



FastLink Products

tekpciLink

VxWorks Driver for FPMC Modules

User's Manual

TM-3A-10
10 Nov 2003

TEK Microsystems has made every effort to ensure that this document is accurate and complete. However, TEK reserves the right to make changes and improvements to the products described in this document at any time and without notice.

This product is covered by a limited warranty that is described in the manual. Other than the stated limited warranty, TEK disclaims all other warranties, including the warranties of merchantability and of fitness for a particular purpose. In the event of a failure of the hardware or software described in this document, TEK's obligation is limited to repair or replacement of the defective item, or, if the item cannot be repaired or replaced, a refund of the purchase price for the item. TEK assumes no liability arising out of the application or use of the hardware or software, and assumes no responsibility for direct, indirect, incidental or consequential damages of any kind.

The electronic equipment described in this document generates, uses, and can radiate radio frequency energy. Operation of this equipment in a residential area is likely to cause radio interference, in which case the user, at his own expense, will be required to take whatever measures may be required to correct the interference.

TEK Microsystems' products are not authorized for use as critical components in life support devices or systems without the express written agreement of an officer of TEK Microsystems.

This document is Copyright © 1998-2003, TEK Microsystems, Incorporated. All Rights Reserved.

FastLink, FastPCI and FastPMC are trademarks of TEK Microsystems, Incorporated.

HOTLink is a trademark of Cypress Semiconductor.

TAXIchip is a trademark of Advanced Micro Devices.

VxWorks and Tornado are trademarks of Wind River Systems, Inc.

Other trademarks and registered trademarks used are owned by their respective manufacturers.

Table of Contents

| | |
|---|----------|
| Product Description | 1 |
| Supported Products and Protocols | 1 |
| Revision History | 1 |
| Architecture | 2 |
| Device Driver Layers..... | 2 |
| Data Structures..... | 2 |
| Status and Diagnostic Messages | 2 |
| Initialization..... | 4 |
| Function Reference | 5 |
| FastLink Driver: Basic Functions..... | 5 |
| tekpciLinkInitDrv () | 6 |
| tekpciLinkGetHandle ()..... | 7 |
| tekpciLinkDisplay() | 8 |
| tekpciLinkDmaEnable() | 9 |
| tekpciLinkDmaConfig ()..... | 10 |
| tekpciLinkDmaStatus () | 11 |
| tekpciLinkSetClockFreq()..... | 12 |
| tekpciLinkSetLocalBusClockFreq()..... | 13 |
| FastLink Driver: FPGA Functions..... | 14 |
| tekpciLinkFpgaGetDwg() | 15 |
| tekpciLinkFpgalsLoaded() | 16 |
| tekpciLinkFpgaKill()..... | 17 |
| tekpciLinkFpgaLoad()..... | 18 |
| tekpciLinkFpgaLoadByDwg() | 19 |
| tekpciLinkFpgaLookupId() | 20 |
| tekpciLinkFpgaReset() | 21 |
| FastLink Driver: Parameter Functions..... | 22 |
| tekpciLinkParmsDisplay() | 23 |
| tekpciLinkRxGetParms() | 24 |
| tekpciLinkRxSetParms() | 25 |
| tekpciLinkTxGetParms()..... | 26 |
| tekpciLinkTxSetParms() | 27 |
| FastLink Driver: Receive Functions..... | 28 |
| tekpciLinkRxClearErrors() | 29 |
| tekpciLinkRxGetCount()..... | 30 |
| tekpciLinkRxGetErrors() | 31 |
| tekpciLinkRxGetPtr()..... | 32 |
| tekpciLinkRxDmaQueue () | 33 |
| tekpciLinkRxDmaQueueChannel ()..... | 34 |
| FastLink Driver: Transmit Functions | 35 |
| tekpciLinkTxClearErrors()..... | 36 |
| tekpciLinkTxGetCount()..... | 37 |
| tekpciLinkTxGetErrors()..... | 38 |
| tekpciLinkTxGetFree() | 39 |

| | |
|---------------------------------------|----|
| tekpciLinkTxGetPtr()..... | 40 |
| tekpciLinkTxDmaQueue ()..... | 41 |
| tekpciLinkTxDmaQueueChannel ()..... | 42 |
| FastLink Driver: Info Functions | 43 |
| tekpciLinkInfoDisplay()..... | 44 |
| tekpciLinkInfoGet() | 45 |
| FastLink Driver: Test Functions | 46 |
| tekpciLinkTestDatabus()..... | 47 |
| tekpciLinkTestClock() | 48 |
| tekpciLinkTestLed()..... | 49 |
| tekpciLinkTestMemory() | 50 |
| tekpciLinkTestLoopback() | 51 |

Product Description

This software driver provides a VxWorks device driver, providing an API for TEK's FastLink family of PMC, PCI and Compact PCI data link products.

TEK's FastLink products use customizable FPGA hardware to implement the low-level interface to the data link hardware. The software driver automatically downloads the appropriate FPGA image for the hardware model and interface being used. The application is responsible for selecting the desired I/O interface, such as raw I/O streams, packet-based, or digital video. Based on this selection, the driver loads the appropriate FPGA image.

This software driver is intended to abstract the differences between the various types of data link products. Specifically, the driver API supports raw streaming I/O for products such as the AMD TAXIchip and Cypress HOTLink interfaces with a wide range of low-level protocols and electrical interfaces.

Supported Products and Protocols

This software driver API supports the following TEK protocols:

- Raw streaming I/O (simplex raw byte stream for each RX or TX channel)
- Packet-based I/O (byte stream with packet boundaries marked with control characters)

For specific hardware supported by this driver, see the README file shipped with the software distribution.

Revision History

For software driver revision information, please see the **CHANGES** file in the driver distribution.

Architecture

Device Driver Layers

The VxWorks software driver is implemented using a layered architecture:

- The lowest layer provides generic PCI services to the driver. This module is the only BSP-specific module, although most of the files are CPU architecture specific due to the use of Big/Little Endian macros.
- The next layer provides interface functions to configure, control and monitor the PCI interface controller.
- The following layer provides an intermediate layer to download FPGAs and manage EEPROMS.
- The top layer provides a set of interface functions to configure, control and monitor products in TEK's FastLink family, including HOTLink and TAXI based PMC and PCI products.

Data Structures

The primary data structure used by the device driver is the **TEKPCI_HANDLE** data structure, defined in **tekpci.h**. The **TEKPCI_HANDLE** structure is used as a "handle" to a device. A user application may support any number of devices by creating separate **TEKPCI_HANDLE** structures, one for each device.

Status and Diagnostic Messages

Some of the driver functions generate formatted text messages to report driver, hardware, or software status information. In order to support diverse operating environments, these function take two output-related arguments: *output* and *handle*. The *output* argument is a pointer to an application-defined message handler. The handler isolates the driver's display functions from the operating system's I/O environment, and may log the messages to a file, display them, or perform other processing.

The *handle* argument is passed to the output function as a convenience to the application. Using this argument, the handler can support different task, process, or window contexts. The driver does not evaluate the *handle* argument. If the *handle* argument is not needed, the caller may set it NULL.

