The Embedded I/O Company

TEWS TECHNOLOGIES

TIP630-SW-42

VxWorks Device Driver

Reconfigurable FPGA Digital I/O

Version 1.1.x

User Manual

Issue 1.1.0

September 2005

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

TIP630-SW-42

Reconfigurable FPGA Digital I/O

VxWorks Device Driver

Issue	Description	Date
1.0	First Issue	September 24, 2003
1.0.1	General Revision, Release information added	July 12, 2005
1.1.0	CARRIER-SW-42 and Intel x86 support	September 16, 2005

Table of Content

1 Introduction

The TIP630-SW-42 VxWorks device driver software allows the programming of the onboard FPGA and the configuration of the onboard clock generator conforming to the VxWorks I/O system specification. This includes a device-independent basic I/O interface with *open()* and *ioctl()* functions.

The TIP630 driver includes the following functions:

- programming of the onboard Xilinx FPGA
- configuration of the onboard Cypress clock generator

2 Installation

Following files are located in the directory TIP630-SW-42 on the distribution media:

t630drv.c	TIP630 Driver Source
tip630.h	TIP630 Driver Include File
t630def.h	TIP630 Internal Driver Include File
t630exam.c	TIP630 Device Driver Example Application
hexfile.zip	TIP630 FPGA Example design as ZIP archive (hexfile.h)
t630FPGAdrv.c	TIP630 FPGA Example Driver Source
tip630FPGA.h	TIP630 FPGA Example Driver Include File
t630FPGAdef.h	TIP630 FPGA Example Driver Internal Include File
t630FPGAexam.c	TIP630 FPGA Example Device Driver Example Application
tdhal.h	Hardware dependent interface functions and definitions
ipac_carrier.h	Carrier driver interface definitions
TIP630-SW-42.pdf	TIP630 VxWorks Device Driver Manual
Release.txt	Information about the Device Driver Release

In order to perform an installation, copy the contents of the directory TIP630-SW-42 on the distribution media into a subdirectory in your project path (e.g. ./tip630). Before including the driver C-files into the Tornado project tree the archive hexfile.zip must be unzipped.

2.1 IPAC Carrier Driver

The TEWS TECHNOLOGIES IPAC carrier driver CARRIER-SW-42 is part of the TIPxxx-SW-42 distribution and should be used to setup supported carrier boards and retrieve resource information used for TIP630 device creation. Please refer to the CARRIER-SW-42 user manual for detailed description how to install and setup the CARRIER-SW-42 device driver.

How to use the carrier driver in the application program is shown in the programming example t630exam.c.

If the IPAC carrier driver isn't used for the TIP630 driver setup, the application software has to setup carrier board hardware and mapping of device memory by itself.

3 I/O system functions

This chapter describes the driver-level interface to the I/O system. The purpose of these functions is to install the driver in the I/O system, add and initialize devices.

3.1 t630Drv()

NAME

t630Drv() - installs the TIP630 device driver in the I/O system and initialize the driver.

SYNOPSIS

```
STATUS t630Drv
(
        void
)
```

DESCRIPTION

This function installs the TIP630 driver in the I/O system, allocates driver resources and initializes them. The call of this function is the first thing we have to do, before adding any device to the system or performing any I/O request.

RETURNS

OK or ERROR if the driver cannot be installed

SEE ALSO

VxWorks Programmer's Guide: I/O System

3.2 t630DevCreate()

NAME

t630DevCreate() - adds a TIP630 device to the system and initialize device hardware

SYNOPSIS

```
STATUS t630DevCreate
(
        char                *name,
        unsigned long       *IpIoAddr,
)
```

DESCRIPTION

This routine creates a device for the module specified by the IP I/O address that will be serviced by the TIP630 device driver. This function must be called before performing any I/O request to this device.

Function Arguments:

Name

Pointer to null terminated unique string to identify this device (e.g. "/t630A").

IpIoAddr

Pointer to the IP I/O space (TIP630 device register). For TIP630 modules plugged onboard on a TVME8240, this is for example 0xFFF58000 for the first IP slot. For TIP630 modules plugged on external carrier boards, this address depends on the mapping of the VMEbus windows and the address configuration of the carrier board.

EXAMPLE

```
#include "tip630.h"

...

/*-------------------------------------------------------------
  Create the device "/t630D" for the module plugged at
  local IP-slot D on a TVME8240.
  -------------------------------------------------------------*/
status = t630DevCreate( "/t630D",
                        0xFFF58300 );

...
```

RETURNS

OK or ERROR if the driver is not installed or the device already exists or any other error occurred during the creation.

