Uniform Driver Interface

UDI PCI Bus Binding Specification Version 1.0

http://www.project-UDI.org/specs.html

UDI PCI Bus Binding Specification

Abstract

The UDI PCI Bus Binding Specification defines the requirements for use of the UDI Physical I/O Specification on the PCI Bus. This is an optional extension to the UDI Physical I/O Specification, which is defined in a separate book. The intended audience for this book includes driver writers, environment implementors, and metalanguage implementors.

UDI drivers that require use of physical I/O on a PCI bus must be written to conform to this specification, and can assume that all services described herein are available. Environments that don't need such drivers may choose not to support the PCI extensions, but any environment that supports UDI PCI drivers must conform to this specification, as well as to the UDI Physical I/O Specification and the UDI Core Specification.

See the Document Organization chapter in the UDI Core Specification for a description of other books in the UDI Specification collection, as well as references to additional tutorial materials.

Status of This Document

This document has been reviewed by Project UDI Members and other interested parties and has been endorsed as a Final Specification. It is a stable document and may be used as reference material or cited as a normative reference from another document. This version of the specification is intended to be ready for use in product design and implementation. Every attempt has been made to ensure a consistent and implementable specification. Implementations should ensure compliance with this version.

Copyright Notice

Copyright © 1999 Adaptec, Inc; Compaq Computer Corporation; Hewlett-Packard Company; International Business Machines Corporation; Interphase Corporation; Lockheed Martin Corporation; The Santa Cruz Operation, Inc; SBS Technologies, Inc; Sun Microsystems ("copyright holders"). All Rights Reserved.

This document and other documents on the Project UDI web site (`www.project-UDI.org`) are provided by the copyright holders under the following license. By obtaining, using and/or copying this document, or the Project UDI document from which this statement is linked, you agree that you have read, understood, and will comply with the following terms and conditions:

Permission to use, copy, and distribute the contents of this document, or the Project UDI document from which this statement is linked, in any medium for any purpose and without fee or royalty is hereby granted, provided that you include all of the following on *ALL* copies of the document, or portions thereof, that you use:

1. A link or URI to the original Project UDI document.
2. The pre-existing copyright notice of the original author, or, if it doesn't exist, a Project UDI copyright notice of the form shown above.
3. *If it exists*, the STATUS of the Project UDI document.

When space permits, inclusion of the full text of this **NOTICE** should be provided. In addition, credit shall be attributed to the copyright holders for any software, documents, or other items or products that you create pursuant to the implementation of the contents of this document, or any portion thereof.

No right to create modifications or derivatives is granted pursuant to this license.

THIS DOCUMENT IS PROVIDED "AS IS," AND COPYRIGHT HOLDERS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THE DOCUMENT ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

COPYRIGHT HOLDERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE DOCUMENT OR THE PERFORMANCE OR IMPLEMENTATION OF THE CONTENTS THEREOF.

The names and trademarks of copyright holders may NOT be used in advertising or publicity pertaining to this document or its contents without specific, written prior permission. Title to copyright in this document will at all times remain with copyright holders.

Acknowledgements

The authors would like to thank everyone who reviewed working drafts of the specification and submitted suggestions and corrections.

The authors would especially like to thank their significant others for putting up with the many hours of overtime put into the development of this specification over long periods.

Thanks to the following folks who contributed significant amounts of time, ideas, or authoring in support of the development of this specification:

> Richard Arndt (IBM), Bob Barned (Lockheed Martin), Mark Bradley (Adaptec), Darren Busing (Adaptec), Thomas Clark (Sun), Mark Evenson (HP), Barry Feild (SCO), Kurt Gollhardt (SCO), Jim Heidbrink (Lockheed Martin), Chris Ilnicki (HP), Bret Indrelee (SBS Technologies), David Kahn (Sun), John Lee (Sun), Robert Lipe (SCO), Lynne McCue (IBM), Mark Parenti (DEC), James Partridge (IBM), Kevin Quick (Interphase), Larry Robinson (Adaptec), James Smart (Compaq), Pete Smoot (HP), David Stoft (HP), Wolfgang Thaler (Sun), Linda Wang (Sun), Mike Wenzel (HP).