For example, in a typical ANSI C environment, the following code may be used to implement the output function and execute **tekpciLinkDisplay()**:

```
static void output_puts (void *handle, const char *s)
{
    printf ("%s", s);
    fflush (stdout);
}

STATUS display_int_status (TEKPCI_HANDLE pci)
{
    if (tekpciLinkDisplay (pci, output_puts, NULL) != OK) {
        return (ERROR);
    }

    return (OK);
}
```

The driver functions do not call printf() or any other I/O functions directly. Note that the output function cannot be NULL unless explicitly stated. If no output is needed, the application may write a dummy handler that discards the output.

Initialization

The device driver is expected to operate in an environment which does not perform PCI “Plug and Play” initialization, and assumes that the application has a predefined hardware configuration and address map for the PCI hardware. Before using any driver calls, each driver layer should be initialized by calling **tekpciLinkInitDrv()**. Each instance of a **TEKPCI_HANDLE** is initialized by calling **tekpciLinkGetHandle()**.

Function Reference

FastLink Driver: Basic Functions

The **tekpciLink** basic functions initialize the driver, attach a **TEKPCI_HANDLE** data structure to the target FPMC device, and provide control and status functions which are not specific to an individual RX or TX channel.

The **tekpciLink** basic functions are:

- **tekpciLinkInitDrv**, which performs global initialization of the **tekpciLink** driver.
- **tekpciLinkGetHandle**, which returns a handle to an FPMC device.
- **tekpciLinkDisplay**, which displays control and status register information.
- **tekpciLinkDmaEnable**, which enables or disables DMA requests.
- **tekpciLinkDmaConfig**, which configures DMA requests.
- **tekpciLinkDmaStatus**, which displays DMA status.
- **tekpciLinkSetClockFreq**, which configures the clock synthesizer.
- **tekpciLinkSetLocalBusClockFreq**, which configures the local bus clock frequency.

Each of these functions is described in more detail below.

tekpciLinkInitDrv ()

NAME *tekpciLinkInitDrv ()* – Get Device Handle.

SYNOPSIS `STATUS tekpciLinkInitDrv (void);`

DESCRIPTION This functions initializes the tekpciLink driver.

INCLUDE FILES **tekpci.h, tekpciLink.h**

RETURNS OK, or ERROR if an error is encountered accessing the hardware.

tekpciLinkGetHandle ()

| | |
|----------------------|--|
| NAME | <i>tekpciLinkGetHandle()</i> – Get Device Handle. |
| SYNOPSIS | <pre>TEKPCI_HANDLE tekpciLinkGetHandle (int slot);</pre> |
| DESCRIPTION | This functions returns a handle to a device. Parameter <i>slot</i> specifies the PMC slot occupied by the target device. For single-slot carriers, specify zero. |
| INCLUDE FILES | tekpci.h, tekpciLink.h |
| RETURNS | A device handle, or NULL if the target device is invalid. |

tekpciLinkDisplay()

| | |
|----------------------|--|
| NAME | <i>tekpciLinkDisplay()</i> – Display control/status registers. |
| SYNOPSIS | <pre>STATUS tekpciLinkDisplay (TEKPCI_HANDLE pci, void (*handler) (void *handle, const char *str), void *handle);</pre> |
| DESCRIPTION | <p>This function reads the current control and status registers from the device specified by <i>pci</i> and compiles a formatted status string for output. Parameter <i>handler</i> specifies a caller-defined handler. Parameter <i>handle</i> specifies a handler argument. The function calls the handler, passing <i>handle</i> and the status string <i>str</i> as arguments.</p> <p>The specific information returned varies according to the FPGA image loaded onto the device. FPGA images not supporting this function return an error.</p> |
| INCLUDE FILES | tekpci.h, tekpciLink.h |
| RETURNS | OK, or ERROR if an error is encountered accessing the hardware. |

tekpciLinkDmaEnable()

| | |
|----------------------|--|
| NAME | <i>tekpciLinkDmaEnable()</i> – Enable or disable DMA requests. |
| SYNOPSIS | <pre>STATUS tekpciLinkDmaEnable (TEKPCI_HANDLE pci, int enable);</pre> |
| DESCRIPTION | If argument <i>enable</i> is zero, this function disables DMA requests from the client process. If argument <i>enable</i> is greater than zero, this function enables DMA requests. Typically, DMA requests must be enabled before queuing DMA requests. |
| INCLUDE FILES | tekpci.h, tekpciLink.h |
| RETURNS | OK, or ERROR if an error is encountered accessing the hardware. |
| CAUTIONS | DMA requests should not be disabled for channels with pending DMA requests in their queues. |

tekpciLinkDmaConfig ()

| | |
|----------------------|--|
| NAME | <i>tekpciLinkDmaConfig ()</i> – Set DMA Timeout. |
| SYNOPSIS | <pre>STATUS tekpciLinkDmaConfig (TEKPCI_HANDLE pci, int channel, int timeout, int threshold);</pre> |
| DESCRIPTION | <p>This function sets configuration values for DMA transfers over the channel specified by <i>channel</i> on the device specified by <i>pci</i>.</p> <p>If parameter <i>timeout</i> is true, parameter <i>threshold</i> specifies the timeout value, which can range from zero to 255.</p> |
| INCLUDE FILES | tekpci.h, tekpciLink.h |
| RETURNS | OK, or ERROR if an error is encountered. |
| SEE ALSO | tekpciLinkDmaStatus () |

tekpciLinkDmaStatus ()

| | |
|----------------------|--|
| NAME | <i>tekpciLinkDmaStatus ()</i> – Display DMA controller information. |
| SYNOPSIS | <pre>STATUS tekpciLinkDmaStatus (TEKPCI_HANDLE pci, void (*handler) (void *handle, const char *str), void *handle);</pre> |
| DESCRIPTION | This function returns a formatted string describing the current settings of the DMA controllers on the device specified by <i>pci</i> . Parameter <i>handler</i> specifies a caller-defined handler. Parameter <i>handle</i> specifies a handler argument. The function calls the handler, passing <i>handle</i> and formatted string <i>str</i> as arguments. |
| INCLUDE FILES | tekpci.h, tekpciLink.h |
| RETURNS | OK, or ERROR if an error is encountered. |
| SEE ALSO | tekpciLinkDmaConfig () |

tekpciLinkSetClockFreq()

| | |
|----------------------|--|
| NAME | <i>tekpciLinkSetClockFreq()</i> – Sets clock synthesizer. |
| SYNOPSIS | <pre>STATUS tekpciLinkSetClockFreq (TEKPCI_HANDLE pci, int channel, int *freq);</pre> |
| DESCRIPTION | <p>This function sets the clock synthesizer for the device specified by <i>pci</i>. The function assumes that the FPGA is loaded and compliant.</p> <p>The hardware module is assumed to have <i>n</i> clock synthesizers, with zero through <i>n</i>-2 mapped to channels zero through <i>n</i>-2, and clock synthesizer <i>n</i>-1 mapped to the local bus. Specifying <i>-1</i> for <i>channel</i> specifies the local bus. Otherwise, <i>channel</i> must range from zero to <i>n</i>-1 where <i>n</i> is the number of synthesizers other than the local bus synthesizer supported by the device.</p> <p>Parameter <i>freq</i> specifies the frequency in Hertz, which is modified by the function to reflect the actual frequency programmed into the synthesizer. For arbitrary frequencies, the programmed frequency is set to within a small tolerance of the specified frequency.</p> <p>If an invalid <i>channel</i> or <i>freq</i> is supplied, the function returns an error.</p> |
| INCLUDE FILES | tekpci.h, tekpciLink.h |
| RETURNS | ERROR if given an invalid parameter or there is a problem accessing hardware. OK otherwise. |