ERRORS

VxWorks error codes	Error codes returned by the VxWorks system functions like iosDevAdd().
S_t630Drv_NOMEM	Unable to allocate memory to store internal device information.
S_t630Drv_NOSEM	Unable to allocate memory for security semaphore.

4 I/O interface functions

This chapter describes the interface to the basic I/O system.

4.1 open()

NAME

open() - open a device or file

SYNOPSIS

```
int open
(
        const char   *name,      /* name of the device to open      */
        int          flags,      /* not used for TIP630 driver, must be 0   */
        int          mode        /* not used for TIP630 driver, must be 0   */
)
```

DESCRIPTION

Before I/O can be performed to the TIP630 device, a file descriptor must be opened by invoking the basic I/O function *open()*.

EXAMPLE

```
/*--------------------------------------
  open the device named "/t630D" for I/O
--------------------------------------*/
fd = open ("/t630D", 0, 0);
```

RETURNS

A device descriptor number, or ERROR if the device does not exist or no device descriptors are available.

SEE ALSO

ioLib, basic I/O routine - open()

4.2 ioctl()

NAME

ioctl() - performs an I/O control function

SYNOPSIS

```
int ioctl
(
        int         fd,         /* device descriptor */
        int         function,   /* function code */
        int         arg         /* optional function-dependent argument */
)
```

DESCRIPTION

Special I/O operation that does not fit to the standard basic I/O calls will be performed by calling the *ioctl()* function with a specific function code and an optional function-dependent argument.

The **function** parameter selects the action, which will be executed by the driver. The structure of the **arg** parameter depends on the function. The different functions are described behind the general description of this function.

FIO_T630_PROGRAM_FPGA	Program the onboard FPGA
FIO_T630_SET_CLOCK	Program the onboard clock generator CY22150

RETURNS

OK or ERROR if the device descriptor does not exist, the function code is unknown or an error occurred.

INCLUDE FILES

ioLib.h, tip630.h

SEE ALSO

ioLib, basic I/O routine - ioctl(), VxWorks Programmer's Guide: I/O System.

4.2.1 FIO_T630_PROGRAM_FPGA

NAME

FIO_T630_PROGRAM_FPGA – Program the onboard FPGA

DESCRIPTION

With this ioctl function the onboard FPGA can be programmed with a special generated byte-array derived from the *raw bit file* of your FPGA-design. Further information on generating the necessary files and structures is available in the TIP630 hardware manual.

The programming array (*T630_PROG_PARAM*) will be passed in the ioctl argument ***arg*** to the driver.

```
typedef struct
{
        unsigned char*          buf;
        unsigned long           len;
} T630_PROG_PARAM;
```

buf

Specifies a pointer to an array, where the programming bytes are located.

len

Specifies the size of the programming array (the number of bytes to be programmed into the FPGA). The length should be approx. 166,980 bytes.

EXAMPLE

```
#include      "tip630.h"

unsigned char bytearray[] = {
     0xFF,
     0xFF,
     0xFF,
     0xFF,
     0x55,
     0x99,
     0xAA,
     0x66,
...
}

...
```

```
int                     fd;
int                     retval;
T630_PROG_PARAM         ProgParam;

...

/*----------------------------------------
  Program the FPGA from a static byte-array
  ----------------------------------------*/
ProgParam.buf = bytearray;
ProgParam.len = sizeof(bytearray);

retval = ioctl(fd, FIO_T630_PROGRAM_FPGA, (int)&ProgParam);
if (retval == ERROR)
{
     /* handle function specific error conditions */
}

...
```

ERRORS

S_t630Drv_TIMEOUT	The FPGA does not respond within a specified time.
S_t630Drv_IRANGE	The number of bytes to program is too big.

SEE ALSO

ioLib, basic I/O routine - ioctl(), VxWorks Programmer's Guide: I/O System.

4.2.2 FIO_T630_SET_CLOCK

NAME

FIO_T630_SET_CLOCK – Program the onboard clock generator CY22150

DESCRIPTION

This ioctl function is used to program the onboard clock generator CY22150.

The programming array (*T630_PROG_PARAM*) will be passed in the ioctl argument ***arg*** to the driver.

```
typedef struct
{
   unsigned char   DeviceAddr;          /*  7bit address */
   unsigned char   ClkOE;               /*  2bit relevant */
   unsigned char   Div1Src;             /*  1bit */
   unsigned char   Div2Src;             /*  1bit */
   unsigned char   Div1N;               /*  7bit */
   unsigned char   Div2N;               /*  7bit */
   unsigned char   XDrv;                /*  2bit */
   unsigned char   CapLoad;             /*  8bit */
   unsigned char   Pump;                /*  3bit */
   unsigned short  PBcount;             /* 10bit */
   unsigned char   POcount;             /*  1bit */
   unsigned char   Qcount;              /*  7bit */
   unsigned char   ClkASrc;             /*  3bit */
   unsigned char   ClkBSrc;             /*  3bit */
} T630_CLOCK_PARAM;
```

DeviceAddr

Specifies the I2C-address of the CY22150, default value is 0x69h.

