Functional Description for

DPIO2 / DPIO2-66

FPDP / FPDP II
Digital Parallel Input Output
PMC Module

Version 2.0 - Valid for DPIO2, PCB rev. B/C and DPIO2-66, PCB rev. A

VMETRO

The information in this Functional Description may be changed without further notification, and should not be construed as a commitment by VMETRO. While reasonable precautions have been taken, VMETRO assumes no responsibility for errors that may appear in this document.

VMETRO

Limited Liability

VMETRO does not assume any liability arising out of the application or use of any product described herein; neither does it convey any license under its patent rights nor the rights of others. VMETRO products are not designed, intended, or authorized for use as components in systems intended to support or sustain life, or for any application in which failure of the VMETRO product could create a situation where personal injury or death may occur. If the Buyer purchase or use VMETRO products for any such unintended or unauthorized application, the Buyer is to indemnify and hold VMETRO and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses. This includes reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that VMETRO was negligent regarding the design or manufacture of the part.

USA: VMETRO, Inc.
1880 Dairy Ashford, Suite 400
Houston, TX 77077, USA
Tel.: (281) 584-0728
Fax: (281) 584-9034
E-mail: info@vmetro.com

Europe, Asia: VMETRO asa
Brynsveien 5
N-0667 OSLO, Norway
Tel.: +47 2210 6090
Fax: +47 2210 6202
E-mail: info@vmetro.no

Web Site: http://www.vmetro.com/

Contents

Introduction

About this Document

This document describes the different features and options DPIO2 and DPIO2-66 have, and discusses considerations to be taken when using it. The document presumes that a SW driver is used when operating DPIO2/DPIO2-66. In addition, the document includes references to SW calls. The adequate SW call for the function referred to, is found in the **User's Manual for DPIO2 Device Driver**. All references to this manual will appear as this:

Use the SW command: **DPIO2_CMD_RES2_OUTPUT_VALUE_SET**

Related Documents

To get a full understanding of how DPIO2/DPIO2-66 works, the following literature should be obtained:

[1] VITA (Tel: 1-602-951-8866) ***http://www.vita.com***	Front Panel Data Port Protocol and Mechanical Specifications (VITA 17-1998), February 11, 1999.
[2] PCI Special Interest Group (Tel: 1-503-797-4207) ***http://www.pcisig.com/***	PCI Local Bus Specification Revision 2.1
[3] IEEE (Fax 1-732-981-9334) ***http://www.ieee.com***	Draft Standard for a Common Mezzanine Card Family: CMC

[4] IEEE (Fax 1-732-981-9334) ***http://www.ieee.com***	Draft Standard Physical and Environmental Layers for PCI Mezzanine Cards: PMC
[5] VMETRO	User's Manual for DPIO2 Device Driver - VxWorks 5.4 - Windows NT/2000/XP - MC/OS
[6] VMETRO	User's Manual for DPIO2
[7] VMETRO	User's Manual for DPIO2-66

Technical Support

VMETRO provides technical documentation with all of our products. This functional description describes the technology and its performance characteristics.

Although, we have attempted to make this document comprehensive, there may be specific problems or issues it does not satisfactorily cover. If you have any questions, you can phone us, or send us an e-mail. Our goal is to offer a combination of products and services that provide complete, easy-to-use solutions for your application.

USA:	VMETRO Inc.	Phone: (281) 584-0728 E-mail: support@vmetro.com
Europe, Asia:	VMETRO asa	Phone: +47 2210 6090 E-mail: support@vmetro.no

Product Overview

DPIO2 / DPIO2-66 – PCI Mezzanine Card

DPIO2 / DPIO2-66 are VMETRO's second-generation high-speed digital parallel input/output PMC modules, and are suitable for advanced parallel I/O. DPIO2 / DPIO2-66 are bidirectional and can be operated as either input or output modules. DPIO2 / DPIO2-66 are available with drivers for VxWorks, MC/OS™ (Mercury) and Windows 2000/NT/XP. Driver source is available for the purpose of porting the driver to other platforms.

DPIO2 / DPIO2-66 combine:

- Digital Parallel Input or Output port on one end.
- 64-bit PCI bus master/slave interface with an advanced DMA controller on the other end.
- A large FIFO in between (DPIO2 can be delivered with two FIFO sizes, either 32K or 128K 32-bit words, DPIO2-66 uses 128K size FIFO only).

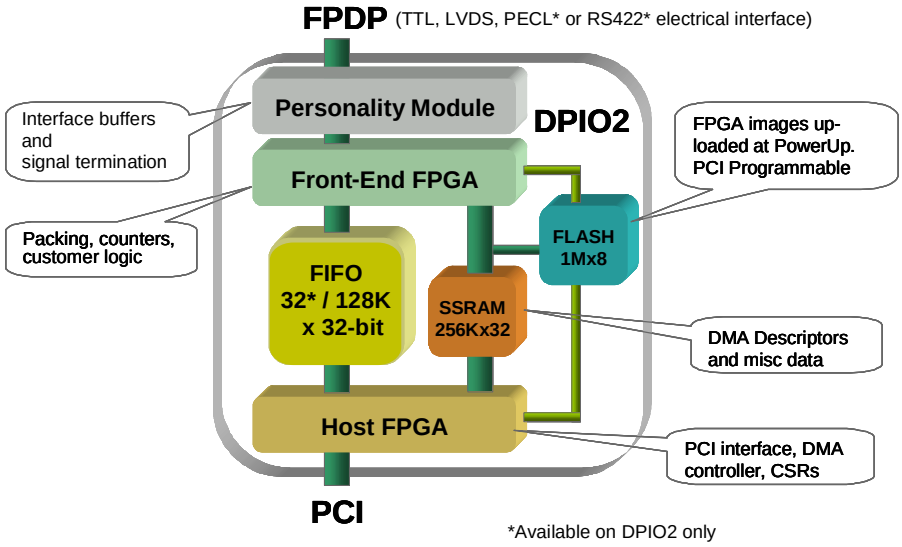

Figure 1. DPIO2 / DPIO2-66 – PCI Mezzanine Card

DPIO2 / DPIO2-66 are designed for digital data acquisition or generation of data rates up to 100 Msamples/sec or 400 MB/s. The advanced capability of the linked list DMA controller combined with the I/O handling of the front-end FPGA, manages most data frame formats used in the data acquisition and signal processing environments. They support FPDP II in addition to the original FPDP features. The speed is achieved by using a clock rate of 50 MHz, sampling data on both edges, which is part of the FPDP II specification.