tekpciLinkSetLocalBusClockFreq()

| | |
|----------------------|---|
| NAME | <i>tekpciLinkSetLocalBusClockFreq()</i> – Set the local bus clock frequency. |
| SYNOPSIS | <pre>STATUS tekpciLinkSetLocalBusClockFreq (TEKPCI_HANDLE pci, int *freq);</pre> |
| DESCRIPTION | <p>This function sets the frequency of the local bus clock, and is equivalent to calling tekpciLinkSetClockFreq() with parameter <i>channel</i> set to <i>-1</i>.</p> <p>For devices with a fixed local bus clock, this function returns an error unless parameter <i>freq</i> specifies the proper value.</p> <p>The local bus clock frequency is typically based on the bus clock frequency. If so, deviations in the bus clock frequency will result in proportional deviations in the local bus clock frequency.</p> |
| INCLUDE FILES | tekpci.h, tekpciLink.h |
| RETURNS | OK, or ERROR if an error is encountered accessing the hardware. |
| SEE ALSO | tekpciLinkGetLocalBusClockFreq() |

FastLink Driver: FPGA Functions

The **tekpciLink** FPGA functions download FPGA images, identify FPGA images loaded into devices, and access standard FPGA images.

FPGA IDs consist of five digit drawing numbers, two digit FPGA sizes, and two digit revision numbers. For example, the standard streaming I/O FPGA for a HOTLink PMC module is drawing number 23623, FPGA size 50, and revision 01, making the ID value 236235001.

When looking up FPGA images, if the revision value is non-zero, the specific revision number is used; otherwise, the highest available revision is used. For example, if the driver defines FPGA images 236235001 (rev A) and 236235002 (rev B), ID 236235001 specifies revision A. An application can specify the latest available revision, in this case rev B, by using ID 236235000.

To download a driver-supplied FPGA image, applications can first check for a loaded FPGA by calling **tekpciLinkFpgaIsLoaded()**. Function **tekpciLinkFpgaGetDwg()** can then be called to see if the desired image is already loaded. If not, calling **tekpciLinkFpgaKill()** removes the FPGA image in preparation for a new one. If that know the correct FPGA image can call **tekpciLinkFpgaLoad()**. Otherwise, applications can call **tekpciLinkFpgaLoadByDwg()**. The driver also provides **tekpciLinkFpgaLookupId()** to map FPGA names to ID values.

tekpciLinkFpgaGetDwg()

| | |
|----------------------|--|
| NAME | <i>tekpciLinkFpgaGetDwg()</i> – Get FPGA drawing and revision information. |
| SYNOPSIS | <pre>STATUS tekpciLinkFpgaGetDwg (TEKPCI_HANDLE pci, int *dwg, int *rev);</pre> |
| DESCRIPTION | <p>This function returns the drawing and revision information of the FPGA image currently loaded into the device specified by <i>pci</i>.</p> <p>The drawing number is returned in the caller-allocated location specified by <i>dwg</i>. The revision number is returned in the caller-allocated location specified by <i>rev</i>. If either value is NULL, NULL is returned for that value. If no FPGA is currently loaded, the function returns an error.</p> |
| INCLUDE FILES | tekpci.h, tekpciLink.h |
| RETURNS | OK if an FPGA is loaded, ERROR otherwise. |
| SEE ALSO | tekpciLinkIsFpgaLoaded(), tekpciFlexFpgaGetDwg() |

tekpciLinkFpgaIsLoaded()

| | |
|----------------------|---|
| NAME | <i>tekpciLinkFpgaIsLoaded()</i> – Test for loaded FPGA. |
| SYNOPSIS | <pre>STATUS tekpciLinkFpgaIsLoaded (TEKPCI_HANDLE pci);</pre> |
| DESCRIPTION | This function tests the device specified by <i>pci</i> for a loaded FPGA image. |
| INCLUDE FILES | tekpci.h, tekpciLink.h |
| RETURNS | OK if an FPGA is loaded, ERROR otherwise. |
| SEE ALSO | tekpciLinkFpgaLoad() |

tekpciLinkFpgaKill()

| | |
|----------------------|---|
| NAME | <i>tekpciLinkFpgaKill()</i> – Reset FPGA image. |
| SYNOPSIS | <pre>STATUS tekpciLinkFpgaKill (TEKPCI_HANDLE pci);</pre> |
| DESCRIPTION | This function resets the FPGA image on the device specified by <i>pci</i> , forcing the FPGA to exit user mode and await a new configuration image. All local bus operations are invalid until a new image is downloaded. |
| INCLUDE FILES | tekpci.h, tekpciLink.h |
| RETURNS | OK, or ERROR if an error is encountered accessing the hardware. |
| SEE ALSO | tekpciLinkFpgaLoad() |

tekpciLinkFpgaLoad()

| | |
|----------------------|---|
| NAME | <i>tekpciLinkFpgaLoad()</i> – Load an FPGA image. |
| SYNOPSIS | <pre> STATUS tekpciLinkFpgaLoad (TEKPCI_HANDLE pci, VERBOSE level, const TEK_FPGA_ENTRY *fep, void (*output) (void *handle, const char *str), void *handle); </pre> |
| DESCRIPTION | <p>This function loads the FPGA image specified by <i>fep</i> into the device specified by <i>pci</i>.</p> <p>Argument <i>level</i> specifies the level of status information returned by the function. Defined values include:</p> <ul style="list-style-type: none"> VERBOSE_NONE No status output. Parameters <i>output</i> and <i>handle</i> may be NULL. VERBOSE_ERROR Generate output for errors only. VERBOSE_ALL Generate output as the FPGA is loaded. <p>The function invokes the caller-defined handler <i>output</i>, passing <i>handle</i> as argument <i>handle</i> and the status display string, if any, as argument <i>str</i>.</p> |
| INCLUDE FILES | tekpci.h, tekpciLink.h, tekFpga.h |
| RETURNS | OK if an FPGA is loaded, ERROR otherwise. |
| SEE ALSO | tekFpga(), tekpciLinkFpgaLoadByDwg(), tekpciLinkFpgaIsLoaded() |

tekpciLinkFpgaLoadByDwg()

| | |
|----------------------|--|
| NAME | <i>tekpciLinkFpgaLoadByDwg()</i> – Load FPGA image by ID value. |
| SYNOPSIS | <pre>STATUS tekpciLinkFpgaLoadByDwg (TEKPCI_HANDLE pci, VERBOSE level, int dwg, int revision, void (*output) (void *handle, const char *str), void *handle);</pre> |
| DESCRIPTION | <p>This function locates the FPGA image specified by <i>dwg</i> and <i>revision</i> and calls tekpciLinkFpgaLoad() to load the device specified by <i>pci</i>. If the image cannot be located or the load fails, an error is returned.</p> <p>Argument <i>dwg</i> specifies the drawing number of the FPGA image. The drawing number may be queried by calling tekpciLinkFpgaLookupId() with the FPGA's name. The <i>dwg</i> and <i>revision</i> arguments combined with the FPGA size generate the FPGA image ID.</p> <p>See tekpciLinkFpgaLoad() for further details and descriptions of other arguments.</p> |
| INCLUDE FILES | tekpci.h, tekpciLink.h |
| RETURNS | OK if the FPGA is loaded, ERROR otherwise. |
| SEE ALSO | tekpciLinkFpgaLoad(), tekpciLinkFpgaIsLoaded() |

