf.flags  = TIP114_FL_NOP |
                       TIP114_FL_NOZERO |
                       TIP114_FL_GRAY;

retval = Send( tip114_procid,
               &in_par,
               &out_par,
               sizeof(in_par),
               sizeof(out_par));

if (retval == 0)
{
    if(out_par.cmdStat == 0)
    {
        /* Send successful completed */
    }
    else
    {
        /* Configure command failed */
    }
}
else
{
    /* Send failed */
}

...
```

## 4.2.4 TIP114\_CONFGET

This function reads the actual configuration of the specified channel. The union **u** is used with the selection **chanNo** with the Send buffer. **chanNo** specifies the channel where the configuration shall be read from. The Reply buffer will be used with the selection **config** in the union **u**, the **config** structure contains four values, **chanNo** is not used. **clock** returns the actual chosen clock rate in steps is 1  $\mu$ s, a value of '0', signals that the channel is disabled. **databits** returns the number of data bits that are actually used. '0' is returned for 32 bits. The last parameter **flags** is a bit field which is an ORed value of the defines from the following categories:

The parity mode is indicated with:

<i>TIP114_FL_NOP</i>	no parity mode
<i>TIP114_FL_ODP</i>	odd parity mode
<i>TIP114_FL_EVP</i>	even parity mode

The zero flag mode is indicated with:

<i>TIP114_FL_NOZERO</i>	don't use the zero flag
<i>TIP114_FL_ZERO</i>	use the zero flag

The gray code mode is indicated with:

<i>TIP114_FL_NOGRAY</i>	The input will be interpreted binary
<i>TIP114_FL_GRAY</i>	The input will be interpreted as gray coded

(More details about configuration can be found in the TIP114 User manual.)



## Example:

```
/*
** Read the actual configuration of channel 2
*/
pid_t          tip114_procid;
TIP114_IN_STRUCT in_par;
TIP114_OUT_STRUCT out_par;
int            retval;

...

in_par.command =      TIP114_CONFIGGET;
in_par.u.chanNo =    2;
retval =      Send( tip114_procid,
                    &in_par,
                    &out_par,
                    sizeof(in_par),
                    sizeof(out_par));

if (retval == 0)
{
    if(out_par.cmdStat == 0)
    {
        /* Send successful completed */
        printf( "Clockrate: %Xh\n",
                out_par.u.conf.clock);

        ...
    }
    else
    {
        /* Read command failed */
    }
}
else
{
    /* Send failed */
}

...
```

# 5 Appendix

## 5.1 Defined Values

### 5.1.1 Function Codes

TIP815_READ	0x01140001	Read the value of a specified channel
TIP815_SIMREAD	0x01140002	Read the values of all channels
TIP815_CONFSET	0x01140003	Set the configuration of a specified channel
TIP815_CONFGET	0x01140004	Read the configuration of a specified channel

### 5.1.2 Flags

TIP114_FL_NOP	(0<<0)	no parity bit used
TIP114_FL_ODP	(1<<0)	odd parity bit used
TIP114_FL_EVP	(2<<0)	even parity bit used
TIP114_FL_NOZERO	(0<<2)	no zero flag used
TIP114_FL_ZERO	(1<<2)	zero flag used
TIP114_FL_NOGRAY	(0<<3)	no gray code used
TIP114_FL_GRAY	(1<<3)	no gray code used

## 5.2 Status and Error Codes

This values are returned in the Reply buffer in the argument *cmdStat*. This values is only valid if the *Send()* function returns successful, error numbers are defined in the QNX-header file 'error.h.'. The following additional status codes are defined in 'tip114.h'.

TIP114_ERR_ICHAN	0x01140101	An illegal channel number has been specified
TIP114_ERR_CLKZERO	0x01140102	The specified channel is disabled, or all channels are disabled
TIP114_ERR_PERR	0x01140103	A parity error is indicated by the TIP114
TIP114_ERR_IPARA	0x01140104	An invalid parameter values has been specified