Uniform Driver Interface

UDI System Bus Binding Specification Version 1.01

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

UDI System Bus Binding Specification

Abstract

The UDI System Bus Binding Specification defines the requirements for use of the UDI Physical I/O Specification on what is known in UDI as the *System Bus*. The *system bus* abstraction may be used with built-in I/O on a system motherboard, and in general is intended for use with any non-self-identifying bus such as legacy ISA.

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 System 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 System Bus extensions, but any environment that supports UDI drivers for devices on a System Bus 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

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 or in working on the prototype implementations which helped us validate the specification:

Mark Evenson (HP)
Kurt Gollhardt (SCO)

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

Table of Contents

Introduction to the System Bus Binding 1

1.1 Overview

The UDI System Bus Binding specifies the usage details for the UDI Physical I/O Specification that are specific to what is known in UDI as the *System Bus*. The *system bus* abstraction may be used with built-in I/O devices on a system motherboard, and in general is intended for use with any non-self-identifying bus, such as legacy ISA. This chapter defines general requirements for use of the UDI System Bus Binding Specification. The next chapter defines the specifics of the UDI System Bus Binding.

Note – While the System Bus Binding can be used for most ISA devices, it specifically does not support ISA slave DMA (using shared DMA controllers on the motherboard).

1.2 General Requirements

Certain basic rules apply to all UDI System Bus drivers (for both System Bus bridges and devices attached to a System Bus). In order to be UDI-compliant, such a driver must follow all of these rules. UDI System Bus drivers must also follow the rules specified in the UDI Physical I/O Specification and the UDI Core Specification. Rules specific to System Bus drivers are listed here.

Before including any UDI header files, the driver must define the preprocessor symbol, UDI_SYSBUS_VERSION, to indicate the version of the UDI System Bus Binding Specification to which it conforms. For this version of the specification, UDI_SYSBUS_VERSION must be set to 0x101:

```
#define UDI_SYSBUS_VERSION 0x101
```

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

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

These header files contain 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 System 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 System Bus Binding Specification references and depends only upon the UDI Core and Physical I/O Specifications. No other UDI or non-UDI specifications are required for the use of the System Bus Specification.

System 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 applicable to what is known in UDI as the *System Bus*. The *system bus* abstraction may be used with built-in I/O devices on a system motherboard, and in general is intended for use with any non-self-identifying bus, such as legacy ISA.

2.1 PIO Bindings

2.1.1 udi_pio_map

The following ***regset_idx*** values are defined for the System Bus:

```
#define UDI_SYSBUS_MEM                      0
#define UDI_SYSBUS_IO                       1
#define UDI_SYSBUS_MEM_BUS                  2
#define UDI_SYSBUS_IO_BUS                   3
```

`UDI_SYSBUS_MEM` and `UDI_SYSBUS_IO` are used, respectively, to map memory and I/O space device memory that is relative to the base of the child device; i.e. device-relative addresses whose corresponding base bus addresses are provided by the `sysbus_mem_addr_xx` and `sysbus_io_addr_xx` enumeration attributes (see Section 2.3.2, “Enumeration Attributes” below).

`UDI_SYSBUS_MEM_BUS` and `UDI_SYSBUS_IO_BUS` are used, respectively, to map memory and I/O space device memory that is relative to the base of the bus; i.e. bus-relative addresses whose corresponding base bus addresses are provided by the `sysbus_mem_bus_addr_xx` and `sysbus_io_bus_xx` enumeration attributes (see Section 2.3.2, “Enumeration Attributes” below).

Note – The use of bus-relative addressing here is an exception to PIO addressing, which is otherwise device-relative.

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

Warning – It is illegal to use a System Bus mapping to map “mainstore” system memory. System Bus mappings must only be used for device memory and I/O space.

2.2 Interrupt Bindings

2.2.1 Interrupt Index Values

The System Bus Binding supports devices with no interrupts or with at most two interrupt sources. The ***`interrupt_idx`*** values zero and one are used to select the first and second interrupt source, respectively. The `sysbus_intr0` and `sysbus_intr1` enumeration attributes, described below, indicate how these interrupt sources are routed to interrupt "lines", or inputs to the system interrupt controller.

2.2.2 Event Info

There is no event info for System Bus interrupts. Event info size must always be zero.

2.3 Instance Attribute Bindings

2.3.1 Enumeration Requirements

A System Bus bridge driver must always respond to normal (non-directed) enumeration requests with ENUMERATE_DONE (i.e. no child devices indicated), and must support directed enumeration.

2.3.2 Enumeration Attributes

The following enumeration attributes are defined for devices on a System Bus. If additional PIO address ranges are needed for a child device, beyond what can be provided via these well-known attributes, they will be given to the System Bus driver via custom parameters, which must be declared in the driver's udiprops.txt file.

Since System Bus devices are non-self-identifying, these attributes will generally be determined by administrative input and/or platform-specific heuristics, and passed to the System Bus bridge driver via directed enumeration.

