Software Specification

PCI eXtensions for Instrumentation

An Implementation of *CompactPCI™*

Revision 2.1
February 4, 2003

IMPORTANT INFORMATION

Copyright

NOTICE

The *PXI Software Specification* is authored and copyrighted by the PXI Systems Alliance. The intent of the PXI Systems Alliance is for the *PXI Software Specification* to be an open industry standard supported by a wide variety of vendors and products. Vendors and users who are interested in developing PXI-compatible products or services, as well as parties who are interested in working with the PXI Systems Alliance to further promote PXI as an open industry standard, are invited to contact the PXI Systems Alliance for further information.

The PXI Systems Alliance wants to receive your comments on this specification. Visit the PXI Systems Alliance web site at `http://www.pxisa.org/` for contact information and to learn more about the PXI Systems Alliance.

The attention of adopters is directed to the possibility that compliance with or adoption of the PXI Systems Alliance specifications may require use of an invention covered by patent rights. The PXI Systems Alliance shall not be responsible for identifying patents for which a license may be required by any PXI Systems Alliance specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. PXI Systems Alliance specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

The information contained in this document is subject to change without notice. The material in this document details a PXI Systems Alliance specification in accordance with the license and notices set forth on this page. This document does not represent a commitment to implement any portion of this specification in any company's products.

The PXI Systems Alliance makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The PXI Systems Alliance shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Compliance with this specification does not absolve manufacturers of PXI equipment from the requirements of safety and regulatory agencies (UL, CSA, FCC, IEC, etc.).

Trademarks

PXI™ is a trademarks of the PXI Systems Alliance.

PICMG™ and CompactPCI® are trademarks of the PCI Industrial Computation Manufacturers Group.

Product and company names are trademarks or trade names of their respective companies.

PXI Software Specification Revision History

This section is an overview of the revision history of the PXI Software Specification.

Revision 2.1, February 4, 2003

This is the first public revision of the PXI specification.

Contents

1. Introduction

This section explains the objectives and scope of the PXI Software Specification. It also describes the intended audience and lists relevant terminology and documents. Note that this specification is intended to supplement the PXI Hardware Specification. Refer to the PXI Hardware Specification for general background on PXI and its electrical and mechanical requirements.

1.1 Objectives

The PXI software architecture, in addition to PXI's mechanical and electrical requirements, is a key component in furthering the standard's interoperability and ease of integration goals. The *PXI Software Specification* was created to supplement the *PXI Hardware Specification* in clarifying and addressing common software requirements in PXI systems. The software specification's purposes are to describe the capabilities of PXI hardware components using standard hardware description files and to promote interoperability among PXI vendors with respect to software requirements. The software specification addresses a variety of issues, including hardware description, hardware resource management, operating system framework definition, and the incorporation of existing instrumentation software standards.

The primary objective of the *PXI Software Specification* is to define a set of hardware description files for characterizing PXI components and their capabilities. Using standard file formats, device drivers, configuration software, and systems integrators can implement ease-of-use features such as geographic slot identification and chassis identification. These hardware description files can also serve as a repository for managing PXI hardware resources, including the PXI trigger bus, the PXI star trigger, and the PXI local bus.

A secondary objective of the *PXI Software Specification* is to define standard operating system frameworks and to incorporate existing instrumentation software standards. Additional software requirements include the support of standard operating system frameworks such as Windows 9*x*/Me, Windows NT, Windows 2000, and Windows XP, and the support of instrumentation software standards developed by the VXI*plug&play* Systems Alliance (VISA).

1.2 Intended Audience and Scope

This specification is primarily intended for product developers interested in implementing and leveraging software features of the PXI platform. Hardware developers will be interested in using hardware description files for identifying and describing the capabilities of PXI hardware products such as chassis and system controller modules. Likewise, software developers and systems integrators should take advantage of hardware description files to manage PXI resources, including PXI triggers and the PXI local bus, and to implement features such as slot identification and chassis identification. Additionally, product developers and systems integrators should reference the operating system framework definitions to ensure system-level interoperability.

1.3 Background and Terminology

This section defines the acronyms and key words that are referred to throughout this specification. This specification uses the following acronyms:

- **API—**Application Programming Interface
- **CompactPCI** – PICMG 2.0 Specification
- **PCI—**Peripheral Component Interconnect; electrical specification defined by PCISIG
- **PCISIG—**PCI Special Interest Group
- **PICMG—**PCI Industrial Computer Manufacturers Group
- **PXI—**PCI eXtensions for Instrumentation
- **VISA—**Virtual Instrument Software Architecture

- **VPP—**VXI*plug&play* Specification

This specification uses several key words, which are defined as follows:

RULE: Rules SHALL be followed to ensure compatibility. A rule is characterized by the use of the words SHALL and SHALL NOT.

RECOMMENDATION: Recommendations consist of advice to implementers that will affect the usability of the final module. A recommendation is characterized by the use of the words SHOULD and SHOULD NOT.

PERMISSION: Permissions clarify the areas of the specification that are not specifically prohibited. Permissions reassure the reader that a certain approach is acceptable and will cause no problems. A permission is characterized by the use of the word MAY.

OBSERVATION: Observations spell out implications of rules and bring attention to things that might otherwise be overlooked. They also give the rationale behind certain rules, so that the reader understands why the rule must be followed.

MAY: A key word indicating flexibility of choice with no implied preference. This word is usually associated with a permission.

SHALL: A key word indicating a mandatory requirement. Designers SHALL implement such mandatory requirements to ensure interchangeability and to claim conformance with the specification. This word is usually associated with a rule.

SHOULD: A key word indicating flexibility of choice with a strongly preferred implementation. This word is usually associated with a recommendation.

1.4 Applicable Documents

This specification defines extensions to the base PCI and CompactPCI specifications referenced in this section. It is assumed that the reader has a thorough understanding of PCI and CompactPCI. The CompactPCI specification refers to several other applicable documents with which the reader may want to become familiar. This specification refers to the following documents directly:

- *PXI Hardware Specification* (current revision)
- *PCI Local Bus Specification*, Rev. 2.2
- *PICMG 2.0 R3.0 CompactPCI Specification*

2. Hardware Description Files

This section defines the formats of the hardware description files and describes their use.

2.1 Overview

The *PXI Hardware Specification* allows many variations of chassis and system controller modules. While many PCI hardware capabilities are self-describing (that is, their identities and capabilities can be determined using standard PCI hardware enumeration techniques), there is no standard hardware mechanism for identifying and managing many of the resources in a PXI system. The *PXI Software Specification* solves this problem by defining a set of hardware description files for PXI systems and the components that comprise them.

A primary goal of PXI's hardware description files is to enable application and device driver software to identify components based on their geographic characteristics (that is, chassis number and slot number) rather than their less user-friendly PCI logical address characteristics (PCI bus number, device number, and function number). For example, using a PXI system description file as a lookup table, an application or driver can map between a module's location on the PCI bus and its physical location in a PXI chassis. This functionality enables operators to quickly and easily distinguish between several similar modules using the chassis number and slot number.

Another goal of the hardware description files is to serve as a repository for managing PXI platform resources. PXI triggers, for example, are a shared hardware resource, and the trigger lines must be managed by a central reservation facility to guarantee the prevention of resource conflicts. Similarly, PXI's local bus lines are managed by software to guarantee that adjacent PXI modules do not use the local bus in a conflicting manner. In both of these cases, hardware description files serve as standard data storage for describing and managing these shared resources.

PXI hardware descriptions are contained in `.ini` files, which consist of ASCII text. The `.ini` file format is useful because it is both human readable and easily parsed by application and driver software.

The *PXI Software Specification* defines two hardware description file formats: system description files and chassis description files. The system description file is used to describe an overall PXI system and the components that comprise it. It is a collection of information obtained from several sources, including other hardware description files. The chassis description file is used to describe a PXI chassis and its features. Both of these file formats are described in detail below.

2.2 Common File Requirements

All hardware description files are `.ini` files. Each `.ini` file contains one or more sections, and each section contains one or more tag lines. Each tag line describes a specific property of the section.