In addition, thanks to everyone who worked on the prototype implementations that helped us validate the specification, including:

> Betty Dall (HP), Don Dugger (Intel), Bob Goudreau (Data General), James Hall (SCO), Matt Kaufman (SCO), Mike Lyons (IBM), Alex Malone (DEC), Guru Pangal (Starcom), Scott Popp (SCO), Richard Schall (Intel), Sam Shteingart (HP), Ajmer Singh (SCO), Ramaswamy Tummala (Starcom).

Finally, thanks to David Roberts (Certek Software Designs) for designing the Project UDI logo.

Table of Contents

Abstract i
Copyright Notice ii
Acknowledgements iii
Table of Contents v

1 Introduction to the PCI Bus Binding 1-1
1.1 Overview 1-1
1.2 General Requirements 1-1
1.3 Normative References 1-1

2 PCI Bus Bindings 2-1
2.1 PIO Bindings 2-1
2.1.1 udi_pio_map 2-1
2.2 Interrupt Bindings 2-1
2.2.1 Interrupt Index Values 2-1
2.2.2 Event Info 2-1
2.3 Instance Attribute Bindings 2-2
2.3.1 Enumeration Attributes 2-2
2.3.2 Filter Attributes 2-2
2.3.3 Generic Enumeration Attributes 2-3
2.3.3.1 identifier attribute 2-3
2.3.3.2 address_locator attribute 2-3
2.3.3.3 physical_locator attribute 2-3
2.3.3.4 physical_label attribute 2-3
2.3.4 Parent-Visible Attributes 2-3

Introduction to the PCI Bus Binding 1

1.1 Overview

The UDI PCI Bus Binding specifies the usage details for the UDI Physical I/O Specification that are specific to the PCI Bus. This chapter defines general requirements for use of the UDI PCI Bus Binding Specification. The next chapter defines the specifics of the UDI PCI Bus Binding.

1.2 General Requirements

Certain basic rules apply to all UDI PCI drivers (both bus bridges and adapter drivers that use the PCI bus). In order to be UDI-compliant, a driver must follow all of these rules. UDI PCI drivers must also follow the rules specified in the UDI Physical I/O Specification and the UDI Core Specification. Rules specific to PCI drivers are listed here.

Before including any UDI header files, the driver must define the preprocessor symbol, UDI_PCI_VERSION, to indicate the version of the UDI PCI Bus Binding Specification to which it conforms. For this version of the specification, UDI_PCI_VERSION must be set to 0x100:

```
#define UDI_PCI_VERSION 0x100
```

Each device driver source file must include the file "`udi_pci.h`" after it includes "`udi.h`" and "`udi_physio.h`", as follows:

```
#include <udi.h>
#include <udi_physio.h>
#include <udi_pci.h>
```

These header file contains environment-specific definitions of standard UDI structures and types, as well as all function prototypes and other definitions needed to use the core, physical I/O, and PCI bus UDI interfaces and services. Additional include files may be needed for other non-core services and metalanguages as defined in other UDI Specifications.

To maintain portability across UDI supportive platforms, device driver writers shall not assume any knowledge of the contents of these header files with respect to implementation-dependent aspects of the UDI interfaces (such as the definition of handles or abstract types). Similarly, drivers shall not access any functions or objects external to the driver except those defined in the UDI Specifications to which they conform.

1.3 Normative References

The UDI PCI Bus Binding Specification references the non-UDI standards listed below. These standards contain provisions that, through reference in this document, constitute provisions of the UDI PCI Bus Binding Specification.

1. PCI Local Bus Specification.

PCI Bus Bindings 2

Some of the UDI services interfaces defined in the UDI Physical I/O Specification require bus binding information to appropriately use the interface and set parameter values. This chapter specifies the bus bindings for the PCI bus.

2.1 PIO Bindings

2.1.1 udi_pio_map

The following ***regset_idx*** values are defined for PCI:

```
#define UDI_PCI_CONFIG_SPACE            255
#define UDI_PCI_BAR_0                   0
#define UDI_PCI_BAR_1                   1
#define UDI_PCI_BAR_2                   2
#define UDI_PCI_BAR_3                   3
#define UDI_PCI_BAR_4                   4
#define UDI_PCI_BAR_5                   5
```

Any 64-bit BARs must be accessed by the lowest of the two BAR numbers used to hold the 64-bit value.

Any other values passed to `udi_pio_map` in the ***regset_idx*** argument are illegal.

2.2 Interrupt Bindings

2.2.1 Interrupt Index Values

Since PCI allows only one interrupt pin per PCI function, only one ***interrupt_idx*** value, zero, is used for PCI devices.