The modules support the original FPDP and the basic board comes with the standard FPDP TTL signaling. Separate models also support other signal standards such as LVDS (both DPIO2 and DPIO2-66), PECL and RS422/485 (DPIO2 only).

The main difference between DPIO2 and DPIO2-66 is that DPIO2 may operate at PCI clock frequencies up to 33 MHz while DPIO2-66 may operate at PCI clock frequencies up to 66 MHz. In addition the DPIO2 is universal with respect to signaling voltages at the PCI bus, i. e. it may be used in 3.3V environments as well as in 5V systems. DPIO2-66 can only be used in 3.3V PCI systems.

DPIO2 / DPIO2-66 may be used in “regular” PCI systems, such as desktop PCs and servers by using PMC-to-PCI adapters.
Note: Different PCI adapters are needed for DPIO2 and DPIO2-66. Contact Vmetro for information about the adapters.

Features

Interface

- 64-bit 33 MHz (DPIO2) or 66MHz (DPIO2-66) PCI Bus
- Advanced DMA (Linked list)
- FIFO level triggered DMA transfer
- End of transfer capability, optional jump to the next DMA descriptor
- All byte swap operations: Byte, Word and Double Word

FIFO

- 32K words (128 KB) or 128K words (512 KB) deep
  (128K words only available for DPIO2-66)
- On-board DMA or Remote PCI Master Data transfer
- PCI interrupt on FIFO levels

Bidirectional Design

- Software controlled input and output mode selection

Front-end Processing

- 1x32, 2x16 or 3x10 bits packing options as standard. Others on request
- Advanced synchronization control features
- Sample Skip & Store counters

Clocking Options

- Software controlled input and output modules that can use internal or external clock
- Programmable clock generator from 500 KHz to 50 MHz

Front Panel Interface Options

- TTL (Standard FPDP)
- RS422 (DPIO2 only)
- LVDS (Low Voltage Differential Signaling)
- PECL (Positive ECL) (DPIO2 only)

FPDP Interface

- FPDP/FPDP II compatible 32-bits parallel data I/O
- Standard FPDP up to 160 MB/s
- FPDP II up to 400 MB/s
- Supports unframed, single or repeating, fixed or dynamic size framed data
- PCI Interrupt on Suspend assertion
- Software controlled or FIFO high watermark SUSPEND assertion
- LED indicators for SUSPEND, STROBE, NRDY, DVALID, and SYNC

Software Drivers

- VxWorks™
- Windows 2000/NT/XP
- MC/OS™ (Mercury)

Miscellaneous

- Rugged and extended temperature versions available

Applications Using DPIO2 / DPIO2-66

- Radar and sonar
- High-speed data acquisition
- Digital receivers
- Spectrum and transient analysis
- Test equipment

PCI Interface

About the PCI Interface

The PCI Interface chip provides a 64-bit PCI bus master/slave interface with a linked list DMA engine. This interface is based on Xilinx FPGA with an integrated PCI core. DPIO2 supports both 5V and 3.3V signaling on PCI. DPIO2-66 supports only 3.3V signaling on PCI.

Maximum sustained PCI data rate is measured at 255.6 MB/s using DMA from DPIO2 to PCI. This represents a 96% usage of the available bandwidth of the 64-bit 33 MHz PCI bus.

For DPIO2-66, maximum sustained PCI data rate is 400 MB/s.

PCI Address Map

DPIO2 / DPIO2-66 use two slave images in PCI address space, and each slave image is 8 MB in size. Configuration Space is used to set up Slave Image 0 and 1 in the PCI memory space.

	Description	***Size***
PCI Configuration Registers	These registers are used to configure the PCI device according to PCI specifications. DPIO2 / DPIO2-66 will respond to PCI Configuration Cycles when IDSEL is asserted.	256 Bytes

Slave Image 0	FLASH, SRAM and all the CSR registers are accessed through Slave Image 0. (BAR0)	8MB
Slave Image 1	This address space is used to access the FIFO directly from a host CPU. This address space is not pre-fetchable. (BAR1)	8MB

Table 1.PCI Address Map.

BAR0 +	***Device***	***Size***
0x00 0000	Control and status registers located in the PCI interface chip; containing DMA registers, interrupt registers and miscellaneous control and status registers.	1MB
0x10 0000	Control and status registers located in the front-end interface chip, containing registers controlling the front panel port options.	1MB
0x20 0000	1 MB SRAM for DMA linked list information. (1 MB reserved for future use)	2MB
0x40 0000	FLASH for FPGA images. Note that the FLASH is only 8 bits wide, and the 24 most significant bits will be ignored.	4MB

Table 2. Slave Image 0

NOTE The addresses in BAR0 and BAR1 are only valid in the PCI domain and not necessarily the address at which the user's software will access DPIO2 / DPIO2-66. Some regions of PCI address space are addressed through address translation windows, which may not always present the same CPU address for consecutive accesses to the same PCI address.

NOTE Slave Image 1 is not prefetchable. This is because when reading the FIFO, the data cannot be read again. When setting the DPIO2 / DPIO2-66 up to be accessed through a PCI-to-PCI bridge, the bridge must also be set up to map Slave Image 1 window as non prefetchable. If this is not done, this can result in loss of data.

When the DPIO2 / DPIO2-66 are operating in output mode, data cannot be read out of the FIFO from the PCI bus. If a host tries to read the FIFO even if DPIO2 / DPIO2-66 are operating in output mode, DPIO2 / DPIO2-66 will return dummy data.

When the DPIO2 / DPIO2-66 are operating in input mode, data can not be written into the FIFO. If a host writes to the FIFO even if DPIO2 / DPIO2-66 are operating in input mode, DPIO2 / DPIO2-66 will ignore the data.

It is possible to write data into DPIO2 / DPIO2-66 when it is configured as an output module, and then configured as an input module, and read the same data out again. This can be done as long as the front-end interface has not been enabled to send/receive data.

PCI Configuration Space

The figure below illustrates the PCI Configuration space, and some of the default values of DPIO2.