tekpciLinkFpgaLookupId()

| | |
|----------------------|--|
| NAME | <i>tekpciLinkFpgaLookupId()</i> – Return FPGA ID. |
| SYNOPSIS | <pre>STATUS tekpciLinkFpgaLookupId (TEKPCI_HANDLE pci, const char *name, int *id);</pre> |
| DESCRIPTION | <p>This function returns the FPGA ID for the FPGA function specified by <i>name</i> and the device specified by <i>pci</i>.</p> <p>Specific names are device-dependent. If “default” or NULL is specified for <i>name</i>, the most generic streaming I/O FPGA is returned.</p> <p>If the name is invalid or if the required FPGA image is not accessible, an error is returned.</p> |
| INCLUDE FILES | tekpci.h, tekpciLink.h |
| RETURNS | OK if an FPGA image is available, ERROR otherwise. |
| SEE ALSO | tekFpgaImageLoad() |

tekpciLinkFpgaReset()

| | |
|----------------------|--|
| NAME | <i>tekpciLinkFpgaReset()</i> – Reset FPGA internal logic. |
| SYNOPSIS | <pre>STATUS tekpciLinkFpgaReset (TEKPCI_HANDLE pci);</pre> |
| DESCRIPTION | <p>This function resets the FPGA on the device specified by <i>pci</i>. The effects of the reset function are FPGA-dependent. Typically, the FPGA resets the RX and TX physical FIFOs and all internal state machines and control registers.</p> <p>The function then enables the TX outputs, waits for a minimum of 1 millisecond, and clears all pending error bits.</p> |
| INCLUDE FILES | tekpci.h, tekpciLink.h |
| RETURNS | OK, or ERROR if an error is encountered accessing the hardware. |
| SEE ALSO | tekpciLinkFpgaLoad(), tekpciLinkFpgaLoadByDwg() |

FastLink Driver: Parameter Functions

The **tekpciLink** driver implements a set of functions which allow the user application to get and set data link parameters. Each data link channel has a separate parameter state structure.

The **tekpciLink** parameter API supports several different hardware configurations with the possibility of different types of interactions between channels. In most cases, TEK's hardware products implement independent data link channels. In cases where data link channel parameters are not independent, the parameter API will override previous settings with new settings.

For example, the FastPMC-HL266 module uses a single clock source for both the HOTLink transmit and receive circuits. When this module is being configured, the **tekpciLinkRxSetParms()** function will have the side effect of changing the previously programmed transmit frequency.

If the application requires detailed knowledge about the hardware module's interactions, the documentation for the hardware module should be consulted. To verify proper parameter settings, the user application can set all desired parameter states for all transmit and receive channels and then read and verify all parameter states. If the hardware limitations resulted in the desired parameters being inconsistent, the resulting verify operation will fail allowing the application to detect a potential problem.

Data link channel parameters are accessed using a generic **TEKPCI_LINK_PARMS** parameter data structure, which is defined in **tekpciLink.h**. The API includes functions to get and set the parameter state for receive (RX) and transmit (TX) channels.

Most of the constant information in the **TEKPCI_LINK_PARMS** data structure is determined by the hardware model information stored in the module's Serial EEPROM..

The **TEKPCI_LINK_PARMS** data structure consists of three portions. The first portion is a read-only set of fields filled in by the get functions and ignored by the set functions. The second set of fields are read-write fields filled in with the current hardware state by get functions and used to set the current hardware state by the set functions. The third set of fields are informational fields which describe the hardware state after the parameter state has changed; these fields are filled in by the set functions and are updated based on the programmed parameter set by the set functions.

The recommended approach for changing the parameter state is to use a get function to get a parameter state structure filled in with current values, modify the desired fields, and then call the appropriate set function to set the new parameter state. This approach automatically supports future driver versions.

tekpciLinkParmsDisplay()

| | |
|----------------------|---|
| NAME | <i>tekpciLinkParmsDisplay()</i> – Display the parameter state information. |
| SYNOPSIS | <pre>STATUS tekpciLinkParmsDisplay (TEKPCI_LINK_PARMS *parms, void (*handler) (void *handle, const char *str), void *handle);</pre> |
| DESCRIPTION | <p>This function returns a formatted string describing the contents of the parameter structure specified by argument <i>parms</i>. Parameter <i>handler</i> specifies a caller-defined handler. Parameter <i>handle</i> specifies a handler argument. The function calls the handler, passing <i>handle</i> and formatted string <i>str</i> as arguments.</p> <p>This function does not access the hardware, and assumes that the info structure specified by <i>parms</i> was returned by a previous call to tekpciLinkRxGetParms() or tekpciLinkTxGetParms().</p> |
| INCLUDE FILES | tekpci.h, tekpciLink.h |
| RETURNS | OK, or ERROR if an error is encountered with the <i>parms</i> data structure. |
| SEE ALSO | tekpciLinkRxGetParms(), tekpciLinkTxGetParms() |

tekpciLinkRxGetParms()

| | |
|----------------------|---|
| NAME | <i>tekpciLinkRxGetParms()</i> – Get RX parameter state. |
| SYNOPSIS | <pre>STATUS tekpciLinkRxGetParms (TEKPCI_HANDLE pci, int channel, TEKPCI_LINK_PARMS *parms);</pre> |
| DESCRIPTION | <p>This function returns the current parameter state of the receive channel specified by <i>channel</i> on the device specified by <i>pci</i>.</p> <p>The parameter state is returned in the caller-allocated structure specified by <i>parms</i>.</p> <p>The parameter state is typically determined by software data structures maintained as a part of the <i>pci</i> handle by the driver. If a process has accessed the device directly, the returned state may not be accurate.</p> |
| INCLUDE FILES | tekpci.h, tekpciLink.h |
| RETURNS | OK, or ERROR if an error is encountered. |
| SEE ALSO | tekpciLinkRxSetParms() |

tekpciLinkRxSetParms()

| | |
|----------------------|---|
| NAME | <i>tekpciLinkRxSetParms()</i> – Set RX parameter state and update device. |
| SYNOPSIS | <pre>STATUS tekpciLinkRxSetParms (TEKPCI_HANDLE pci, int channel, TEKPCI_LINK_PARMS *parms);</pre> |
| DESCRIPTION | <p>This function sets the parameter state of the receive channel specified by <i>channel</i> on the device specified by <i>pci</i>.</p> <p>The parameter state is specified by <i>parms</i>. The data structure is validated and then used to set the new parameter state. Structure members set to negative values are set to the state that exists at the time of call.</p> <p>The function minimizes hardware accesses where possible to avoid data corruption. If the new parameter state matches the old parameter state, no hardware accesses are performed. If a forced hardware update is desired, the application should call tekpciLinkFpgaReset() before calling this function.</p> <p>Updating the operating parameters will typically result in spurious data being received as the hardware is initialized to the new state. In particular, receive error status flags may be set as the hardware transitions to the new state.</p> |
| INCLUDE FILES | tekpci.h, tekpciLink.h |
| RETURNS | OK, or ERROR if an error is encountered. |
| SEE ALSO | tekpciLinkRxGetParms() |