In the context of PXI hardware description files, `.ini` file sections form *descriptors*. Descriptors describe PXI systems and the components that comprise them. Descriptors always correspond to unique `.ini` file sections.

RULE: Each `.ini` file SHALL contain only ASCII text.

RULE: Each `.ini` file SHALL contain only the following types of lines:
comment lines, section headers, or tag lines.

RULE: A comment line SHALL begin with the '#' character.

RULE: A section header line SHALL begin with the '[' character and end with the ']' character. Text between the two brackets SHALL identify the type of section.

RULE: A tag line SHALL consist of the following three fields:
tag, '=' character, and value. The three fields SHALL be separated by a single space character.

.ini File Format Example

```
# This line is a comment.
[Section1]
ExampleTag = 1
IsSpecialSectionTag = No

[Section2]
ExampleTag = 2
IsSpecialSectionTag = Yes
```

2.2.1 Version Descriptor

PXI hardware description files include a version descriptor section. The version descriptor allows software to distinguish between `.ini` file formats as the *PXI Software Specification* evolves. For information on backward compatibility, refer to the *Backward Compatibility with Previous PXI Specifications* section below.

RULE: A hardware description file SHALL include a single version descriptor.

RULE: A version descriptor `.ini` section SHALL be named "Version".

RULE: Each version descriptor section SHALL contain one of each tag line type described in Table 2-1.

Table 2-1. Version Information Tag Line Descriptions

Tag	Valid Values	Description
Major	*x*, where *x* is a positive decimal integer.	This field indicates the major version number of a version *x.y*, where *x* is the major number and *y* is the minor number of the *PXI Software Specification* version that this file complies with.
Minor	*y*, where *y* is a positive decimal integer.	This field indicates the major version number of a version *x.y*, where *x* is the major number and *y* is the minor number of the *PXI Software Specification* version that this file complies with.

Version Descriptor Example

```
[Version]
Major = 2
Minor = 1
```

OBSERVATION: A version descriptor is useful for identifying the *PXI Software Specification* file format that a hardware description file complies with.

2.2.2 Backward Compatibility with Previous PXI Specifications

Backward compatibility is a key feature of the *PXI Software Specification*, and hardware descriptions files must be structured so that compatibility with previous PXI specification revisions can be achieved.

Beginning with the *PXI Software Specification*, Revision 2.1, the format of each type of hardware description file has been modified so that new features, such as multi-chassis PXI systems and multi-segmented PXI chassis, can be accurately described. To maintain backward compatibility, the following applies:

PERMISSION: In addition to the format defined in this specification, A PXI System Description file MAY include sections in the format of PXI Specifications prior to revision 2.1 of the *PXI Software Specification*.

Sections of the PXI System Description file that are in the format of specification revisions prior to version 2.1 are referred to as *legacy sections*.

OBSERVATION: None of the section headings in this specification overlap with headings defined in previous specifications. Multiple versions of the PXI System Description file format may exist together in the same System Description file.

RECOMMENDATION: A software tool that generates a PXI System Description file compliant with this specification SHOULD allow the user to generate the legacy file format also.

RULE: Even if a PXI System Description file includes the legacy sections, it MUST include the sections defined in this specification.

OBSERVATION: For a System Description file to accurately describe a majority of PXI systems in use today, it must use the format defined in this specification. The legacy PXI System Description format cannot sufficiently describe modern PXI systems.

2.3 System Description Files

System description files describe PXI systems and their components. The system controller module and the one or more chassis that comprise a PXI system determine a system description. A system description enables a variety of software functionality, including geographic slot identification. Chassis description files, from which much of the system description content is derived, are discussed later in this chapter.

2.3.1 System Description Definitions

To develop a system description, it is useful to define descriptors for the following PXI system components:

- **System** – A PXI System descriptor corresponds to a physical PXI system. A PXI System is a collection of chassis. Multiple chassis in a system are coupled in a software-transparent manner (that is, they are coupled via PCI-PCI bridging).
    - **Chassis** – A chassis descriptor corresponds to a physical PXI chassis in a system. Chassis can include PCI bus segments, trigger buses, star triggers, and slots.
        - **PCI Bus Segment** – A PCI bus segment descriptor corresponds to a distinct, physical PCI bus in a chassis. PCI bus segments can contain slots, bridges, and other backplane devices. Multiple PCI bus segments are linked within a chassis using PCI-PCI bridging.
        - **Trigger Bus** – A PXI trigger bus descriptor corresponds to a physical trigger bus in a chassis. A trigger bus is characterized by a list of slots sharing the physical trigger bus connection. Chassis can contain multiple trigger buses.
        - **Star Triggers –** A PXI star trigger descriptor corresponds to a physical set of star triggers in a chassis. A set of star triggers is characterized by a star trigger controller slot number and a mapping of PXI_STAR lines (defined in the *PXI Hardware Specification*) to peripheral slot numbers. A chassis can contain multiple sets of star triggers.
        - **Slot** –A PXI slot descriptor corresponds to a physical slot in a chassis. A slot is characterized by a geographic address, a PCI logical address, local bus routings, and other special capabilities. A chassis has multiple slots.

In addition, a *Resource Manager* is defined as the entity responsible for creating a PXI system description file. For example, the responsibilities of a Resource Manager might be accomplished by a systems integrator, or a software utility might be provided to automate the Resource Manager algorithm.

RULE: A system controller module manufacturer SHALL provide either a system description file for each supported system configuratiovn or a Resource Manager utility that can manage the system description file.

RECOMMENDATION: A system controller module manufacturer SHOULD provide a utility that can automate the Resource Manager algorithm.

RULE: A system description file SHALL be named `pxisys.ini`. The `pxisys.ini` file SHALL be located in the `<windows>` directory (for example, `c:\windows or c:\winnt`).

RECOMMENDATION: To aid systems integrators and operators, PXI module configuration and driver software SHOULD use geographic addressing information, available in a PXI system description file, to present chassis and slot locations for PXI modules via a user interface.

2.3.2 System Descriptor

The system descriptor contains highest-level information about a PXI system. PXI systems are characterized by the chassis that comprise the system, and the system descriptor contains a list of these chassis.

RULE: A system description file SHALL contain one and only one system descriptor.

RULE: The system descriptor `.ini` section header SHALL be named “System”.

RULE: Each system descriptor section SHALL contain one of each tag line types described in Table 2-2.

Table 2-2. System Description File - System Tag Line Descriptions

Tag	Valid Values	Description
Chassis List	A comma-separated list of *n*, where *n* is a decimal integer such that *n* >= 1.	This tag enumerates the chassis in a PXI system.

System Descriptor Example

```
# This section describes a PXI system with two chassis.
[System]
ChassisList = 1,2
```

RULE: PXI chassis, specified with the ChassisList tag, SHALL be enumerated by a Resource Manager when collecting information regarding each chassis in the PXI system.

OBSERVATION: A Resource Manager can enumerate chassis using a variety of mechanisms. For example, a Resource Manager utility can present a user interface, allowing a user to identify the types of chassis included in the system.

RULE: Multiple chassis SHALL be uniquely numbered in the ChassisList tag.

OBSERVATION: Chassis can be numbered in an arbitrary fashion. For example, chassis can be numbered according to their order of discovery using a depth-first PCI traversal algorithm.

2.3.3 Chassis Descriptor

A chassis descriptor provides a high-level description of an individual PXI chassis in a system. A chassis descriptor contains collections of the components that comprise a chassis, including PCI bus segments, trigger buses, sets of star triggers, and slots.

RULE: A system description file SHALL contain a distinct chassis descriptor for each physical chassis that comprises the PXI system.

OBSERVATION: Chassis are enumerated using a system descriptor's ChassisList tag.

RULE: A chassis descriptor SHALL be named "Chassis*N*", where *N* is the chassis number.

RULE: A Resource Manager SHALL derive chassis numbers from the ChassisList tag of a system descriptor (see Table 2-2).

RECOMMENDATION: The chassis number SHOULD be physically viewable on a chassis to assist operators in locating peripheral modules.

RULE: Each chassis descriptor SHALL contain one of each of tag line type described in Table 2-3.