Device ID = DD11h for DPIO2 = DD12h for DPIO2-66		Vendor ID = 129Ah		00h
Status = 0020h		Command = 0080h		04h
Class Code = 110000h			RevisionID = 00h	08h
BIST	Header Type	Latency Timer	Cache Line Size	0Ch
BAR0				10h
BAR1				14h
Base Address Registers Reserved				18h 1Ch 20h 24h
Cardbus CIS Pointer				28h
Subsystem ID = DD10h		Subsystems Vendor ID = 129Ah		2Ch
Expansion ROM Base Address				30h
Reserved				34h
Reserved				38h
Max_Lat = 00h	Min_Gnt = 00h	Interrupt Pin = 01h	Interrupt Line = FFh	3Ch

Figure 2. Configuration Space Header

Latency Timer

The driver sets the latency timer to 0xFF by default. DPIO2 / DPIO2-66 use a large amount of the PCI bandwidth when DMA is running. If shorter bursts are needed to let other masters get more access to the PCI bus, write to the latency counter with a lower value. To do this, use the SW command:

DPIO2_CMD_LATENCY_TIMER_SET

Interrupts to PCI

For maximum flexibility and throughput, DPIO2 / DPIO2-66 can be programmed to generate four types of interrupts to PCI. These types are:

- End of DMA Transfer (any DMA in linked list can give interrupt)

- FIFO state: Not Empty, Empty, Almost Empty, Half Full, Almost Full, Full and Overflow
- User Input Signal asserted: PIO1, PIO2, SYNC, SUSPEND (level sensitive)
- PCI Bus error

To enable a desired interrupt, use the SW command:

DPIO2_CMD_INTERRUPT_ENABLE

Configuration

For further details on which actions to take to configure DPIO2 / DPIO2-66 to operate as input or output, see **User's Manual for DPIO2 Device Driver**.

Clocking Options

Clocking

Under normal operation of DPIO2 / DPIO2-66, the data source generates the strobe. Strobe is a continuous clock at a given frequency. This is also how the DPIO2 / DPIO2-66 default operate. However, both as input and as output, DPIO2 / DPIO2-66 have a wide range of options to change the default operation. All clocking modes are supported, both clock master and clock slave, and both in input and output modes. As a clock master DPIO2 / DPIO2-66 have a software programmable clock rate from 500 KHz to 50 MHz.

NOTE DPIO2 / DPIO2-66 use a PLL with a settling time of 1 ms. The strobe must be stable at least for this amount of time before transferring data. This requirement must be met both as a clock master and clock slave.

Input Generating Strobe, Output Receiving Strobe

In a standard setup, the output board always provides the strobe. DPIO2 / DPIO2-66 are also designed to be able to source the strobe in input mode, as well as to receive an external strobe in output mode. The same SW commands have to be used when setting up the board as in standard setup.

NOTE The timing requirements are quite different in these modes.

Programming the Output Frequency

When generating strobe, the SW programmable oscillator can be set up with frequencies from 500 kHz to 50 MHz. This oscillator generally produces an output frequency within 0.1% of the desired output frequency. If the exact frequency that the application requires cannot be achieved, it is possible to mount an oscillator carrier board (“OSCAR”) with the desired frequency. For further details, see OSCAR on page 16. To set the frequency or generate strobe, use the SW command below:

DPIO2_CMD_STROBE_FREQUENCY_ SET

The SW call will return the actual frequency that was achieved.

When DPIO2 / DPIO2-66 are operating as input or output, the following SW call will enable strobe:

DPIO2_CMD_STROBE_GENERATION_ENABLE

Low Frequency Operation

DPIO2 / DPIO2-66 use a PLL when offering the different clocking options. For low frequency operation at less than 10 MHz, the PLL must be bypassed. The driver takes care of the programming and bypassing of the PLL when using the two SW commands above.

This applies for both input and output when generating the strobe.

Receiving strobe

The PLL has different operating modes for different frequencies, and when receiving strobe, the PLL needs to be set up according to the input frequency. This is controlled by the SW command:

DPIO2_CMD_STROBE_FREQUENCY_RANGE_SET

When the desired frequency is below 10 MHz, SW will set up DPIO2 / DPIO2-66 to bypass the PLL. Unreliable operation may be the result if this range is not programmed correctly.

Double clocking (both edges)

DPIO2 / DPIO2-66 support clocking on both edges of the strobe. When this functionality is used, a PLL doubles the frequency locally. This way the frequency on the cable is half of the actual data frequency. Use the SW command below to enable double clocking:

DPIO2_CMD_CLOCKING_ON_BOTH_STROBE_EDGES_SEL

This applies for both input and output.

OSCAR

If the exact frequency that the application requires cannot be achieved with the onboard programmable oscillator, it is possible to mount an “OSCAR” with the desired frequency. OSCAR is an oscillator carrier board that can be plugged into a socket on DPIO2 / DPIO2-66. A wide range of oscillators is available on request, both in frequency and stability.

FPDP Specific Option

The FPDP specification describes two versions of the strobe signal. One is a TTL strobe and the other is a differential PECL strobe. DPIO2 / DPIO2-66 supports both these clocking options. In addition, FPDP specifies two different termination methods when using PECL strobe, and DPIO2 / DPIO2-66 support both of these termination methods.

Jumper settings

When using PECL strobe, the settings are controlled by both SW and by jumper settings. For jumper setting details, refer to the **User’s Manual for DPIO2** or **User’s Manual for DPIO2-66**. Jumpers need to be moved when:

- Setting the DPIO2 / DPIO2-66 to generate PECL strobe
- Changing the termination method for the PECL strobe

Input

When the DPIO2 / DPIO2-66 receive clock, it is possible to choose (SW controllable) between the TTL or the PECL version of the clock, using the SW command:

DPIO2_CMD_STROBE_RECEPTION_ENABLE

Output

When DPIO2 / DPIO2-66 operate as output, and are generating strobe, they can be set up to drive the differential PECL strobe. (The TTL strobe will in this case also be driven.) Jumpers need to be configured accordingly.

NOTE DPIO2 / DPIO2-66 cannot generate PECL strobe when operating as input.

DMA

About the DMA Controller

DPIO2 / DPIO2-66 have a linked list DMA controller in the PCI interface chip. The DMA controller loads DMA descriptors from the local SRAM. Each DMA descriptor points to the next descriptor making a linked list of DMA descriptors. The DMA descriptors have to be set up in DPIO2 / DPIO2-66 SRAM before starting the DMA. Up to 64K descriptors are allowed.