ClkOE

Specifies the clock lines which are enabled. Only LCLK1 and LCLK2 are valid for the TIP630, i.e. values from 0 to 3 for this item.

Div1Src

Specifies the internal clock source for Clock1 (0=VCO, 1=REF)

Div2Src

Specifies the internal clock source for Clock2 (0=VCO, 1=REF)

Div1N

Specifies the frequency divider for Clock1 (values from 4 to 127 allowed)

Div2N

Specifies the frequency divider for Clock2 (values from 4 to 127 allowed)

XDrv

Specifies the input crystal oscillator drive control (0=1x, 1=2x, 2=4x, 3=8x)

CapLoad

Specifies the input load capacitor control (values from 0 to 255 allowed)

Pump

Specifies the charge pump (values from 0 to 7 allowed)

PBcount

Specifies a part of the PLL P counter (10bit values allowed).

POcount

Specifies a part of the PLL P counter (value 0 or 1 allowed)

Qcount

Specifies the PLL Q counter (7bit values allowed)

ClkASrc

Specifies the cross point switching matrix for Clock1

0	Reference input
1	DIV1CLK / Div1N
2	DIV1CLK / 2 (fixed /2 divider option. Div1N must be divisable by 4)
3	DIV1CLK / 3 (fixed /3 divider option. Div1N must be set to 6)
4	DIV2CLK / Div2N
5	DIV2CLK / 2 (fixed /2 divider option. Div2N must be divisable by 4)
6	DIV2CLK / 4 (fixed /4 divider option. Div2N must be divisable by 8)
7	Reserverd – do not use.

ClkASrc

Specifies the cross point switching matrix for Clock2

0	Reference input
1	DIV1CLK / Div1N
2	DIV1CLK / 2 (fixed /2 divider option. Div1N must be divisable by 4)
3	DIV1CLK / 3 (fixed /3 divider option. Div1N must be set to 6)
4	DIV2CLK / Div2N
5	DIV2CLK / 2 (fixed /2 divider option. Div2N must be divisable by 4)
6	DIV2CLK / 4 (fixed /4 divider option. Div2N must be divisable by 8)
7	Reserverd – do not use.

Be sure to specify correct values for each parameter. Otherwise the clock generator may run unstable. To calculate the values you can use the tool *CyberClocks* from Cypress.

EXAMPLE

```
#include      "tip630.h"

int                     fd;
int                     retval;
T630_CLOCK_PARAM        ClockParam;

/*----------------------------------
  Set Parameters for Clock Generator:
    Clock1: 10MHz
    Clock2: 50MHz
  ----------------------------------*/
    ClockParam.DeviceAddr   = 0x69;
    ClockParam.ClkOE        = 0x03;
    ClockParam.CapLoad      = 0x00;
    ClockParam.ClkASrc      = 0x01;
    ClockParam.ClkBSrc      = 0x05;
    ClockParam.Div1N        = 0x0A;
    ClockParam.Div2N        = 0x04;
    ClockParam.Div1Src      = 0x00;
    ClockParam.Div2Src      = 0x00;
    ClockParam.PBcount      = 0x08;
    ClockParam.POcount      = 0x01;
    ClockParam.Qcount       = 0x06;
    ClockParam.Pump         = 0x01;
    ClockParam.XDrv         = 0x01;

retval = ioctl(fd, FIO_T630_SET_CLOCK, (int)&ClockParam);
if (retval == ERROR)
{
     /* handle function specific error conditions */
}
```

ERRORS

S_t630Drv_I2C_DEVERR	No answer from specified I2C address.
S_t630Drv_I2C_COMERR	Error during communication with clock generator via I2C

SEE ALSO

ioLib, basic I/O routine - ioctl(), VxWorks Programmer's Guide: I/O System.

5 FPGA Example Driver

In addition to the TIP630 low level driver there is a sample driver included in this package. This driver is held very simple and just supports register access to the provided FPGA-design example. It can be used to create own drivers to use special FPGA designs in the desired way. The TIP630 low level driver does not register any interrupts, as well as the FPGA example driver, so the interrupt handling has to be added separately.