Table 2-3. System Description File - Chassis Tag Line Description

Tag	Valid Values	Description
PCIBusSegmentList	A comma-separated list of *n*, where *n* is a decimal integer such that 1 <= *n* <= 255.	This tag enumerates the PCI bus segments in a chassis.
TriggerBusList	A comma-separated list of *n*, where *n* is a decimal integer such that *n* >= 1.	This tag enumerates the trigger buses in a chassis.
StarTriggerList	A comma-separated list of *n*, where *n* is a decimal integer such that *n* >= 1.	This tag enumerates the sets of star triggers in a chassis.
SlotList	A comma-separated list of *n*, where *n* is a decimal integer such that *n* >= 1.	This tag enumerates the slots in a chassis.
Model	A string indicating the model name for this chassis.	This tag identifies a chassis model name.
Vendor	A string indicating the vendor name for this chassis.	This tag identifies a chassis vendor name.

Chassis Descriptor Example

```
# This example describes a 3-segment, 18-slot PXI chassis
[Chassis1]
PCIBusSegmentList = 1,2,3
TriggerBusList = 1,2,3
StarTriggerList = 1
SlotList = 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18
Model = "Example 18-Slot Chassis"
Vendor = "PXISA"
```

RULE: A Resource Manager SHALL derive the tag values in Table 2-3 from the tag values of the corresponding chassis description file's chassis descriptor (see Table 2-8).

2.3.4 PCI Bus Segment Descriptor

A PCI bus segment descriptor describes an individual PCI bus segment in a chassis.

RULEl: A system description file SHALL contain a distinct PCI bus segment descriptor for each physical PCI bus segment in the system.

RULE: A PCI bus segment descriptor SHALL be named "Chassis*M*PCIBusSegment*N*", where *M* is the chassis number, and *N* is the PCI bus segment number.

RULE: A Resource Manager SHALL derive PCI bus segment numbers from the PCIBusSegmentList tag of the corresponding chassis descriptor (see Table 2-3).

OBSERVATION: While each PCI bus segment number will uniquely correspond to a PCI bus number, the PCI bus segment number will not necessarily be equal to the corresponding PCI bus number.

RULE: Each PCI bus segment descriptor SHALL contain one of each of tag line type described in Table 2-4.

Table 2-4. System Description File – PCI Bus Segment Tag Line Descriptions

Tag	Valid Values	Description
SlotList	A comma-separated list of *n*, where *n* is a decimal integer such that *n* >= 1.	This tag enumerates the physical slots on a PCI bus segment.

PCI Bus Segment Descriptor Example

```
# This example describes the third bus segment of
# an 18-slot PXI chassis
[Chassis1PCIBusSegment3]
SlotList = 13,14,15,16,17,18
```

RULE: A Resource Manager SHALL derive the tag values in Table 2-4 from the tag values of the corresponding chassis description file's PCI Bus Segment descriptor (see Table 2-9).

PERMISSION: A PCI bus segment descriptor that describes a segment with no PXI slots will contain an empty slot list. In this case, the PCI bus segment descriptor MAY be excluded from the system description file.

2.3.5 Trigger Bus Descriptor

A trigger bus descriptor describes an individual trigger bus in a PXI chassis. A trigger bus is characterized by a list of slots that reside on the trigger bus.

RULE: A system description file SHALL contain a distinct PXI trigger bus descriptor for each physical PXI trigger bus in the system.

RULE: A trigger bus descriptor SHALL be named "Chassis*M*TriggerBus*N*", where *M* is the chassis number and *N* is the trigger bus number.

RULE: A Resource Manager SHALL derive trigger bus numbers from the TriggerBusList tag of the corresponding chassis descriptor (see Table 2-3).

OBSERVATION: While each trigger bus number will uniquely correspond to a set of PXI slots, there is not necessarily a one-to-one correspondence between trigger buses and PCI bus segments.

RULE: Each trigger bus descriptor SHALL contain one of each of the tag line types described in Table 2-5.

Table 2-5. System Description File - Trigger Bus Tag Line Descriptions

Tag	Valid Values	Description
SlotList	A comma-separated list of *n*, where *n* is a decimal integer such that *n* >= 1.	This tag enumerates the slots on a trigger bus.

Trigger Bus Descriptor Example

```
# This example describes the first trigger bus of a
# 3-segment, 18-slot chassis.
[Chassis1TriggerBus1]
SlotList = 1,2,3,4,5,6
```

RULE: A Resource Manager SHALL derive the tag values in Table 2-5 from the tag values of the corresponding chassis description file's Trigger Bus descriptor (see Table 2-10).

2.3.6 Star Trigger Descriptor

A star trigger descriptor describes an individual set of star triggers in a PXI chassis. A star trigger descriptor is characterized by a star trigger controller slot number and a mapping of PXI_STAR lines, as defined in the *PXI Hardware Specification*, to peripheral slot numbers.

RULE: A system description file SHALL contain a distinct PXI star trigger descriptor for each physical set of PXI star triggers in the system.

RULE: A trigger bus descriptor SHALL be named "Chassis*M*StarTrigger*N*", where *M* is the chassis number and *N* is the number for the set of star triggers.

RULE: A Resource Manager SHALL derive star trigger descriptor numbers from the StarTriggerList tag of the corresponding chassis descriptor (see Table 2-3).

RULE: Each star trigger descriptor SHALL contain one of each of the tag line types described in Table 2-6.

Table 2-6. System Description File - Star Trigger Tag Line Descriptions

Tag	Valid Values	Description
ControllerSlot	A decimal integer *n*, where *n* is a decimal integer such that *n* >= 1.	This tag specifies the star trigger controller slot number for a PXI_STAR lines in a set of star triggers.
PXI_STAR*n* (where *n* is a decimal integer such that 0 <= *n* <= 12), for each PXI star trigger line physically routed to a PXI slot	A comma-separated list of *m*, where *m* is a decimal integer, corresponding to a PXI slot number, such that *m* >= 2.	This tag specifies the PXI_STAR line to slot mapping for a set of star triggers.

Star Trigger Descriptor Example

```
# This example describes a set of star triggers for a
# 3-segment, 18-slot chassis.
[Chassis1StarTrigger1]
ControllerSlot = 2
PXI_STAR0 = 3
PXI_STAR1 = 4
PXI_STAR2 = 5
PXI_STAR3 = 6
PXI_STAR4 = 7
PXI_STAR5 = 8
PXI_STAR6 = 9
PXI_STAR7 = 10
PXI_STAR8 = 11
PXI_STAR9 = 12
PXI_STAR10 = 13
PXI_STAR11 = 14
PXI_STAR12 = 15
```

RULE: A Resource Manager SHALL derive the tag values in Table 2-6 from the tag values of the corresponding chassis description file's Star Trigger descriptor (see Table 2-11).

OBSERVATION: The star trigger descriptor allows configuration software to describe alternative star trigger line mappings.

OBSERVATION: If a star trigger line is not routed to a PXI slot, the corresponding PXI_STAR*n* tag will not be listed in the star trigger bus descriptor.

2.3.7 Slot Descriptor

A slot descriptor describes an individual slot in a chassis. A slot descriptor is characterized by the features of the slot it describes, including routing information for the slot's local bus lines and the PCI logical address for a module that might occupy the slot. The slot descriptor serves as a lookup facility for applications and driver software interested in geographic slot identification.

RULE: A system description file SHALL contain a distinct slot descriptor for each physical slot in the PXI system.

RULE: A slot descriptor SHALL be named "Chassis*M*Slot*N*", where *M* is the chassis number, and *N* is the physical slot number.

RULE: A Resource Manager SHALL derive slot numbers from the SlotList tag of the corresponding chassis descriptor (see Table 2-3).

RULE: Each slot descriptor SHALL contain one of each of non-shaded tag line type described in Table 2-7.

PERMISSION: Application and device driver software MAY continue to use the shaded fields of Table 2-7. These fields may be removed in a future revision.

RECOMMENDATION: New software development SHOULD use the non-shaded fields.