The DMA controller can operate in both 64-bit and 32-bit systems. EOT (End Of Transfer) can be set up to abort current transfer and jump to the next DMA descriptor. For further details about EOT, see End of Transfer on page 32.

DMA Descriptor

Next Descriptor
PCI Address
Option (D64)
Transfer Size in bytes

Table 3. DMA Descriptor.

Next Descriptor is the local address in SRAM where the next descriptor starts.

PCI Address is the start address where the DMA controller will read from / write to. The addresses must be 4 byte aligned for 32-bit accesses and 8 byte aligned for 64-bit accesses.

The Option register is used to enable/disable different options. D64 can be enabled (in 64-bit systems) through this register.

The Transfer Size can be from 16 bytes up to 16M-4 bytes. The number must be 4 byte aligned for 32-bit PCI and 8 byte aligned for 64-bit PCI.

To set the DMA descriptor, use the SW command:

DPIO2_CMD_DMA_SET_DESC

64-bit vs. 32-bit DMA

NOTE Do not set the DPIO2 / DPIO2-66 DMA to operate in 64-bit mode in 32-bit systems.

Endless DMA

By setting the DMA up as a single DMA where the next descriptor points to itself, an endless DMA is achieved. Another way to achieve an endless DMA is to set up a linked list DMA chain where the last DMA in the list points to the first.

Start

After setting up the descriptors in local SRAM, the DMA controller can be started. The DMA controller can be started independently of whether the front-end interface has been enabled or not. The DMA controller used in input mode will start to transfer data to PCI as soon as there is data in the FIFO. The DMA controller used in output mode will start fetching data immediately from PCI and write data into the FIFO.

To select the first descriptor in a chain, use the following SW command:

DPIO2_CMD_DMA_SET_START_DESCRIPTOR

Start DMA with the SW command:

DPIO2_CMD_DMA_START

Note that it is possible to have several linked list DMA descriptors in the SRAM at the same time, since the first descriptor can be located anywhere in the SRAM.

Suspend/resume

While the DMA controller is running, the DMA transfer can be paused. The DMA controller will temporarily stop, and no more transfers will be issued on PCI. Pause the DMA with the SW command:

DPIO2_CMD_DMA_SUSPEND

To continue the DMA after a pause use the SW command:

DPIO2_CMD_DMA_RESUME

While the DMA controller is suspended, the DPIO2 / DPIO2-66 front-end interface will still be enabled, unless disabled separately. When DPIO2 / DPIO2-66 are operating as output and there is buffered data in the FIFO, this data will be transferred out on to the front-end interface. When DPIO2 / DPIO2-66 are operating as input, the DPIO2 / DPIO2-66 will accept data until the FIFO gets almost full. A SUSPEND signal will then be asserted to the device sending data to indicate that the FIFO on DPIO2 / DPIO2-66 is getting full. If data is still being transferred to the DPIO2 / DPIO2-66, the FIFO will get full, and eventually data will be lost.

Stop

To stop an ongoing DMA, use the SW command:

DPIO2_CMD_DMA_STOP

Data may be lost when stopping an ongoing DMA. There might be data in the PCI interface chip, which will be lost. The data in the FIFO will still remain. If this data is not needed, a flush of the FIFO should be done.

FIFO

About the FIFO

DPIO2 include a large FIFO of up to 32K words (128KB) or optionally 128K words (512KB) (128K words only for DPIO2-66). The FIFO contains status flags, which indicate the amount of data it holds. Available FIFO statuses are: *Empty, Almost Empty, Less than Half Full, More Than Half Full, Almost Full* and *Full*. The status can be accessed via a PCI register. Certain FIFO statuses can also be set up to generate interrupts and to trigger DMA transfers. The FIFO can only have one of the statuses above, however, several flags can be active in one status. For example, when the FIFO has status Full, the Half Full, Almost Full and Full flags are all asserted. To get the current FIFO status, use the SW command:

DPIO2_CMD_GET_CURRENT_FIFO_STATUS

FIFO Size

Use the SW command below to determine the size of the FIFO:
DPIO2_CMD_GET_DEVICE_FSIZE

Resetting the FIFO

Occasionally, data can be left in the FIFO after transfer. If this data is not needed, the FIFO should be reset. When resetting the FIFO, the data in the FIFO will be lost. To reset the FIFO, use the SW command:

DPIO2_CMD_FIFO_FLUSH

Occurred Flags

If a certain FIFO flag has been activated, DPIO2 / DPIO2-66 log the flag. All FIFO flags have this occurrence logging. To check if a certain FIFO flag has occurred, use the SW commands below:

DPIO2_CMD_GET_FIFO_FULL_OCCURRED_FLAG

DPIO2_CMD_GET_FIFO_ALMOST_FULL_OCCURRED_FLAG

DPIO2_CMD_GET_FIFO_HALF_FULL_OCCURRED_FLAG

DPIO2_CMD_GET_FIFO_ALMOST_EMPTY_OCCURRED_FLAG

DPIO2_CMD_GET_FIFO_EMPTY_OCCURRED_FLAG

To reset the occurred flags, use the SW command:

DPIO2_CMD_RESET_OCCURRED_FLAGS

Overflow Detection

If the SUSPEND functionality is disabled or the output module does not support this functionality, there may be occasions where the input module receives data even if the FIFO is full. In this case, an overflow has occurred and data is lost. To detect such an occurrence, an interrupt can be set up with the SW command below:

DPIO2_CMD_INTERRUPT_ENABLE

To read the overflow register, use the command:

DPIO2_CMD_GET_FIFO_OVERFLOW_OCCURRED_FLAG

Miscellaneous Features

Bidirectionality

By setting the jumpers on DPIO2 / DPIO2-66, it is possible to configure DPIO2's power-up operating mode to either input or output mode (see the **User's Manual for DPIO2** / **User's Manual for DPIO2-66** for further details). However, SW can override the jumper settings. When using the driver, DPIO2 / DPIO2-66 are set up independently of the jumper setting.

It is possible to change the direction by SW without shutting the system down. To do that, the front panel interface has to be disabled with the SW command:

DPIO2_CMD_FPDP_ACTIVATION_SELECT

In addition the DMA controller must not be running. The direction of the DPIO2 / DPIO2-66 is set when using the SW command:

open (VxWorks)

dpio2Init (WIN2K / NT / XP / MC/OS)

If there is data in the FIFO when changing the direction of the DPIO2 / DPIO2-66, the data will remain in the FIFO. In many cases a reset of the FIFO is needed for correct operation.