All numeric attributes are stored as `UDI_ATTR_UBIT32` type attributes, which are automatically converted to/from each driver's endianness.

Table 2-1 System Bus Enumeration Attributes

ATTRIBUTE NAME	TYPE	SIZE	Description
`bus_type`	UDI_ATTR_STRING	7	"`system`"
`sysbus_mem_addr_lo`	UDI_ATTR_UBIT32	4	The least significant 32 bits of the child device's base memory address
`sysbus_mem_addr_hi`	UDI_ATTR_UBIT32	4	The most significant 32 bits of the child device's base memory address
`sysbus_mem_size`	UDI_ATTR_UBIT32	4	The size, in bytes, of the child device's base memory area (may be zero; treated as zero if not present)

Table 2-1 System Bus Enumeration Attributes

ATTRIBUTE NAME	**TYPE**	**SIZE**	**Description**
`sysbus_io_addr_lo`	UDI_ATTR_UBIT32	4	The least significant 32 bits of the child device's base I/O space address
`sysbus_io_addr_hi`	UDI_ATTR_UBIT32	4	The most significant 32 bits of the child device's base I/O space address
`sysbus_io_size`	UDI_ATTR_UBIT32	4	The size, in bytes, of the child device's base memory area (may be zero; treated as zero if not present)
`sysbus_intr0`	UDI_ATTR_UBIT32	4	The interrupt line number for the first interrupt source, if any; 0xFFFFFFFF or not present if no such interrupt source
`sysbus_intr1`	UDI_ATTR_UBIT32	4	The interrupt line number for the second interrupt source, if any; 0xFFFFFFFF or not present if no such interrupt source

If `sysbus_mem_size` is zero or not present, `sysbus_mem_addr_lo` and `sysbus_mem_addr_hi` are not required and must be ignored.

If `sysbus_io_size` is zero or not present, `sysbus_io_addr_lo` and `sysbus_io_addr_hi` are not required and must be ignored.

In some cases, a System Bus device may have multiple discontiguous memory or I/O address ranges, which cannot be reflected in the above enumeration attributes. The device driver must determine the secondary ranges in a driver-specific fashion. For some devices, the driver will be able to query its device (via registers in the primary range) to determine the secondary addresses. Drivers for devices without this capability will have to declare driver-specific custom parameters (see the "Custom Declaration" section of *"Static Driver Properties"* in the UDI Core Specification) that administrators can use to configure the secondary range(s). In either case, the driver must use bus-relative addresses with SYSBUS_MEM_BUS/SYSBUS_IO_BUS to map the secondary range(s).

For `sysbus_intr0` and `sysbus_intr1`, the "interrupt line number" indicates which of possibly several distinguishable inputs to the system's interrupt controller is associated with a particular interrupt source. These might indicate interrupt vectors or interrupt request lines. The mapping between interrupt line numbers and physical interrupts, as well as the range of legal interrupt line number values, is platform-specific.

2.3.3 Filter Attributes

There are no enumeration attribute filters defined for the enumeration of the System Bus.

2.3.4 Generic Enumeration Attributes

2.3.4.1 identifier attribute

Since System Bus devices are not self-identifying, the System Bus bridge cannot create a meaningful "`identifier`" attribute. If the System Bus bridge driver is not passed an "`identifier`" attribute value as part of directed enumeration, it must add an "`identifier`" attribute with a value of "`none`".

2.3.4.2 address_locator attribute

For System Bus devices, the "`address_locator`" attribute encodes the `sysbus_mem_addr_lo`, `sysbus_mem_addr_hi`, `sysbus_io_addr_lo`, and `sysbus_io_addr_hi` attributes, using the following syntax:

```
      address_locator = MMMMMMMMmmmmmmmmIIIIIIIIiiiiiiii
```

where `MMMMMMMM` is an eight-digit upper-case hexidecimal-encoded ASCII representation of `sysbus_mem_addr_hi`, `mmmmmmmm` is an eight-digit upper-case hexidecimal-encoded ASCII representation of `sysbus_mem_addr_lo`, `IIIIIIII` is an eight-digit upper-case hexidecimal-encoded ASCII representation of `sysbus_io_addr_hi`, and `iiiiiiii` is an eight-digit upper-case hexidecimal-encoded ASCII representation of `sysbus_io_addr_lo`. The System Bus bridge driver must set this attribute if it was not already set in the list of attributes passed to it as part of the directed enumeration.

2.3.4.3 physical_locator attribute

No "`physical_locator`" attribute is defined generically for the System Bus.

2.3.4.4 physical_label attribute

No "`physical_label`" attribute is defined generically for the System Bus. Platforms that have access to such information may set `physical_label` attributes.

2.3.5 Enumeration Attribute Ranking

Enumeration attribute ranking does not apply to the System Bus binding, since all child driver instances are created via directed enumeration.

2.3.6 Parent-Visible Attributes

No parent-visible attributes are defined for System Bus bridge drivers.