tekpciLinkTxGetParms()

| | |
|----------------------|--|
| NAME | <i>tekpciLinkTxGetParms()</i> – Get TX parameter state. |
| SYNOPSIS | <pre>STATUS tekpciLinkTxGetParms (TEKPCI_HANDLE pci, int channel, TEKPCI_LINK_PARMS *parms);</pre> |
| DESCRIPTION | <p>This function returns the current parameter state of the transmit channel specified by <i>channel</i> on the device specified by <i>pci</i>.</p> <p>The parameter state is returned in the caller-allocated structure specified by <i>parms</i>.</p> <p>The parameter state is typically determined by software data structures maintained as a part of the <i>pci</i> handle by the driver. If a process has accessed the device directly, the returned state may not be accurate.</p> |
| INCLUDE FILES | tekpci.h, tekpciLink.h |
| RETURNS | OK, or ERROR if an error is encountered. |
| SEE ALSO | tekpciLinkTxSetParms() |

tekpciLinkTxSetParms()

| | |
|----------------------|--|
| NAME | <i>tekpciLinkTxSetParms()</i> – Update Device TX parameter state. |
| SYNOPSIS | <pre>STATUS tekpciLinkTxSetParms (TEKPCI_HANDLE pci, int channel, TEKPCI_LINK_PARMS *parms);</pre> |
| DESCRIPTION | <p>This function sets the parameter state of the transmit channel specified by <i>channel</i> on the device specified by <i>pci</i>.</p> <p>The parameter state is specified by <i>parms</i>. The data structure is validated and then used to set the new parameter state. Structure members set to negative values are set to the state that exists at the time of call.</p> <p>The function minimizes hardware accesses where possible to avoid data corruption. If the new parameter state matches the old parameter state, no hardware accesses are performed. If a forced hardware update is desired, the application should call tekpciLinkFpgaReset() before calling this function.</p> <p>Updating the operating parameters will typically result in spurious data being transmitted as the hardware is initialized to the new state.</p> |
| INCLUDE FILES | tekpci.h, tekpciLink.h |
| RETURNS | OK, or ERROR if an error is encountered. |
| SEE ALSO | tekpciLinkTxGetParms() |

FastLink Driver: Receive Functions

The **tekpciLink** Receive functions provide the interface between the application and the receive data stream from the hardware module. Using the Receive functions, applications can get and clear error status conditions and transfer data received by the module using either software memory accesses or DMA transfers.

The receive API consists of the following functions:

- **tekpciLinkRxGetErrors()** and **tekpciLinkRxClearErrors()** allow the application to monitor error status conditions and clear error flags.
- **tekpciLinkRxGetPtr()** and **tekpciLinkRxGetCount()** allow the application to access a direct FIFO interface to the receive data stream.
- **tekpciLinkRxDmaQueue()** and **tekpciLinkRxDmaQueueChannel()** allow the application to perform DMA transfers from the receive data stream into user-specified buffer memory.

tekpciLinkRxClearErrors()

| | |
|----------------------|---|
| NAME | <i>tekpciLinkRxClearErrors()</i> – Reset RX error conditions. |
| SYNOPSIS | <pre>STATUS tekpciLinkRxClearErrors (TEKPCI_HANDLE pci, int channel, int errors);</pre> |
| DESCRIPTION | <p>This function resets error status bits for the receive channel specified by <i>channel</i> on the device specified by <i>pci</i>.</p> <p>Parameter <i>errors</i> is a bitmask specifying which status bits to reset. If zero, all status bits are reset. FPMC devices typically implement errors as “sticky” status bits, which are set when an error occurs and remain set until reset under software control.</p> <p>Typically, applications call tekpciLinkRxGetErrors() to detect and handle error conditions, then call this function to reset errors.</p> |
| INCLUDE FILES | tekpci.h, tekpciLink.h |
| RETURNS | OK, or ERROR if an error is encountered accessing the hardware. |
| SEE ALSO | tekpciLinkRxGetErrors() |

tekpciLinkRxGetCount()

| | |
|----------------------|---|
| NAME | <i>tekpciLinkRxGetCount()</i> – Get current RX FIFO count. |
| SYNOPSIS | <pre>STATUS tekpciLinkRxGetCount (TEKPCI_HANDLE pci, int channel, int *count);</pre> |
| DESCRIPTION | <p>This function returns a count of 32-bit words in the receive FIFO on the receive channel specified by <i>channel</i> on the device specified by <i>pci</i>. The count is returned in the caller-allocated location specified by <i>count</i>.</p> <p>After calling this function, the caller is guaranteed that at least <i>count</i> words may be read using the FIFO pointer returned by <i>tekpciLinkRxGetPtr()</i>.</p> <p>This function supports raw I/O streams. Devices implementing packet or message based I/O should generally not be interrogated by this function. In addition, the receive FIFO interface should not be used on channels using DMA.</p> |
| INCLUDE FILES | <i>tekpci.h, tekpciLink.h</i> |
| RETURNS | OK, or ERROR if an error is encountered accessing the hardware. |
| SEE ALSO | <i>tekpciLinkRxGetPtr()</i> |

tekpciLinkRxGetErrors()

| | |
|----------------------|--|
| NAME | <i>tekpciLinkRxGetErrors()</i> – Get RX error conditions. |
| SYNOPSIS | <pre>STATUS tekpciLinkRxGetErrors (TEKPCI_HANDLE pci, int channel, int *errors);</pre> |
| DESCRIPTION | <p>This function reads the current error status on the receive channel specified by <i>channel</i> on the device specified by <i>pci</i>. Error bits are returned in the caller-allocated location specified by <i>errors</i>.</p> <p>Devices typically implement receive errors as “sticky” status bits which are set when an error occurs and reset under software control. Errors reported by this function may be cleared by calling tekpciLinkRxClearErrors().</p> <p>Definitions of specific error conditions are device-dependent. Common error types defined for all devices include</p> <pre> TEKPCI_LINK_ERROR_FIFO Low-level receive FIFO overflow.</pre> <pre> TEKPCI_LINK_ERROR_DATA. Data quality error at the input FIFO.</pre> <p>Device-specific error types may also be defined.</p> |
| INCLUDE FILES | tekpci.h, tekpciLink.h |
| RETURNS | OK, or ERROR if an error is encountered accessing the hardware. |
| SEE ALSO | tekpciLinkRxClearErrors() |

tekpciLinkRxGetPtr()

| | |
|----------------------|---|
| NAME | <i>tekpciLinkRxGetPtr()</i> – Get pointer to RX FIFO data register. |
| SYNOPSIS | <pre>STATUS tekpciLinkRxGetPtr (TEKPCI_HANDLE pci, int channel, volatile WORD32 **ptr);</pre> |
| DESCRIPTION | <p>This function returns an address pointer to the receive FIFO data register for the receive channel specified by <i>channel</i> on the device specified by <i>pci</i>. A pointer that is valid in the caller's address space is returned in the caller-allocated location specified by <i>ptr</i>.</p> <p>The returned pointer may be used to read data from the FIFO. Calling tekpciLinkRxGetCount() returns the number of words that may be safely read by the calling process using this pointer.</p> <p>This function supports raw I/O streams. Devices implementing packet or message based I/O should generally not be interrogated by this function. In addition, the receive FIFO interface should not be used on channels using DMA.</p> <p>The receive FIFO address is fixed once the hardware module has been loaded, the FPGA has been initialized, and tekpciLinkRxSetParms() is called.</p> |
| INCLUDE FILES | tekpci.h, tekpciLink.h |
| RETURNS | OK, or ERROR if an error is encountered accessing the hardware. |
| SEE ALSO | tekpciLinkRxGetCount() |