NOTE Bidirectionality is not applicable to PECL and RS422 versions of DPIO2.

Front Panel Interfaces

The DPIO2 / DPIO2-66 logical interface is based on the FPDP protocol. For DPIO2, the electrical interface can be TTL (FPDP compliant), RS422, PECL or LVDS. For DPIO2-66, the electrical interface can be TTL or LVDS.

Programmable I/O bits / Reserved Signals

DPIO2 / DPIO2-66 provide two programmable I/O bits (PIO1 and PIO2) through the front panel connector. These I/O bits can be set up as either input or output (not applicable for PECL versions of DPIO2), independent of whether DPIO2 / DPIO2-66 are programmed for Input or Output. The PIO1 and PIO2 bits can be read/written via the PCI, and can generate a PCI interrupt if they are programmed as input.

DPIO2-FBU / DPIO2-FB3 (TTL)

In addition, FPDP has defined three signals as reserved, and by default, all reserved bits have status *Not Connected* on DPIO2–FBU / DPIO2-FB3. However, by moving jumpers the reserved bits can be connected (see the **User's Manual for DPIO2** / **User's Manual for DPIO2-66** for further details) if more I/O bits are needed. All three reserved signals can be set up as input or output individually.

DPIO2-LBU / DPIO2-LB3 (LVDS)

PIO1 and PIO2 can be set up as input or output individually.

DPIO2-DIU (RS-422 Input)

PIO1 and PIO2 can be set up as input or output individually.

DPIO2-DOU (RS-422 Output)

PIO1 and PIO2 can be set up as input or output individually.

DPIO2-EIU (PECL Input)

PIO1 is input only and PIO2 is output only.

DPIO2-EOU (PECL Output)

PIO1 is output only and PIO2 is input only.

Signal Termination

See the User's Manual for DPIO2 / DPIO2-66 of how to change the signal termination on the different models.

DPIO2-FBU / DPIO2-FB3 (TTL)

The FPDP specification describes different signal termination schemes for the various modes of operation. When DPIO2 / DPIO2-66 are mounted at one end of the cable as data transmitter TM (Transmit Master), the signal termination is different from when the DPIO2 / DPIO2-66 are operating as input. As input, the DPIO2 / DPIO2-66 may be terminated either as Receiver Master (RM) or as Receiver (R). RM is a receiver at the end of a cable and must terminate signals. R is a receiver located between a TM and a RM on a cable and shall have no termination. Jumper settings determine the termination of the different signals. SW can override the jumper settings for DPIO2-FBU and DPIO2-FB3.

DPIO2-LBU / DPIO2-LB3 (LVDS)

The termination for LVDS is equal when operating as input and output.

DPIO2-DIU / DPIO2-DOU (RS-422)

The termination for RS-422 is different for the transmitting side and receiving side. These modules are not bidirectional, and the termination is different for the two modules.

DPIO2-EIU / DPIO2-EOU (PECL)

The termination for PECL is different for the transmitting and receiving side. These modules are not bidirectional, and the termination is different for the two modules.

Byte Swapping

The DPIO2 / DPIO2-66 can swap data that is presented to PCI. This is done to avoid software intervention unscrambling data that is transferred to big endian systems.

All types of swapping can be performed in any combinations of 8-bit, 16-bit and 32-bit swapping:

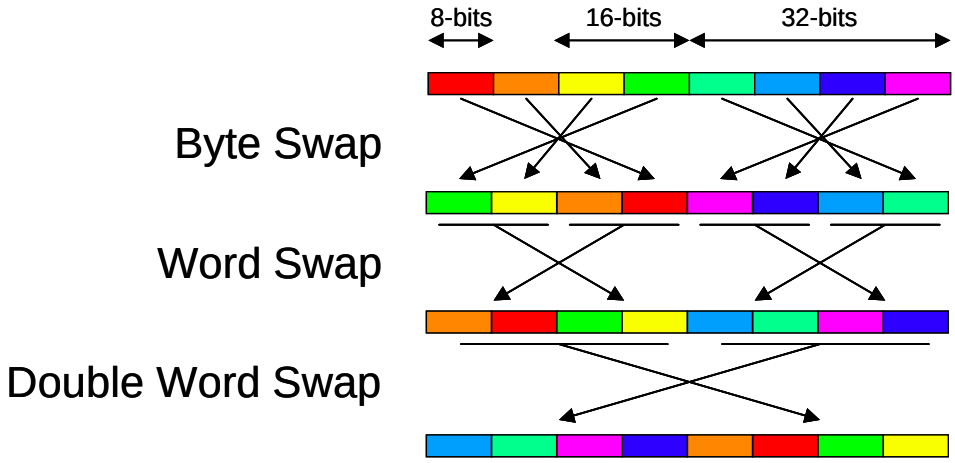

Figure 3. Byte Swapping

DPIO2 / DPIO2-66 Swapping Data Details

DPIO2 / DPIO2-66 support all types of swapping within a 64-bit word: Byte, Word and Double Word swap. The table below shows how DPIO2 / DPIO2-66 swap data. Note that 32-bit swapping only applies when using 64-bit DMA.

Byte Swap	*Word Swap*	*Dword Swap*	*Bytes*
0	0	0	11223344 55667788
1	0	0	44332211 88776655
0	1	0	33441122 77885566
0	0	1	55667788 11223344
1	1	0	22114433 66558877
1	0	1	88776655 44332211
0	1	1	77885566 33441122
1	1	1	66558877 22114433

Table 4. PIO2 Swapping Data Details

To select a swapping mode, use the SW command below:

DPIO2_CMD_DATA_SWAP_MODE_SELECT

Packing

DPIO2 / DPIO2-66 support data packing, which is useful when the front panel data is 16 bits or less. The packing/unpacking stage is located in the front-end FPGA. Packing reduces the load on the PCI bus, as it allows data I/O width to be of different sizes from the 32-bit FIFO word size.

The standard packing options provided are 3x10 bit, 2x16 bit or unpacked from a 32-bit word. Other packing options are available on request. The figure below shows an example of 3x10 bits packing.