Table 2-7. System Description File - Slot Tag Line Descriptions

Tag	Valid Values	Description
PCISlotPath	A comma-separated list of *n*, where *n* is a hexadecimal integer indicating the slot path of this PXI slot.	This tag indicates the PCI slot path for a slot.
LocalBusLeft	A valid slot descriptor. A valid star trigger descriptor. (Other).	This tag indicates how a slot routes its local bus pins to the left.
LocalBusRight	A valid slot descriptor. (Other).	This tag indicates how a slot routes its local bus pins to the right.
PCIBusNumber	*n*, where *n* is a decimal integer such that 0 <= *n* <= 255.	This tag indicates the PCI bus number for a slot.
PCIDeviceNumber	*n*, where *n* is a decimal integer such that 0 <= *n* <= 31.	This tag indicates the PCI device number for a slot.
ExternalBackplaneInterface	None. (Other).	If a slot routes to an external backplane interface, this tag specifies the name of that interface.

Slot Descriptor Example

```
# This example describes Slot 2 of an 8-slot PXI chassis.
[Chassis1Slot2]
# To calculate the slot path, we note that this chassis sits
# behind a PCI-PCI bridge residing on PCI bus 0 at PCI
# device 17
PCISlotPath = 98,88
PCIBusNumber = 2
PCIDeviceNumber = 19
LocalBusLeft = StarTrigger1
LocalBusRight = Chassis1Slot3
```

RULE: A Resource Manager SHALL derive the LocalBusLeft, LocalBusRight, and ExternalBackplaneInterface tag values from the tag values in the corresponding chassis description file's slot descriptor (see Table 2-13).

RULE: A PXI Resource Manager SHALL derive the PCISlotPath, PCIBusNumber, and PCIDeviceNumber tag values from the controller IDSEL routing information, the PCI bus segment IDSEL routing information (see Table 2-9), and the PCI bus hierarchy.

OBSERVATION: A PXI slot that does not implement the full set of PXI features, such as a CompactPCI-only slot, will have tag values corresponding to PXI features set to "None". For example, a CompactPCI-only slot descriptor would have LocalBusLeft and LocalBusRight tags set to "None". In addition, this slot would not be present in the SlotList for a trigger bus, and it would not belong to a set of star triggers.

2.3.7.1 PCI Slot Path

To facilitate geographic slot identification, it is useful to introduce the concept of a PCI slot path. The purpose of a PCI slot path is to describe the PCI bus hierarchy in a manner independent of the PCI bus number. PCI slot paths are a sequence of hexadecimal values representing the PCI device number and function number of a PCI module and each parent PCI bridge that routes the module to the host PCI bridge (bus 0). Each byte of a slot path corresponds to the PCI BIOS device/function number encoding for the current bridge in the path. The encoding is calculated as follows:

PCI Slot Path Byte = (PCI Device Number << 3) | PCI Function Number

PCI Slot Path Example

Consider a PXI slot located on PCI bus #2, device #17d. This slot is subordinate to a PCI-PCI bridge with primary bus #0, device #14d. The slot path for this device would be “88,70”, that is, ((17d << 3) | 0) followed by ((14d << 3) | 0).

OBSERVATION: Slot paths are useful because they change less frequently than PCI bus numbers. In PXI systems, if a PCI bus number changes, the PCI slot path for slots on that segment do not necessarily change.

RECOMMENDATION: Slot paths SHOULD be used for mapping between PCI logical addresses to PXI geographic addresses.

2.3.7.2 Local Bus Routings

The slot descriptor provides a means for specifying how the local bus lines are routed for a given slot.

OBSERVATION: The LocalBusLeft and LocalBusRight tags will usually specify the slot descriptor for the slot to the left and right of the current slot, respectively.

OBSERVATION: If a slot does not route its local bus pins (left or right) to a neighboring slot’s local bus, the LocalBus tags can be used to specify a special slot capability.

OBSERVATION: The LocalBusLeft tag for PXI slot 2 can be used to specify the Star Trigger capability, pointing to a chassis’ StarTrigger descriptor.

OBSERVATION: The LocalBusRight tag to the rightmost slot in a chassis can be used to specify a connection to an external backplane interface.

RECOMMENDATION: The LocalBusLeft and LocalBusRight tags SHOULD be used to specify an external backplane connection. The ExternalBackplaneInterface tag is provided for backward compatibility only.