tekpciLinkRxDmaQueue ()

| | |
|----------------------|---|
| NAME | <i>tekpciLinkRxDmaQueue ()</i> – Queue DMA transfer. |
| SYNOPSIS | <pre>STATUS tekpciLinkRxDmaQueue (TEKPCI_HANDLE pci, int channel, unsigned long pciaddr, int length, void (*callback) (void *handle), void *handle);</pre> |
| DESCRIPTION | <p>This function queues a DMA transfer on the receive channel specified by <i>channel</i> on the device specified by <i>pci</i>. The input data is written to <i>pciaddr</i>, which must be a valid address in PCI address space. A maximum of <i>length</i> bytes are transferred.</p> <p>Parameter <i>callback</i> specifies a caller-defined callback function, which is called when the transfer completes. Parameter <i>handle</i> is passed to the callback function as an argument.</p> <p>For raw I/O streams, transfers always equal <i>length</i> bytes. For formatted I/O streams, the hardware may optionally terminate a DMA transfer before length bytes are transferred. Transfers never exceed <i>length</i> bytes.</p> |
| INCLUDE FILES | tekpci.h, tekpciLink.h |
| RETURNS | OK, or ERROR if an error is encountered accessing the hardware. |
| SEE ALSO | tekpciLinkRxDmaQueueChannel () |

tekpciLinkRxDmaQueueChannel ()

| | |
|----------------------|---|
| NAME | <i>tekpciLinkRxDmaQueueChannel ()</i> – Queue DMA transfer. |
| SYNOPSIS | <pre>STATUS tekpciLinkRxDmaQueueChannel (TEKPCI_HANDLE pci, int channel, int dmachannel, unsigned long pciaddr, int length, void (*callback) (void *handle), void *handle);</pre> |
| DESCRIPTION | <p>This function queues a DMA transfer on the receive channel specified by <i>channel</i> on the device specified by <i>pci</i>. Parameter <i>dmachannel</i> specifies the DMA channel. The input data is written to <i>pciaddr</i>, which must be a valid address in PCI address space. A maximum of <i>length</i> bytes are transferred.</p> <p>Parameter <i>callback</i> specifies a caller-defined callback function, which is called when the transfer completes. Parameter <i>handle</i> is passed to the callback function as an argument.</p> <p>For raw I/O streams, transfers always equal <i>length</i> bytes. For formatted I/O streams, the hardware may optionally terminate a DMA transfer before <i>length</i> bytes are transferred. Transfers never exceed <i>length</i> bytes.</p> |
| INCLUDE FILES | tekpci.h, tekpciLink.h |
| RETURNS | OK, or ERROR if an error is encountered accessing the hardware. |
| SEE ALSO | tekpciLinkRxDmaQueue () |

FastLink Driver: Transmit Functions

The **tekpciLink** transmit functions provide the interface between the application and the transmit data stream on the FPMC device. Using the transmit functions, the application can get and clear error status conditions and transfer data to the module using either software memory accesses or DMA transfers.

The Transmit API consists of the following functions:

- **tekpciLinkTxGetErrors()** and **tekpciLinkTxClearErrors()** allow the user application to monitor error status conditions and clear error flags.
- **tekpciLinkTxGetPtr()**, **tekpciLinkTxGetCount()** and **tekpciLinkTxGetFree()** allow the user application to access a direct FIFO interface to the transmit data stream.
- **tekpciLinkTxDmaQueue()** and **tekpciLinkTxDmaQueueChannel()** allow the application to perform DMA transfers from the receive data stream into user-specified buffer memory.

tekpciLinkTxClearErrors()

| | |
|----------------------|--|
| NAME | <i>tekpciLinkTxClearErrors()</i> – Reset TX error conditions. |
| SYNOPSIS | <pre>STATUS tekpciLinkTxClearErrors (TEKPCI_HANDLE pci, int channel, int errors);</pre> |
| DESCRIPTION | <p>This function resets error status bits for the transmit channel specified by <i>channel</i> on the device specified by <i>pci</i>.</p> <p>Parameter <i>errors</i> is a bitmask specifying which status bits to reset. If zero, all status bits are reset. FPMC devices typically implement errors as “sticky” status bits, which are set when an error occurs and remain set until reset under software control.</p> <p>Typically, applications call tekpciLinkTxGetErrors() to detect and handle error conditions, then call this function to reset errors.</p> |
| INCLUDE FILES | tekpci.h, tekpciLink.h |
| RETURNS | OK, or ERROR if an error is encountered accessing the hardware. |
| SEE ALSO | tekpciLinkTxGetErrors() |

tekpciLinkTxGetCount()

| | |
|----------------------|---|
| NAME | <i>tekpciLinkTxGetCount()</i> – Get current TX FIFO count. |
| SYNOPSIS | <pre>STATUS tekpciLinkTxGetCount (TEKPCI_HANDLE pci, int channel, int *count);</pre> |
| DESCRIPTION | <p>This function returns a count of 32-bit words in the transmit FIFO on the channel specified by <i>channel</i> on the device specified by <i>pci</i>. The count is returned in the caller-allocated location specified by <i>count</i>.</p> <p>This function is provided for diagnostic purposes, and supports raw I/O streams. Devices implementing packet or message based I/O should generally not be interrogated by this function. In addition, the FIFO interface should not be used on channels using DMA.</p> |
| INCLUDE FILES | tekpci.h, tekpciLink.h |
| RETURNS | OK, or ERROR if an error is encountered accessing the hardware. |
| SEE ALSO | tekpciLinkTxGetFree(), tekpciLinkTxGetPtr() |

tekpciLinkTxGetErrors()

| | |
|----------------------|---|
| NAME | <i>tekpciLinkTxGetErrors()</i> – Get TX error conditions. |
| SYNOPSIS | <pre>STATUS tekpciLinkTxGetErrors (TEKPCI_HANDLE pci, int channel, int *errors);</pre> |
| DESCRIPTION | <p>This function reads the current error status on the channel specified by <i>channel</i> on the device specified by <i>pci</i>. Error bits are returned in the caller-allocated location specified by <i>errors</i>.</p> <p>Devices typically implement transmit errors as “sticky” status bits which are set when an error occurs and reset under software control. Errors reported by this function may be cleared by calling tekpciLinkTxClearErrors().</p> <p>Definitions of specific error conditions are device-dependent. Common error types defined for all transmit devices include</p> <pre>TEKPCI_LINK_ERROR_FIFO</pre> <p>Low-level transmit FIFO overflow.</p> <p>Device-specific error types may also be defined.</p> |
| INCLUDE FILES | tekpci.h, tekpciLink.h |
| RETURNS | OK, or ERROR if an error is encountered accessing the hardware. |
| SEE ALSO | tekpciLinkTxClearErrors() |