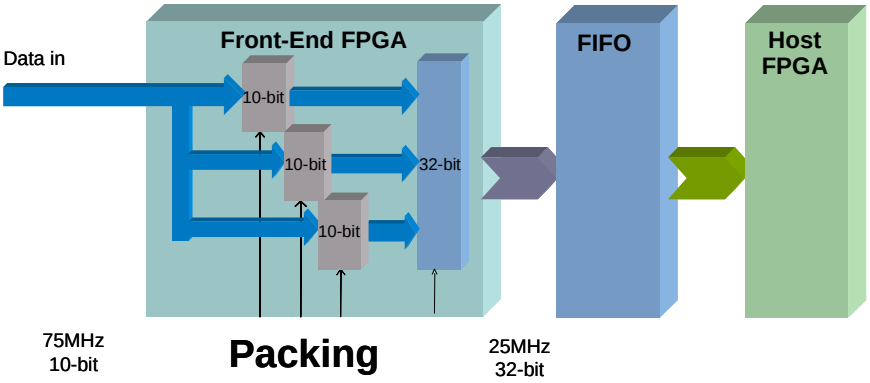

Figure 4. DPIO2 / DPIO2-66 Data Packing, 10-bit example

16-bit packing

When using 16-bit packing, it is possible to use either the 16 most significant bits or the 16 least significant bits of the front-end interface. This is done with the SW command:

DPIO2_CMD_DATA_PACKING_ENABLE

When operating in input mode, the DPIO2 / DPIO2-66 route the first 16 bit data received to the 16 least significant bits of the 32-bit word. When operating in output mode, the 16 least significant bits of the 32-bit word are sent out on the front-end interface first, followed the 16 most significant bits. Use word swap if another routing is desired.

Other packing options

When using other packing options than 16 bit, DPIO2 / DPIO2-66 use the least significant bits of the front-end interface. When operating in input mode, the DPIO2 / DPIO2-66 route the first data received to the least significant bits of the 32-bit word. When operating in output mode, the least significant bits of the 32-bit word are sent out on the front-end interface first, followed by the more significant bits until all 32 bits have been transferred.

When using 10-bit packing the two most significant bits are ignored.

Empty Data in Pipeline

The empty data in pipeline functionality is only valid when DPIO2 / DPIO2-66 operate as input and is set up in packing mode. If DPIO2 / DPIO2-66 do not receive enough data to fill up the 32-bit word, it is possible that data gets stuck in the registers. To empty data in the pipeline, stop the acquisition with the following SW command:

DPIO2_CMD_FPDP_ACTIVATION_SELECT

Then empty the data with the following command:

DPIO2_CMD_DATA_PACKING_PIPELINE_FLUSH

LEDs

DPIO2 / DPIO2-66 have five LEDs that may be enabled to monitor activity on control signals. By default, these LEDs are not enabled. The figure below shows where the LEDs are positioned.

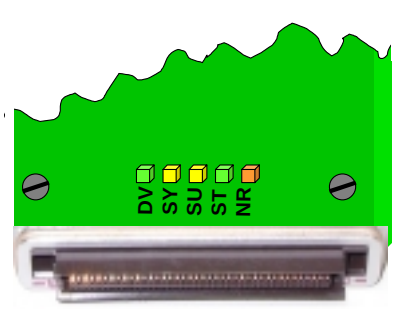

Figure 5. LEDs

Abbreviation	*Name*	*Color*	*Lights*
DV	DVALID	Green	Lights when DVALID is asserted.
SY	SYNC	Yellow	Lights when SYNC is asserted.
SU	SUSPEND	Yellow	Lights when SUSPEND is asserted.
ST	STROBE	Green	Will glow when STROBE is running.
NR	NRDY	Orange	Lights when NRDY is asserted.

Table 5. DPIO2 / DPIO2-66 LEDs

Synchronization Modes

FPDP describes different ways of dividing data into frames. In many cases, it is necessary to know when a frame begins and when it ends for synchronizing input to output modules. FPDP uses the SYNC signal as synchronization signal. The FPDP specification divides frames into three groups.

- Unframed data
- Single frame
- Repeating frame, fixed size and dynamic size

When using unframed data, there is no synchronization. This means that the SYNC signal is not in use.

Single Frame

Single frame is used to synchronize one time and one time only. The SYNC signal is asserted previous to the first data.

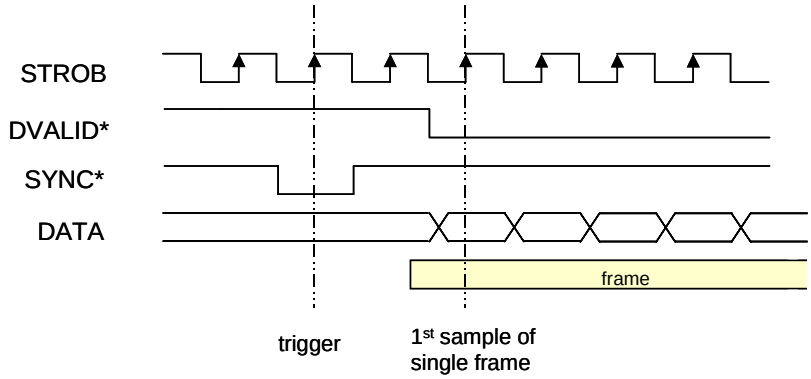

Figure 6. Sync without DVALID, simple trigger for start of single frame.

Fixed Size Repeating Frame

Fixed size repeating frame is indicated by SYNC asserted together with data at the end of every frame. The amount of data in each frame is fixed.

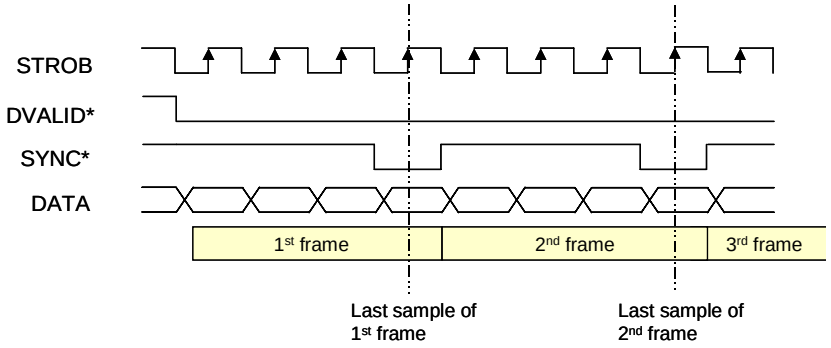

Figure 7. Sync with DVALID, fixed size repeating frame

Dynamic Size Repeating Frame

Dynamic size repeating frame is also indicated by SYNC asserted together with data at the end of every frame. The amount of data in each frame is not known in advance.