2.3.8 System Description File Example

```
# This example describes a PXI system with two chassis.
# The first chassis (Chassis1) has a single PXI bus
# segment and 8 PXI slots, described by the chassis
# description file in Example 2.4.8.1.
# The second chassis (Chassis2) has 3 PXI bus
# segments and 18 PXI slots, described by the chassis
# description file in Example 2.4.8.2.

# Assumptions:
# The two chassis (Chassis1 and Chassis2) are linked
```

```
# together using PXI-PXI bridging.  The first chassis
# in the daisy-chain (Chassis1) contains a system
# controller module with a PCI-PCI bridge at PCI bus #0,
# device #30, function #0.  Its corresponding PCI slot
# path node is 0xF0.  This bridge forms the PXI bus
# segment (PCI bus #1) for Chassis1.  The PXI-PXI bridge
# resides in slot #5 (PCI bus #1, device #12, function #0)
# of Chassis1.  This split-bridge forms the first PXI
# segment of Chassis2 (PCI bus #3).  Its corresponding
# PCI slot path node is 0x60.  Chassis2 contains three
# PXI bus segments.  The first segment contains a PCI-PCI
# bridge at PCI bus #3 device #12, function #0.  Its
# corresponding PCI slot path node is 0x60.  This
# bridge forms the second PXI bus segment (PCI bus #4).
# The second segment contains a PCI-PCI bridge at
# PCI bus #4, device #12, function #0.  Its corresponding
# PCI slot path node is 0x60.  This bridge forms the third
# PXI bus segment.

[Version]
Major = 2
Minor = 1

[PXI System]
ChassisList = 1,2

[Chassis1]
Model = "Example 8-Slot Chassis"
Vendor = "PXISA"
PCIBusSegmentList = 1
SlotList = 1,2,3,4,5,6,7,8
TriggerBusList = 1
StarTriggerList = 1

[Chassis1StarTrigger1]
ControllerSlot = 2
PXI_STAR0 = 3
PXI_STAR1 = 4
PXI_STAR2 = 5
PXI_STAR3 = 6
PXI_STAR4 = 7
PXI_STAR5 = 8

[Chassis1PCIBusSegment1]
SlotList = 1,2,3,4,5,6,7,8

[Chassis1TriggerBus1]
SlotList = 1,2,3,4,5,6,7,8

[Chassis1Slot1]
PCISlotPath = None
PCIBusNumber = None
PCIDeviceNumber = None
LocalBusLeft = None
```

```
LocalBusRight = None
ExternalBackplaneInterface = None

[Chassis1Slot2]
PCISlotPath = 78,F0
PCIBusNumber = 1
PCIDeviceNumber = 15
LocalBusLeft = StarTrigger1
LocalBusRight = Slot3
ExternalBackplaneInterface = None

[Chassis1Slot3]
PCISlotPath = 70,F0
PCIBusNumber = 1
PCIDeviceNumber = 14
LocalBusLeft = Slot2
LocalBusRight = Slot4
ExternalBackplaneInterface = None

[Chassis1Slot4]
PCISlotPath = 68,F0
PCIBusNumber = 1
PCIDeviceNumber = 13
LocalBusLeft = Slot3
LocalBusRight = Slot5
ExternalBackplaneInterface = None

[Chassis1Slot5]
PCISlotPath = 60,F0
PCIBusNumber = 1
PCIDeviceNumber = 12
LocalBusLeft = Slot4
LocalBusRight = Slot6
ExternalBackplaneInterface = None

[Chassis1Slot6]
PCISlotPath = 58,F0
PCIBusNumber = 1
PCIDeviceNumber = 11
LocalBusLeft = Slot5
LocalBusRight = Slot7
ExternalBackplaneInterface = None

[Chassis1Slot7]
PCISlotPath = 50,F0
PCIBusNumber = 1
PCIDeviceNumber = 10
LocalBusLeft = Slot6
LocalBusRight = Slot8
ExternalBackplaneInterface = None

[Chassis1Slot8]
PCISlotPath = 48,F0
PCIBusNumber = 1
```

```
PCIDeviceNumber = 9
LocalBusLeft = Slot7
LocalBusRight = None
ExternalBackplaneInterface = None

[Chassis2]
Model = "Example 18-Slot Chassis"
Vendor = "PXISA"
PCIBusSegmentList = 1,2,3
SlotList = 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18
TriggerBusList = 1,2,3
StarTriggerList = 1

[Chassis2StarTrigger1]
ControllerSlot = 2
PXI_STAR0 = 3
PXI_STAR1 = 4
PXI_STAR2 = 5
PXI_STAR3 = 6
PXI_STAR4 = 7
PXI_STAR5 = 8
PXI_STAR6 = 9
PXI_STAR7 = 10
PXI_STAR8 = 11
PXI_STAR9 = 12
PXI_STAR10 = 13
PXI_STAR11 = 14
PXI_STAR12 = 15

[Chassis2PCIBusSegment1]
SlotList = 1,2,3,4,5,6

[Chassis2TriggerBus1]
SlotList = 1,2,3,4,5,6

[Chassis2Slot1]
PCISlotPath = None
PCIBusNumber = None
PCIDeviceNumber = None
LocalBusLeft = None
LocalBusRight = None
ExternalBackplaneInterface = None

[Chassis2Slot2]
PCISlotPath = 78,60,F0
PCIBusNumber = 3
PCIDeviceNumber = 15
LocalBusLeft = StarTrigger1
LocalBusRight = Slot3
ExternalBackplaneInterface = None

[Chassis2Slot3]
PCISlotPath = 70,60,F0
PCIBusNumber = 3
```

```
PCIDeviceNumber = 14
LocalBusLeft = Slot2
LocalBusRight = Slot4
ExternalBackplaneInterface = None

[Chassis2Slot4]
PCISlotPath = 68,60,F0
PCIBusNumber = 3
PCIDeviceNumber = 13
LocalBusLeft = Slot3
LocalBusRight = Slot5
ExternalBackplaneInterface = None

[Chassis2Slot5]
PCISlotPath = 58,60,F0
PCIBusNumber = 3
PCIDeviceNumber = 11
LocalBusLeft = Slot4
LocalBusRight = Slot6
ExternalBackplaneInterface = None

[Chassis2Slot6]
PCISlotPath = 50,60,F0
PCIBusNumber = 3
PCIDeviceNumber = 10
LocalBusLeft = Slot5
LocalBusRight = Slot7
ExternalBackplaneInterface = None

[Chassis2PCIBusSegment2]
SlotList = 7,8,9,10,11,12

[Chassis2TriggerBus2]
SlotList = 7,8,9,10,11,12

[Chassis2Slot7]
PCISlotPath = 78,60,60,F0
PCIBusNumber = 4
PCIDeviceNumber = 15
LocalBusLeft = Slot6
LocalBusRight = Slot8
ExternalBackplaneInterface = None

[Chassis2Slot8]
PCISlotPath = 70,60,60,F0
PCIBusNumber = 4
PCIDeviceNumber = 14
LocalBusLeft = Slot7
LocalBusRight = Slot9
ExternalBackplaneInterface = None

[Chassis2Slot9]
PCISlotPath = 68,60,60,F0
PCIBusNumber = 4
```

```
PCIDeviceNumber = 13
LocalBusLeft = Slot8
LocalBusRight = Slot10
ExternalBackplaneInterface = None

[Chassis2Slot10]
PCISlotPath = 58,60,60,F0
PCIBusNumber = 4
PCIDeviceNumber = 11
LocalBusLeft = Slot9
LocalBusRight = Slot11
ExternalBackplaneInterface = None

[Chassis2Slot11]
PCISlotPath = 50,60,60,F0
PCIBusNumber = 4
PCIDeviceNumber = 10
LocalBusLeft = Slot10
LocalBusRight = Slot12
ExternalBackplaneInterface = None

[Chassis2Slot12]
PCISlotPath = 48,60,60,F0
PCIBusNumber = 4
PCIDeviceNumber = 9
LocalBusLeft = Slot11
LocalBusRight = Slot13
ExternalBackplaneInterface = None

[Chassis2PCIBusSegment3]
SlotList = 13,14,15,16,17,18

[Chassis2TriggerBus3]
SlotList = 13,14,15,16,17,18

[Chassis2Slot13]
PCISlotPath = 78,60,60,60,F0
PCIBusNumber = 5
PCIDeviceNumber = 15
LocalBusLeft = Slot12
LocalBusRight = Slot14
ExternalBackplaneInterface = None

[Chassis2Slot14]
PCISlotPath = 70,60,60,60,F0
PCIBusNumber = 5
PCIDeviceNumber = 14
LocalBusLeft = Slot13
LocalBusRight = Slot15
ExternalBackplaneInterface = None

[Chassis2Slot15]
PCISlotPath = 68,60,60,60,F0
PCIBusNumber = 5
```

```
PCIDeviceNumber = 13
LocalBusLeft = Slot14
LocalBusRight = Slot16
ExternalBackplaneInterface = None

[Chassis2Slot16]
PCISlotPath = 60,60,60,60,F0
PCIBusNumber = 5
PCIDeviceNumber = 12
LocalBusLeft = Slot15
LocalBusRight = Slot17
ExternalBackplaneInterface = None

[Chassis2Slot17]
PCISlotPath = 58,60,60,60,F0
PCIBusNumber = 5
PCIDeviceNumber = 11
LocalBusLeft = Slot16
LocalBusRight = Slot18
ExternalBackplaneInterface = None

[Chassis2Slot18]
PCISlotPath = 50,60,60,60,F0
PCIBusNumber = 5
PCIDeviceNumber = 10
LocalBusLeft = Slot17
LocalBusRight = None
ExternalBackplaneInterface = None
```

2.4 Chassis Description Files

Chassis description files characterize PXI chassis. The primary purpose of a chassis description file is to enumerate PCI bus segments, trigger buses, sets of star triggers, and slots. Chassis description files are a key component in the PXI hardware description architecture, enabling a Resource Manager to generate a PXI system description.

RULE: A chassis manufacturer SHALL provide a chassis description file for each chassis model produced.

RULE: A chassis description file SHALL be named `chassis_`*`vendorDefinedText`*`.ini`, where *`vendorDefinedText`* is a vendor-defined string used to uniquely name a chassis description file.

RULE: A system controller module SHALL provide the following Windows registry value for specifying a location of chassis description files:

Key: HKEY_LOCAL_MACHINE\SOFTWARE\PXISA\CurrentVersion

Value: ChassisDescriptionFilePath

The ChassisDescriptionFilePath SHALL be a string value that specifies the complete path of a directory that holds chassis description files.

PERMISSION: A system controller module MAY place chassis description files in any location, provided that the ChassisDescriptionFilePath registry value correctly points to this location.

OBSERVATION: Using the ChassisDescriptionFilePath registry value, chassis description file installers can copy their chassis description files to a standard location. In addition, a PXI Resource Manager can use this location to identify the types of chassis available for a PXI system.

2.4.1 Chassis Description Definitions

To develop a chassis description, it is useful to define descriptors for the following chasis components:

- **Chassis** – A chassis descriptor corresponds to a physical PXI chassis. Chassis can include PCI bus segments, trigger buses, star triggers, and slots.
  - **PCI Bus Segment** – A PCI bus segment descriptor corresponds to a distinct, physical PCI bus in a chassis. PCI bus segments can contain slots, bridges, and other backplane devices. Multiple PCI bus segments are linked within a chassis using PCI-PCI bridging.
  - **Trigger Bus** – A PXI trigger bus descriptor corresponds to a physical trigger bus in a PXI chassis. A trigger bus is characterized by a list of slots sharing the physical trigger bus connection. Chassis can contain multiple trigger buses.
  - **Star Triggers –** A PXI star trigger descriptor corresponds to a physical set of star triggers in a chassis. A set of star triggers is characterized by a star trigger controller slot number and a mapping of PXI_STAR lines to peripheral slot numbers. A chassis can contain multiple sets of star triggers.
  - **Bridge** – A bridge descriptor corresponds to a physical PCI-PCI bridge. A bridge is characterized by a secondary bus segment (that is, the PCI bus segment subordinate to the bridge). A chassis can have multiple bridges.
  - **Slot** –A PXI slot descriptor corresponds to a physical slot in a chassis. A slot is characterized by a geographic address, a PCI logical address, local bus routings, and other special capabilities. A chassis has multiple slots.