tekpciLinkTxGetFree()

| | |
|----------------------|---|
| NAME | <i>tekpciLinkTxGetFree()</i> – Get current TX FIFO free space. |
| SYNOPSIS | <pre>STATUS tekpciLinkTxGetFree (TEKPCI_HANDLE pci, int channel, int *count);</pre> |
| DESCRIPTION | <p>This function returns a count of free 32-bit storage locations in the transmit FIFO on the channel specified by <i>channel</i> on the device specified by <i>pci</i>. The count is returned in the caller-allocated location specified by <i>count</i>.</p> <p>After calling this function, the caller is guaranteed that at least <i>count</i> words may be written using the FIFO pointer returned by tekpciLinkTxGetPtr().</p> <p>This function supports raw I/O streams. Devices implementing packet or message based I/O should generally not be interrogated by this function. In addition, the receive FIFO interface should not be used on channels using DMA.</p> |
| INCLUDE FILES | tekpci.h, tekpciLink.h |
| RETURNS | OK, or ERROR if an error is encountered accessing the hardware. |
| SEE ALSO | tekpciLinkTxGetPtr() |

tekpciLinkTxGetPtr()

| | |
|----------------------|---|
| NAME | <i>tekpciLinkTxGetPtr()</i> – Get a pointer to the TX FIFO data register. |
| SYNOPSIS | <pre>STATUS tekpciLinkTxGetPtr (TEKPCI_HANDLE pci, int channel, volatile WORD32 **ptr);</pre> |
| DESCRIPTION | <p>This function returns an address pointer to the transmit FIFO data register for the channel specified by <i>channel</i> on the device specified by <i>pci</i>. A pointer that is valid in the caller's address space is returned in the caller-allocated location specified by <i>ptr</i>.</p> <p>The returned pointer may be used to write data to the transmit FIFO. Calling tekpciLinkTxGetCount() returns the number of words that may be safely written by the calling process using this pointer.</p> <p>This function supports raw I/O streams. Devices implementing packet or message based I/O should generally not be interrogated by this function. In addition, the receive FIFO interface should not be used on channels using DMA.</p> <p>The FIFO address is fixed once the hardware module has been loaded, the FPGA has been initialized, and tekpciLinkTxSetParms() is called.</p> |
| INCLUDE FILES | tekpci.h, tekpciLink.h |
| RETURNS | OK, or ERROR if an error is encountered accessing the hardware. |
| SEE ALSO | tekpciLinkTxGetFree() |

tekpciLinkTxDmaQueue ()

| | |
|----------------------|---|
| NAME | <i>tekpciLinkTxDmaQueue ()</i> – Queue DMA transfer. |
| SYNOPSIS | <pre>STATUS tekpciLinkTxDmaQueue (TEKPCI_HANDLE pci, int channel, unsigned long pciaddr, int length, void (*callback) (void *handle), void *handle);</pre> |
| DESCRIPTION | <p>This function queues a DMA transfer on the transmit channel specified by <i>channel</i> on the device specified by <i>pci</i>. The output data is read from <i>pciaddr</i>, which must be a valid address in PCI address space. A maximum of <i>length</i> bytes are transferred.</p> <p>Parameter <i>callback</i> specifies a caller-defined callback function, which is called when the transfer completes. Parameter <i>handle</i> is passed to the callback function as an argument.</p> <p>For raw I/O streams, transfers always equal <i>length</i> bytes. For formatted I/O streams, the hardware may optionally terminate a DMA transfer before <i>length</i> bytes are transferred. Transfers never exceed <i>length</i> bytes.</p> |
| INCLUDE FILES | tekpci.h, tekpciLink.h |
| RETURNS | OK, or ERROR if an error is encountered accessing the hardware. |
| SEE ALSO | tekpciLinkTxDmaQueueChannel () |

tekpciLinkTxDmaQueueChannel ()

| | |
|----------------------|--|
| NAME | <i>tekpciLinkTxDmaQueueChannel ()</i> – Queue DMA transfer. |
| SYNOPSIS | <pre>STATUS tekpciLinkTxDmaQueueChannel (TEKPCI_HANDLE pci, int channel, int dmachannel, unsigned long pciaddr, int length, void (*callback) (void *handle), void *handle);</pre> |
| DESCRIPTION | <p>This function queues a DMA transfer on the transmit channel specified by <i>channel</i> on the device specified by <i>pci</i>. Parameter <i>dmachannel</i> specifies the DMA channel. The input data is written to <i>pciaddr</i>, which must be a valid address in PCI address space. A maximum of <i>length</i> bytes are transferred.</p> <p>Parameter <i>callback</i> specifies a caller-defined callback function, which is called when the transfer completes. Parameter <i>handle</i> is passed to the callback function as an argument.</p> <p>For raw I/O streams, transfers always equal <i>length</i> bytes. For formatted I/O streams, the hardware may optionally terminate a DMA transfer before <i>length</i> bytes are transferred. Transfers never exceed <i>length</i> bytes.</p> |
| INCLUDE FILES | tekpci.h, tekpciLink.h |
| RETURNS | OK, or ERROR if an error is encountered accessing the hardware. |
| SEE ALSO | tekpciLinkTxDmaQueue () |

FastLink Driver: Info Functions

The **tekpciLinkInfo** functions allow applications to read and display information about the type of hardware module installed. These functions work with a **TEKPCI_LINK_INFO** data structure, defined in **tekpciLink.h**. Functions include **tekpciLinkInfoGet()**, which returns an information data structure, and **tekpciLinkInfoDisplay()**, which compiles a formatted string from the information data structure.

tekpciLinkInfoDisplay()

| | |
|----------------------|--|
| NAME | <i>tekpciLinkInfoDisplay()</i> – Display the module information. |
| SYNOPSIS | <pre>STATUS tekpciLinkInfoDisplay (TEKPCI_LINK_INFO *infop, void (*handler) (void *arg, const char *str), void *arg);</pre> |
| DESCRIPTION | This function reads a device information structure and compiles a formatted string for output. Parameter <i>infop</i> is an information structure returned by tekpciLinkInfoGet() . Argument <i>handler</i> specifies a caller-defined handler. Argument <i>arg</i> specifies a handler argument. The function calls the handler, passing <i>arg</i> and formatted string <i>str</i> as arguments. |
| INCLUDE FILES | tekpci.h, tekpciLink.h |
| RETURNS | OK, or ERROR if an error is encountered accessing the hardware. |
| SEE ALSO | tekpciLinkInfoGet() |

tekpciLinkInfoGet()

| | |
|----------------------|---|
| NAME | <i>tekpciLinkInfoGet()</i> – Read device-specific info. |
| SYNOPSIS | <pre>STATUS tekpciLinkInfoGet (TEKPCI_HANDLE pci, TEKPCI_LINK_INFO *infop);</pre> |
| DESCRIPTION | <p>This function reads device-specific information from the configuration PROM of the device specified by <i>pci</i>. The information is returned in the caller-allocated structure specified by <i>infop</i>.</p> <p>If the module has not been initialized, the call fails and the contents of <i>infop</i> is undefined.</p> |
| INCLUDE FILES | tekpci.h, tekpciLink.h |
| RETURNS | OK, or ERROR if an error is encountered accessing the hardware. |
| SEE ALSO | tekpciLinkInfoDisplay() |