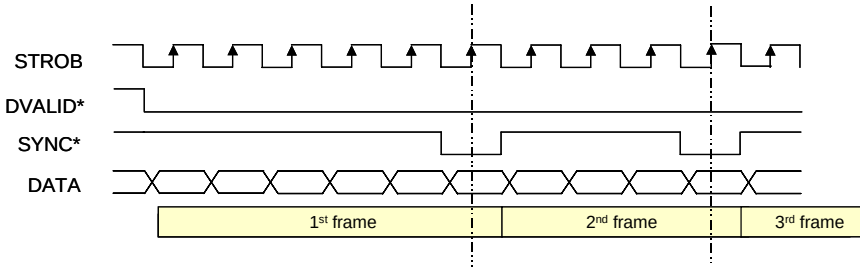

Figure 8. Sync with DVALID, dynamic size repeating frame

Synchronization in Output Mode

In output mode the SYNC signal can be generated at particular points in the data stream in order to signal data frame synchronization. The available modes are:

- No synchronization
- Assert SYNC prior to first data (Will be reasserted every time the front-end interface is enabled)
- Assert SYNC together with last data of frame (*)
- Assert SYNC together with first data of frame (*)
- Assert SYNC every n'th data (independent of linked list DMA operation)
- Assert SYNC during entire frame (backward compatible with first generation DPIO from VMETRO. Not recommended for future use.)

(*) Frame size in these two modes will correspond to the amount of data transferred by one DMA descriptor. If dynamic size frames are wanted, use different DMA sizes.

To set the desired synchronization option on DPIO2 / DPIO2-66 use the SW command:

DPIO2_CMD_SYNC_GENERATION_SELECT

When asserting SYNC together with every n'th data, use the following SW command to set the value of n:

DPIO2_CMD_SYNC_GENERATION_COUNTER_SET

D0 as SYNC

In addition it is possible to route data bit 0 out as SYNC. When this option is selected, the options listed above are not in affect. Use the SW command below to activate D0 as SYNC:

DPIO2_CMD_D0_TO_BE_USED_FOR_SYNC_SELECT

Synchronization in Input Mode

When the DPIO2 / DPIO2-66 are operating in input mode, the following synchronization options are available:

- No synchronization
- Wait for SYNC prior to first sample of frame
- SYNC as EOT signal (controls DMA progress)

Use the SW command below to set synchronization on input:

DPIO2_CMD_SYNC_RESEPTION_SELECT

When using the "Wait for SYNC" option, the DPIO2 / DPIO2-66 will ignore all valid data until an active SYNC signal is detected. After that all valid data will be accepted. To be able to reset the functionality again, and wait for a new SYNC, the front-end interface has to be disabled and enabled again. See SW command:

DPIO2_CMD_FPDP_ACTIVATION_SELECT

For details about EOT (End Of Transfer), see below.

SYNC as D0

In addition it is possible to route SYNC into the data stream as data bit 0. This option is independent of the options above. Use the SW command below to activate D0 as SYNC:

DPIO2_CMD_D0_TO_BE_USED_FOR_SYNC_SELECT

End of Transfer

End-Of-Transfer (EOT) is a condition that can abort an ongoing DMA transfer. If there exists a chain of linked DMA descriptors, the DMA engine will abort the current DMA transfer and optionally jump to the next DMA descriptor in the chain. This function is especially useful in handling variable length frames delimited by a synchronizing signal. The signal SYNC is used to generate EOT.

Byte Count Register

When using the End-Of-Transfer functionality, it is possible to read out how many bytes of the DMA transfer that was not transferred. Use the following SW command to get the remaining byte count of last DMA:

DPIO2_CMD_REMAINING_BYTE_COUNT_GET

A situation may occur, due to short DMAs or slow interrupt handling, that this register is overwritten before it is read. When this occurs, an overflow flag indicates that the byte count value has been overwritten. For further details about this, see **User's Manual for DPIO2 Device Driver**.

Advanced Features

DPIO2 / DPIO2-66 DMA Modes

Default Mode

The default DMA mode is the most common mode for DPIO2 / DPIO2-66. The DMA controller waits until there is data in the FIFO, and then transfers data as long as there is data in the FIFO. This gives low latency on data transmitted to the memory. As data can be sent in single cycles or small bursts, the efficiency on the PCI bus might not be optimal. This will not represent any problems in most systems.

PCI bus has mechanisms that can improve the utilization of the PCI bus. Latency counters are used to ensure that PCI devices share the PCI bus fairly, so that all devices have enough bandwidth. See the PCI specification for further details on latency.

By default, the latency counter in DPIO2 / DPIO2-66 is set to maximum by the driver.

Demand Mode

The demand DMA mode is set to send a large amount of data to the PCI bus in longer bursts. To ensure efficient use of the PCI bus when DPIO2 / DPIO2-66 act as a BUS MASTER, DPIO2 / DPIO2-66 can be set to start the DMA when the FIFO is *More than Half Full*. Half the FIFO size is then emptied onto the PCI bus, and the controller waits for the FIFO to get *More than Half Full* again. This means that as much as 64 KB or 256 KB, depending on the DPIO2 model, is transferred at the time using the PCI bus to its maximum. This mode of operation optimizes the use of PCI bandwidth in applications where this is a bottleneck, at the cost of latency.

Flow Control

The FPDP specification describes how flow control between modules can be achieved. When a FPDP device is powered up, configured as input, the module issues “not ready” to the output module, by automatically activating the NRDY signal. The output will not send data until all input modules are ready, indicated by deassertion of NRDY. The DPIO2 / DPIO2-66 operate according to the FPDP specification. On DPIO2 / DPIO2-66, NRDY will be deasserted to indicate that it is ready to receive data, when the front-end interface is enabled.

During transfer, the input module can assert “suspend” which indicates that the module temporarily can not receive more data. The output must then suspend the transfer until the input is ready to receive data again. DPIO2 / DPIO2-66 as input assert “suspend” when FIFO is almost full. It is possible for DPIO2 / DPIO2-66, both as input and output, to disable this functionality by using the command:

DPIO2_CMD_SUSPEND_FLOW_CONTROL_SELECT

Followed by the command:

DPIO2_CMD_NRDY_FLOW_CONTROL_SELECT

Input will then not assert “not ready” and “suspend”, and output ignores “not ready” and “suspend”.