2.4.2 Chassis Descriptor

A chassis descriptor provides a high-level description of a PXI chassis. A chassis descriptor contains collections of the components that comprise a chassis, including PCI bus segments, trigger buses, sets of star triggers, and slots.

RULE: A chassis description file SHALL contain one and only one chassis descriptor.

RULE: The chassis descriptor section SHALL be named “Chassis”.

RULE: Each chassis descriptor section SHALL contain one of each tag line type described in Table 2-8.

Table 2-8. Chassis Description File - Chassis Tag Line Descriptions

Tag	Valid Values	Description
PCIBusSegmentList	A comma-separated list of n, where *n* is a decimal integer such that 1 <= *n* <= 255.	This tag enumerates the PCI bus segments in a chassis.
TriggerBusList	A comma-separated list of *n*, where *n* is a decimal integer such that *n* >= 1.	This tag enumerates the trigger buses in a chassis.
StarTriggerList	A comma-separated list of *n*, where *n* is a decimal integer such that *n* >= 1.	This tag enumerates the sets of star trigger in a chassis.
SlotList	A comma-separated list of *n*, where *n* is a decimal integer such that *n* >= 1.	This tag enumerates the slots in a chassis.
Model	A string indicating the model of this chassis.	This tag identifies the chassis model name.
Vendor	A string indicating the vendor of this chassis.	This tag identifies the chassis vendor name.

Chassis Descriptor Example

```
# This example describe a 3-segment, 18-slot PXI chassis
[Chassis]
PCIBusSegmentList = 1,2,3
TriggerBusList = 1,2,3
StarTriggerList = 1
SlotList = 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18
Model = "18-Slot Chassis"
Vendor = "PXISA"
```

RULE: Multiple PCI bus segments SHALL be uniquely numbered in the PCIBusSegmentList tag.

OBSERVATION: PCI bus segments can be numbered in an arbitrary fashion. For example, PCI bus segments can be numbered according to their order of discovery using a depth-first PCI traversal algorithm.

RULE: Mulitple trigger buses SHALL be uniquely numbered in the TriggerBusList tag.

OBSERVATION: Trigger buses can be numbered in an arbitrary fashion. For example, a trigger bus can be sequentially numbered based on the relative order of the slots it contains.

RULE: Multiple sets of star triggers SHALL be uniquely numbered in the StarTriggerList tag.

OBSERVATION: Sets of star triggers can be numbered in an arbitrary fashion. For example, a set of star triggers can be sequentially numbered based on the relative order of the slots the set contains.

RULE: PXI slots SHALL be uniquely numbered according to their corresponding physically-viewable slot numbers.

2.4.3 PCI Bus Segment Descriptor

A PCI bus segment descriptor characterizes a PCI bus segment in a chassis. The most important aspect of a PCI bus segment descriptor is that it describes the mapping from PCI address lines (AD[31:0]) to IDSEL assignments for the segment's slots, bridges, and backplane devices.

RULE: A chassis description file SHALL contain a distinct PCI bus segment descriptor for each physical PCI bus segment in a chassis.

RULE: A PCI bus segment descriptor SHALL be named "PCIBusSegment*N*", where *N* is the PCI bus segment number.

RULE: PCI bus segment numbers SHALL be derived from the PCIBusSegmentBusList tag of the chassis descriptor (see Table 2-8).

OBSERVATION: While each PCI bus segment number will uniquely correspond to a PCI bus number, the PCI bus segment number will not necessarily be equal to the corresponding PCI bus number.

RULE: Each PCI bus segment descriptor SHALL contain one of each of the tag line types described in Table 2-9.

Table 2-9. Chassis Description File – PCI Bus Segment Tag Line Descriptions

Tag	Valid Values	Description
SlotList	A comma-separated list of *n*, where *n* is a decimal integer such that *n* >= 1.	This tag enumerates the slots on a PCI bus segment.
BridgeList	A comma-separated list of *n*, where *n* is a decimal integer such that 1 <= *n* <= 255.	This tag enumerates the PCI-PCI bridges on a PCI bus segment.
IDSELList	*n*, where *n* is a decimal integer such that 1 <= *n* <= 31.	This tag lists the PCI address line numbers (AD[31:0]) used to implement the IDSEL signals for devices on a PCI bus segment.
IDSEL*n,* where *n* is a decimal integer corresponding to a PCI address line (AD[31:0]), for each *n* contained in the IDSELList	A slot descriptor. A bridge descriptor. (Other).	This tag specifies the PCI address line number (AD[31:0]) used to implement the IDSEL signal for a given slot, bridge, or backplane device on a PCI bus segment.

PCI Bus Segment Descriptor Example

```
# This example describes the first PCI bus segment of a
# 3-segment, 18-slot chassis.
[PCIBusSegment1]
SlotList = 1,2,3,4,5,6
BridgeList = 1
IDSELList = 31,30,29,28,27,26
IDSEL31 = Slot2
IDSEL30 = Slot3
IDSEL29 = Slot4
IDSEL28 = Bridge1
IDSEL27 = Slot5
IDSEL26 = Slot6
```

RULE: Slots SHALL be uniquely numbered in the SlotList tag.

OBSERVATION: Slot numbers will correspond to physically-viewable slot numbers for a PCI bus segment. In addition, the SlotList will be a subset of the SlotList specified in the chassis descriptor (see Table 2-8).

RULE: Multiple bridges SHALL be uniquely numbered in the BridgeList tag.

OBSERVATION: Bridges can be numbered in an arbitrary fashion. For example, bridges can be numbered according to their order of discovery using a depth-first PCI traversal algorithm.

PERMISSION: A PCI bus segment descriptor MAY contain an empty list of slots. For example, a chassis might use multiple levels of PCI-PCI bridging to bridge two PXI bus segments. In this case, the first of these PCI bus segments would not contain a list of slots. This type of PCI bus segment will contain a bridge routing, however.

PERMISSION: A PCI bus segment descriptor MAY specify an IDSEL routing to a backplane device other than a slot or a bridge.

2.4.3.1 System Controller Module Slot Considerations

The system controller module slot presents a special challenge in describing PCI bus segments.

OBSERVATION: Because the system controller module does not have its own IDSEL assignment, it is not included in the IDSELList. The slot number of the system controller module is included in the SlotList tag for a chassis' first PCI bus segment descriptor, however, and a Resource Manager can enumerate the system controller module in this manner.

2.4.4 Trigger Bus Descriptor

A trigger bus descriptor describes an individual trigger bus in a PXI chassis. A trigger bus is characterized by a list of slots that reside on the trigger bus.

RULE: A chassis description file SHALL contain a distinct PXI trigger bus descriptor for each physical PXI trigger bus in the chassis.

RULE: A trigger bus descriptor SHALL be named "TriggerBus*N*", where *N* is the trigger bus number.

RULE: Trigger bus numbers SHALL be derived from the TriggerBusList tag of the chassis descriptor (see Table 2-8).

OBSERVATION: While each trigger bus number will uniquely correspond to a set of PXI slots, there is not necessarily a one-to-one correspondence between trigger buses and PCI bus segments.

RULE: Each trigger bus descriptor SHALL contain one of each of the tag line types described in Table 2-10.

Table 2-10. Chassis Description File - Trigger Bus Tag Line Descriptions

Tag	Valid Values	Description
SlotList	A comma-separated list of *n*, where *n* is a decimal integer such that *n* >= 1.	This tag enumerates the slots on a trigger bus.

Trigger Bus Descriptor Example

```
# This example describes the first trigger bus segment of a
# 3-segment, 18-slot chassis.
```

```
[TriggerBus1]
SlotList = 1,2,3,4,5,6
```

2.4.5 Star Trigger Descriptor

A star trigger descriptor describes an individual set of star triggers in a PXI chassis. A star trigger descriptor is characterized by a star trigger controller slot number and a mapping of PXI_STAR lines, as defined in the *PXI Hardware Specification*, to peripheral slot numbers.