FastLink Driver: Test Functions

The driver implements Built-In-Test functions, which support module confidence testing. The test functions verify basic operation of the hardware module. A typical confidence test sequence consists of the following steps:

- Once the module has been attached to a driver, verify data bus functionality using the **tekpciLinkTestDatabus()** function.
- If internal manufacturing tests are desired, call the **tekpciLinkTestClock()** and **tekpciLinkTestMemory()** functions to perform internal testing of the module.
- If external LED display and loopback tests are desired, call the **tekpciLinkTestLed()** and **tekpciLinkTestLoopback()** functions.

tekpciLinkTestDatabus()

| | |
|----------------------|---|
| NAME | <i>tekpciLinkTestDatabus()</i> – Test local data bus. |
| SYNOPSIS | <pre>STATUS tekpciLinkTestDatabus (TEKPCI_HANDLE pci, VERBOSE level, void (*output) (void *handle, const char *str), void *handle);</pre> |
| DESCRIPTION | <p>This function first performs a test of FPMC module specified by <i>pci</i> hardware module, then tests the local data bus.</p> <p>Argument <i>level</i> specifies the level of status information returned by the function. Defined values include:</p> <ul style="list-style-type: none">VERBOSE_NONE No status output. Parameters <i>output</i> and <i>handle</i> may be NULL.VERBOSE_ERROR Generate output for errors only.VERBOSE_ALL Generate test output as the tests progress. <p>The function invokes the caller-defined handler <i>output</i>, passing <i>handle</i> as argument <i>handle</i> and the status display string, if any, as argument <i>str</i>.</p> |
| INCLUDE FILES | tekpci.h, tekpciLink.h |
| RETURNS | OK, or ERROR if an error is encountered. |

tekpciLinkTestClock()

| | |
|----------------------|--|
| NAME | <i>tekpciLinkTestClock()</i> – Test the local clock synthesizer and oscillator functions. |
| SYNOPSIS | <pre> STATUS tekpciLinkTestClock (TEKPCI_HANDLE pci, VERBOSE level, int channel, void (*output) (void *handle, const char *str), void *handle); </pre> |
| DESCRIPTION | <p>This function tests a channel's clock generation function. Parameter <i>pci</i> specifies the device and parameter <i>channel</i> specifies the channel. Programmable onboard oscillators and clock synthesizers are exercised. If the target device includes local bus clock capabilities, those capabilities are also exercised.</p> <p>If the device uses customizable FPGA programs, this function assumes that a standard FPGA is loaded. Specific test functions are device-dependent. After this test completes, the onboard clock generator state is undefined.</p> <p>Argument <i>level</i> specifies the level of status information returned by the function. Defined values include:</p> <ul style="list-style-type: none"> VERBOSE_NONE No status output. Parameters <i>output</i> and <i>handle</i> may be NULL. VERBOSE_ERROR Generate output for errors only. VERBOSE_ALL Generate test output as the tests progress. <p>The function invokes the caller-defined handler <i>output</i>, passing <i>handle</i> as argument <i>handle</i> and the status display string, if any, as argument <i>str</i>.</p> |
| INCLUDE FILES | tekpci.h, tekpciLink.h |
| RETURNS | OK, or ERROR if an error is encountered. |

tekpciLinkTestLed()

| | |
|----------------------|---|
| NAME | <i>tekpciLinkTestLed()</i> – Test the onboard LED indicators. |
| SYNOPSIS | <pre> STATUS tekpciLinkTestLed (TEKPCI_HANDLE pci, VERBOSE level, void (*delay) (void), void (*output) (void *handle, const char *str), void *handle); </pre> |
| DESCRIPTION | <p>This function tests any LEDs present on the device specified by <i>pci</i>.</p> <p>For each LED, the output state is changed from off to each possible output color in turn (typically green and then red) with a delay between each state. The specific LED sequence is module dependent.</p> <p>To implement the delay between LED tests, the function invokes the caller-defined handler <i>delay</i>, which is assumed to provide a one second delay.</p> <p>Argument <i>level</i> specifies the level of status information returned by the function. Defined values include:</p> <pre> VERBOSE_NONE No status output. Parameters <i>output</i> and <i>handle</i> may be NULL. VERBOSE_ERROR Generate output for errors. </pre> <p>The function invokes the caller-defined handler <i>output</i>, passing <i>handle</i> as argument <i>handle</i> and the status display string, if any, as argument <i>str</i>.</p> |
| INCLUDE FILES | tekpci.h, tekpciLink.h |
| RETURNS | OK, or ERROR if an error is encountered accessing the hardware. |

tekpciLinkTestMemory()

| | |
|----------------------|--|
| NAME | <i>tekpciLinkTestMemory()</i> – Test onboard memory resources. |
| SYNOPSIS | <pre>STATUS tekpciLinkTestMemory (TEKPCI_HANDLE pci, VERBOSE level, void (*output) (void *handle, const char *str), void *handle);</pre> |
| DESCRIPTION | <p>This function tests any accessible memory resources on the device specified by <i>pci</i>. The memory devices tested are device-dependent.</p> <p>Argument <i>level</i> specifies the level of status information returned by the function. Defined values include:</p> <ul style="list-style-type: none">VERBOSE_NONE No status output. Parameters <i>output</i> and <i>handle</i> may be NULL.VERBOSE_ERROR Generate output for errors.VERBOSE_ALL Generate test output as the tests progress. <p>The function invokes the caller-defined handler <i>output</i>, passing <i>handle</i> as argument <i>handle</i> and the status display string, if any, as argument <i>str</i>.</p> |
| INCLUDE FILES | tekpci.h, tekpciLink.h |
| RETURNS | OK, or ERROR if an error is encountered accessing the hardware. |

tekpciLinkTestLoopback()

| | |
|----------------------|--|
| NAME | <i>tekpciLinkTestLoopback()</i> – Channel Loopback Test. |
| SYNOPSIS | <pre> STATUS tekpciLinkTestLoopback (TEKPCI_HANDLE pci, VERBOSE level, int channel, int input, void (*output) (void *handle, const char *str), void *handle); </pre> |
| DESCRIPTION | <p>This function tests the receive channel specified by <i>channel</i> using the device's most appropriate transmit channel. Parameter <i>pci</i> specifies the device. The specific tests is device dependent. If the device supports an internal loopback feature, the internal loopback test is executed followed by the external loopback test. This function assumes the necessary external loopback connectors are installed.</p> <p>Parameter <i>input</i> specifies the input source for the channel and is identical to the <i>inputSource</i> parameter in <i>tekpciLinkRxGetParms()</i>.</p> <p>Argument <i>level</i> specifies the level of status information returned by the function. Defined values include:</p> <ul style="list-style-type: none"> VERBOSE_NONE No status output. Parameters <i>output</i> and <i>handle</i> may be NULL. VERBOSE_ERROR Generate output for errors. VERBOSE_ALL Generate test output as the tests progress. <p>The function invokes the caller-defined handler <i>output</i>, passing <i>handle</i> as argument <i>handle</i> and the status display string, if any, as argument <i>str</i>.</p> |
| INCLUDE FILES | <i>tekpci.h, tekpciLink.h</i> |
| RETURNS | OK, or ERROR if an error is encountered. |