NOTE DPIO2 / DPIO2-66 have adopted the flow control that is described in the FPDP specification on all models, except on the PECL version. On PECL version, the NRDY signal and its functionality are not implemented. SUSPEND works as for the other modules.

Sample Skip and Store

The sample skip and store functionality is typically used when more than one module receives data. The three registers that need to be set up are:

- Initial Sample Skip Count
- Samples to Accept Count
- Sample Skip Count

Use the following SW command to enter this feature:

DPIO2_CMD_COUNTER_ADDRESSING_ENABLE

Example

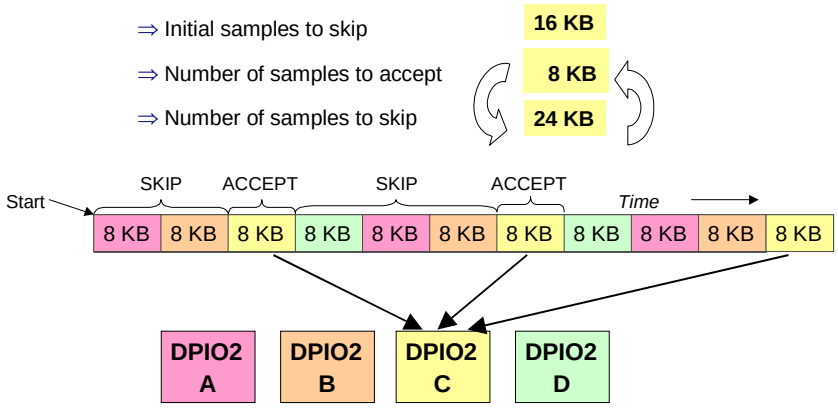

Figure 9. Sample skip and store

The way it works is that the DPIO2 / DPIO2-66 in the example above (module C) will first skip 16KB of data. This initial skip is sort of a synchronization of the four modules. Then the DPIO2 / DPIO2-66 will accept the next 8KB of data, skip 24KB of data, accept 8KB, skip 24KB etc. The other modules (A, B and D) will have different initial samples to skip, but will also accept 8KB and skip 24KB after the initial sample skip.

Video Mode

In video mode, the DPIO2 / DPIO2-66 interpret the SYNC and DVALID signal differently than in standard mode. In standard mode, data is valid whenever DVALID is active, and frames are divided up by a single SYNC pulse.

In video mode, DVALID is treated as vertical sync and SYNC is treated as horizontal sync. Data is valid whenever both signals are active, and the frames start when the vertical sync goes from inactive state to active state.

To set the DPIO2 / DPIO2-66 to interpret SYNC and DVALID as video synchronization signals, use the SW command:

DPIO2_CMD_VIDEO_MODE_SELECT

To have an effect, this command must be issued before the front-end interface is activated.

NOTE This feature is only applicable if DPIO2 / DPIO2-66 are configured as *input*.

Bus Mode Signals

DPIO2 / DPIO2-66 are PMC (PCI Mezzanine Cards). As a part of the PMC specification, there are four BUS MODE signals. These signals are used to determine that the DPIO2 / DPIO2-66 are sitting in a PMC site.

When BUSMODE[4:2]# = ‘001’, BUSMODE1# is driven active to ‘0’, and the board will operate as a PCI board.

When BUSMODE[4:2]# = ‘001’, BUSMODE1# is driven active to ‘0’, but all PCI signals will be three-stated.

For all other BUSMODE[4:2]# combinations BUSMODE1# will be driven inactive to ‘1’ and all PCI signals will be tri-stated.

Refer to the **PMC specification** for further details.

Pattern Generation

In the front-end FPGA on the DPIO2 / DPIO2-66 there is a pattern generator. The four patterns that can be generated are walking-zero, walking one, a 32 bit count pattern starting at 0 and counting up, and a 32-bit count pattern starting at a predefined value and counting up.

When operating in output mode, the DPIO2 / DPIO2-66 can generate a pattern out on the front panel interface, without DMA being used, and without having to read data from PCI. Data will be generated in the front-end FPGA and sent directly out on the front-end interface. The pattern generator uses the programmable oscillator, and by tuning the frequency, the DPIO2 / DPIO2-66 can output almost any throughput between 2MB/s and 400MB/s.

When operating in input mode, the DPIO2 / DPIO2-66 can generate a pattern and write the pattern into the DPIO2 / DPIO2-66 FIFO. A DMA can then be set up to write data out on to PCI bus. This means that the DPIO2 / DPIO2-66 can be tested without any data source. The pattern generator uses the programmable oscillator. The DPIO2 / DPIO2-66 can generate data at any throughput between 2MB/s and 400MB/s by setting the frequency. The throughput on PCI is dependent on many factors and if the PCI bandwidth is lower than that of the generation of data, the FIFO will be filled. The data generation will then pause until there is room for more data in the FIFO.

NOTE When generating a pattern with DPIO2 / DPIO2-66 set up in input mode, the DPIO2 / DPIO2-66 need to generate its own strobe. The strobe will be driven out on to the front panel interface.

Front-End-FPGA

On advanced applications, the factory can reprogram the front-end FPGA for custom functions regarding synchronization of input data. This can be done on functions that handle incoming data, control signals and insertion of timestamps, tags etc. Please contact factory for any custom requirements (volume purchase required.)

Support

Updating FLASH with New FPGA Image

The two FPGAs used on DPIO2 / DPIO2-66 get the images from an onboard FLASH. SW can upgrade this FLASH. This means that upgrades can be done on site. On DPIO2-66 it is possible to WRITE protect the FLASH (DIP switch setting).

DPIO2 / DPIO2-66 Software Support

DPIO2 / DPIO2-66 offer drivers for VxWorks (PowerPC and i960), Mercury MC/OS and Windows 2000/NT/XP. For customers who want to modify the driver, or port the driver to another hardware platform or other operating system, VMETRO also provides the source code (Source Kit).

The MC/OS driver provides direct DPIO2 / DPIO2-66 I/O support inside the processors of a Mercury RACEway or RACE++ multiprocessor system. The Windows 2000/NT/XP driver is useful for applications where DPIO2 / DPIO2-66 are installed in “regular” PCI systems, such as desktop PCs and servers where DPIO2 / DPIO2-66 are installed on PCI adapters.

Figures and Tables

List of Tables

List of Figures