RULE: A chassis description file SHALL contain a distinct PXI star trigger descriptor for each physical set of star triggers in the chassis.

RULE: A star trigger descriptor SHALL be named "StarTrigger*N*", where *N* is the number for the set of star triggers.

RULE: Star trigger descriptor numbers SHALL be derived from the StarTriggerList tag of the chassis descriptor (see Table 2-8).

RULE: Each star trigger descriptor SHALL contain one of each of the tag line types described in Table 2-11.

Table 2-11. Chassis Description File - Star Trigger Tag Line Descriptions

Tag	Valid Values	Description
ControllerSlot	A decimal integer *n*, where *n* is a decimal integer such that *n* >= 1.	This tag specifies the star trigger controller slot number for a set of star triggers.
PXI_STAR*n* (where *n* is a decimal integer such that 0 <= *n* <= 12), for each PXI star trigger line routed to a PXI slot.	A comma-separated list of *m*, where *m* is a decimal integer, corresponding to a PXI slot number, such that *m* >= 2.	This tag specifies the PXI_STAR line to slot number mapping for a set of star triggers.

Star Trigger Descriptor Example

```
# This example describes the star trigger bus of a
# 3-segment, 18-slot PXI chassis.
[StarTrigger1]
ControllerSlot = 2
PXI_STAR0 = 3
PXI_STAR1 = 4
PXI_STAR2 = 5
PXI_STAR3 = 6
PXI_STAR4 = 7
PXI_STAR5 = 8
PXI_STAR6 = 9
PXI_STAR7 = 10
PXI_STAR8 = 11
PXI_STAR9 = 12
PXI_STAR10 = 13
PXI_STAR11 = 14
PXI_STAR12 = 15
```

2.4.6 Bridge Descriptor

A bridge descriptor characterizes PCI-PCI bridges linking two PCI bus segments in a multisegment PXI chassis. A bridge contains a pointer to a secondary bus segment (the PCI bus segment subordinate to the bridge).

RULE: A chassis description file SHALL contain a distinct bridge descriptor for each PCI-PCI bridge linking multiple PCI bus segments.

RULE: A bridge descriptor SHALL be named "Bridge*N*", where *N* is the bridge number.

RULE: Bridge numbers SHALL be derived from the BridgeList tag of the PCI bus segment descriptor (see Table 2-9).

RULE: Each bridge descriptor SHALL contain one of each of the tag line types described in Table 2-12.

Table 2-12. Chassis Description File - Bridge Tag Line Descriptions

Tag	Valid Values	Description
SecondaryBusSegment	A PCI bus segment descriptor for the segment subordinate to this bridge.	This tag points to the PCI bus segment subordinate to this bridge.

Bridge Descriptor Example

```
# This example describes a bridge that links segment 1
# to segment 2 in a 3-segment PXI chassis.
[Bridge1]
SecondaryBusSegment = PCIBusSegment2
```

2.4.7 Slot Descriptor

A slot descriptor describes an individual slot in a PXI chassis. A slot descriptor is characterized by the features of the slot it describes, including routing information for the slot's local bus lines.

RULE: A chassis description file SHALL contain a distinct slot descriptor for each physical slot in the chassis.

RULE: A slot descriptor SHALL be named "Slot*N*", where *N* is the physical slot number.

RULE: A slot number SHALL be derived by enumerating IDSEL assignments for the corresponding PCI bus segment descriptor (see Table 2-9).

PERMISSION: A slot number MAY be derived from alternate sources, including a chassis descriptor's SlotList tag (see Table 2-8) or the corresponding PCI bus segment descriptor's SlotList tag (see Table 2-9).

RULE: Each slot descriptor SHALL contain one of each of the non-shaded tag line types described in Table 2-13.

PERMISSION: A chassis description MAY continue to use the shaded fields of Table 2-13.These fields may be removed in a future revision.

Table 2-13. Chassis Description File - Slot Tag Line Descriptions

Tag	Valid Values	Description
LocalBusLeft	A valid slot descriptor. A valid star trigger descriptor. (Other).	This tag indicates how this slot routes its local bus pins to the left.
LocalBusRight	A valid slot descriptor. (Other).	This tag indicates how this slot routes its local bus pins to the right.
ExternalBackplaneInterface	None. (Other).	If this slot routes to an external backplane interface, this tag specifies the name of that interface.

Slot Descriptor Example

```
# This example describes a PXI slot in a 3-segment, 18-slot
# PXI chassis.
[Slot12]
LocalBusLeft = Slot11
LocalBusRight = Slot13
ExternalBackplaneInterface = None
```

OBSERVATION: A PXI slot that does not implement the full set of PXI features, such as a CompactPCI-only slot, will have tag values corresponding to PXI features set to “None”. For example, a CompactPCI-only slot would set its LocalBusLeft and LocalBusRight tags to “None”. In addition, this slot would not be present in the SlotList for a trigger bus, and it would not belong to a set of star triggers.

2.4.8 Chassis Description File Examples

The following are complete examples of chassis description files.

2.4.8.1 Example 8-Slot PXI Chassis

```
# This example describes an 8-slot PXI chassis with
# 1 PCI bus segment, 1 trigger bus, and 1 star trigger.

[Version]
Major = 2
Minor = 1

[Chassis]
Model = "Example 8-Slot Chassis"
Vendor = "PXISA"
PCIBusSegmentList = 1
TriggerBusList = 1
StarTriggerList = 1
SlotList = 1,2,3,4,5,6,7,8

[PCIBusSegment1]
SlotList = 1,2,3,4,5,6,7,8
```

```
BridgeList = None
IDSELList = 31,30,29,28,27,26,25
IDSEL31 = Slot2
IDSEL30 = Slot3
IDSEL29 = Slot4
IDSEL28 = Slot5
IDSEL27 = Slot6
IDSEL26 = Slot7
IDSEL25 = Slot8

[TriggerBus1]
SlotList = 1,2,3,4,5,6,7,8

[StarTrigger1]
ControllerSlot = 2
PXI_STAR0 = 3
PXI_STAR1 = 4
PXI_STAR2 = 5
PXI_STAR3 = 6
PXI_STAR4 = 7
PXI_STAR5 = 8

[Slot1]
LocalBusLeft = None
LocalBusRight = None
ExternalBackplaneInterface = None

[Slot2]
LocalBusLeft = StarTrigger1
LocalBusRight = Slot3
ExternalBackplaneInterface = None

[Slot3]
LocalBusLeft = Slot2
LocalBusRight = Slot4
ExternalBackplaneInterface = None

[Slot4]
LocalBusLeft = Slot3
LocalBusRight = Slot5
ExternalBackplaneInterface = None

[Slot5]
LocalBusLeft = Slot4
LocalBusRight = Slot6
ExternalBackplaneInterface = None

[Slot6]
LocalBusLeft = Slot5
LocalBusRight = Slot7
ExternalBackplaneInterface = None

[Slot7]
LocalBusLeft = Slot6
```

```
LocalBusRight = Slot8
ExternalBackplaneInterface = None

[Slot8]
LocalBusLeft = Slot7
LocalBusRight = None
ExternalBackplaneInterface = None
```