2.2.2 Event Info

There is no event info for PCI bus devices. Event info size must always be zero.

2.3 Instance Attribute Bindings

2.3.1 Enumeration Attributes

The following enumeration attributes are defined for PCI devices. All numeric attributes are stored as `UDI_ATTR_UBIT32` type attributes, which are automatically converted to/from each driver's endianness. Since PCI configuration space values are little-endian, the bus bridge driver must combine individual bytes into a numeric value (e.g. "`attr_value = lo_byte + (hi_byte << 8)`"), or store the little-endian values directly in ***`attr_value`*** of the `udi_instance_attr_list_t` rather than using `UDI_ATTR32_SET()`.

Table 2-1 PCI Enumeration Attributes

ATTRIBUTE NAME	TYPE	SIZE	Description
`bus_type`	UDI_ATTR_STRING	4	"`pci`"
`pci_vendor_id`	UDI_ATTR_UBIT32	4	16-bit numeric PCI Vendor ID
`pci_device_id`	UDI_ATTR_UBIT32	4	16-bit numeric PCI Device ID
`pci_revision_id`	UDI_ATTR_UBIT32	4	8-bit numeric PCI Revision ID
`pci_baseclass`	UDI_ATTR_UBIT32	4	8-bit numeric PCI Base Class Code
`pci_sub_class`	UDI_ATTR_UBIT32	4	8-bit numeric PCI Sub-Class Code
`pci_prog_if`	UDI_ATTR_UBIT32	4	8-bit numeric PCI Programming Interface
`pci_subsystem_vendor_id`	UDI_ATTR_UBIT32	4	16-bit numeric PCI Subsystem Vendor ID
`pci_subsystem_id`	UDI_ATTR_UBIT32	4	16-bit numeric PCI Subsystem ID
`pci_unit_address`	UDI_ATTR_UBIT32	4	low-order 3 bits: PCI Function Number next 5 bits: PCI Device Number next 8 bits: PCI Bus Number
`pci_slot`	UDI_ATTR_UBIT32	4	8-bit physical slot number, if known

2.3.2 Filter Attributes

Of the above listed enumeration attributes, the following are valid filter attributes for enumeration filtering. For both of these, stride is interpreted linearly; that is, the stride value is simply added to the numeric value of these attributes.

```
pci_unit_address
pci_slot
```

2.3.3 Generic Enumeration Attributes

2.3.3.1 identifier attribute

For PCI devices, the "identifier" attribute encodes a combination of the pci_vendor_id, pci_device_id, pci_revision_id, pci_subsystem_vendor_id, and pci_subsystem_id attributes, as follows:

```
identifier = VVVVDDDDRRvvvvdddd
```

where VVVV is a four-digit upper-case hexidecimal-encoded ASCII representation of the PCI Vendor ID, DDDD is a four-digit upper-case hexidecimal-encoded ASCII representation of the PCI Device ID, RR is a two-digit upper-case hexidecimal-encoded ASCII representation of the PCI Revision ID, vvvv is a four-digit upper-case hexidecimal-encoded ASCII representation of the PCI Subsystem Vendor ID, and dddd is a four-digit upper-case hexidecimal-encoded ASCII representation of the PCI Subsystem ID.

2.3.3.2 address_locator attribute

For PCI devices, the "address_locator" attribute encodes the pci_unit_address attribute using the following syntax:

```
address_locator = BBDDF
```

where BB is a two-digit upper-case hexidecimal-encoded ASCII representation of the PCI Bus Number, DD is a two-digit upper-case hexidecimal-encoded ASCII representation of the PCI Device Number, and F is a one-digit upper-case hexidecimal-encoded ASCII representation of the PCI Function Number.

2.3.3.3 physical_locator attribute

For PCI devices, the "physical_locator" attribute encodes the pci_slot attribute using the following syntax:

```
physical_locator = SS
```

where SS is a two-digit upper-case hexidecimal-encoded ASCII representation of the physical slot number, if known. The physical_locator attribute is present if and only if pci_slot is provided for this device.

2.3.3.4 physical_label attribute

No "physical_label" attribute is defined generically for PCI. Platforms that have access to such information may set physical_label attributes.

2.3.4 Parent-Visible Attributes

No parent-visible attributes are defined for PCI bus bridge drivers.