2.4.8.2 Example 18-Slot PXI Chassis

```
# This example describes an 18-slot PXI chassis with
# 3 PCI bus segment, 3 trigger buses, and 1 star
# trigger.

[Version]
Major = 2
Minor = 1

[Chassis]
Model = "Example 18-Slot Chassis"
Vendor = "PXISA"
PCIBusSegmentList = 1,2,3
TriggerBusList = 1,2,3
StarTriggerList = 1
SlotList = 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18

[PCIBusSegment1]
SlotList = 1,2,3,4,5,6
BridgeList = 1
IDSELList = 31,30,29,28,27,26
IDSEL31 = Slot2
IDSEL30 = Slot3
IDSEL29 = Slot4
IDSEL28 = Bridge1
IDSEL27 = Slot5
IDSEL26 = Slot6

[TriggerBus1]
SlotList = 1,2,3,4,5,6

[StarTrigger1]
ControllerSlot = 2
PXI_STAR0 = 3
PXI_STAR1 = 4
PXI_STAR2 = 5
PXI_STAR3 = 6
PXI_STAR4 = 7
PXI_STAR5 = 8
PXI_STAR6 = 9
PXI_STAR7 = 10
PXI_STAR8 = 11
PXI_STAR9 = 12
PXI_STAR10 = 13
PXI_STAR11 = 14
```

```
PXI_STAR12 = 15

[Slot1]
LocalBusLeft = None
LocalBusRight = None
ExternalBackplaneInterface = None

[Slot2]
LocalBusLeft = StarTrigger1
LocalBusRight = Slot3
ExternalBackplaneInterface = None

[Slot3]
LocalBusLeft = Slot2
LocalBusRight = Slot4
ExternalBackplaneInterface = None

[Slot4]
LocalBusLeft = Slot3
LocalBusRight = Slot5
ExternalBackplaneInterface = None

[Slot5]
LocalBusLeft = Slot4
LocalBusRight = Slot6
ExternalBackplaneInterface = None

[Slot6]
LocalBusLeft = Slot5
LocalBusRight = Slot7
ExternalBackplaneInterface = None

[Bridge1]
SecondaryBusSegment = PCIBusSegment2

[PCIBusSegment2]
SlotList = 7,8,9,10,11,12
BridgeList = 2
IDSELList = 31,30,29,28,27,26,25
IDSEL31 = Slot7
IDSEL30 = Slot8
IDSEL29 = Slot9
IDSEL28 = Bridge2
IDSEL27 = Slot10
IDSEL26 = Slot11
IDSEL25 = Slot12

[TriggerBus2]
SlotList = 7,8,9,10,11,12

[Slot7]
LocalBusLeft = Slot6
LocalBusRight = Slot8
ExternalBackplaneInterface = None
```

```
[Slot8]
LocalBusLeft = Slot7
LocalBusRight = Slot9
ExternalBackplaneInterface = None

[Slot9]
LocalBusLeft = Slot8
LocalBusRight = Slot10
ExternalBackplaneInterface = None

[Slot10]
LocalBusLeft = Slot9
LocalBusRight = Slot11
ExternalBackplaneInterface = None

[Slot11]
LocalBusLeft = Slot10
LocalBusRight = Slot12
ExternalBackplaneInterface = None

[Slot12]
LocalBusLeft = Slot11
LocalBusRight = Slot13
ExternalBackplaneInterface = None

[Bridge2]
SecondaryBusSegment = PCIBusSegment3

[PCIBusSegment3]
SlotList = 13,14,15,16,17,18
BridgeList = None
IDSELList = 31,30,29,28,27,26
IDSEL31 = Slot13
IDSEL30 = Slot14
IDSEL29 = Slot15
IDSEL28 = Slot16
IDSEL27 = Slot17
IDSEL26 = Slot18

[TriggerBus3]
SlotList = 13,14,15,16,17,18

[Slot13]
LocalBusLeft = Slot12
LocalBusRight = Slot14
ExternalBackplaneInterface = None

[Slot14]
LocalBusLeft = Slot13
LocalBusRight = Slot15
ExternalBackplaneInterface = None

[Slot15]
```

```
LocalBusLeft = Slot14
LocalBusRight = Slot16
ExternalBackplaneInterface = None

[Slot16]
LocalBusLeft = Slot15
LocalBusRight = Slot17
ExternalBackplaneInterface = None

[Slot17]
LocalBusLeft = Slot16
LocalBusRight = Slot18
ExternalBackplaneInterface = None

[Slot18]
LocalBusLeft = Slot17
LocalBusRight = None
ExternalBackplaneInterface = None
```

3. Software Frameworks and Requirements

This section discusses the software features associated with a PXI system. It gives an overview of the general motivating factors behind the *PXI Software Specification*, along with specific software frameworks.

3.1 Overview

Like other bus architectures, PXI defines standards that allow products from multiple vendors to work together at the bus level. The *PXI Software Specification* goes on to mandate software requirements in addition to these bus level requirements. Other buses that have failed to designate software standards have seen the market fragment into competing standards from multiple vendors.

3.2 Motivation

Low-cost, rugged, reliable computer systems are needed in instrumentation and automation applications. The demands of reducing product cost and time to market, while increasing reliability, are severely straining custom-built systems. Hardware vendors have implemented many modular, multivendor solutions to tackle these problems. However, the majority of costs associated with any system development are more likely related to the software development time, rather than the hardware development time.

PXI, unlike other standards, defines *software frameworks* as part of its specification. These software frameworks ensure that a user has a complete, multivendor system solution from the start.

The frameworks that have been chosen for this specification reflect the dominance of these operating systems on current desktop PCs. As other operating systems become widely accepted and offer the same degree of software leverage as the current frameworks, they may be added to the supported PXI frameworks. Each framework is required to support the VISA software standard. VISA provides an industry-standard mechanism for locating and controlling PXI modules.

The currently supported frameworks are the 32-bit Windows (Windows 2000/NT/XP/9*x*) frameworks. Many organizations have aligned their entire offices around Microsoft operating systems, because of the reduction in training and support costs that come from a common, familiar computing environment across the office. These same benefits extend to the factory floor or test department.

Thousands of 32-bit applications are running today on the Windows platforms, ranging from technical and engineering applications to complete manufacturing and financial solutions. All of these tools can be leveraged to improve instrumentation systems.

The ever-increasing interconnection between computers that design, produce, and test goods is driving the requirement that these systems easily interact with each other. The use of a common operating system across these computers means that a richer set of tools is available for exchanging and sharing data. All of the work being done to interconnect the desktop—COM, ODBC, .NET, and so on—is now available to interconnect machines on the factory floor to each other and to the rest of the corporation.

3.3 Framework Definition

The software frameworks define PXI system software requirements for both system controller modules and PXI peripheral modules. System controller modules and PXI peripheral modules have to meet certain requirements for operating system and tool support in order to be considered compliant with a given PXI software framework.

RULE: PXI system controller modules and PXI peripheral modules SHALL support the 32-bit Windows framework (2000/NT/XP/9*x*).

3.4 32-bit Windows System Framework

3.4.1 Introduction

This section defines the specific requirements for the 32-bit Windows system framework. It defines all of the unique components that must exist to support this framework. It also describes the optional recommended components.

3.4.2 Overview of the Framework

The 32-bit Windows system framework defines a system based on the popular PC architecture, and is based on the Windows operating system from Microsoft.

3.4.3 Controller Requirements

This section defines the system requirements for the 32-bit Windows framework for the system controller module.

RULE: The system controller module SHALL be based on the 80x86 architecture.

OBSERVATION: Processors other than the 80x86 that are supported under 32-bit Windows may be added in additional frameworks.

RECOMMENDATION: All system controller modules SHOULD be supplied with a VISA implementation that supports the PXI bus.

RULE: The above RECOMMENDATION SHALL be made a RULE three (3) months after a VISA for PXI specification is published. At that time, all system controller modules SHALL be required to provide a VISA implementation that supports the PXI bus.

OBSERVATION: The target date for the transition to this requirement is September 30, 2003.

3.4.4 PXI Peripheral Module Requirements

Hardware vendors for other industrial buses that do not have software standards often do not provide any software drivers for their modules. The customer is often given only a manual describing how to write software to control the module. The cost to the customer, in terms of engineering effort to support these modules, is huge. PXI removes this burden by requiring that manufacturers, rather than customers, develop this software.

RULE: Peripheral modules SHALL provide software for installing, configuring, and controlling the modules under Windows.

RECOMMENDATION: PXI peripheral modules that are instrumentation class modules SHOULD provide a user-level interface that is supported under the development environments specified in Table 3-1.

Table 3-1. Development Environments Supported by PXI Modules Under Windows

Product	Company	Revision
LabVIEW	National Instruments	4.0 or higher
LabWindows/CVI	National Instruments	4.0 or higher

Table 3-1. Development Environments Supported by PXI Modules Under Windows (Continued)

Product	Company	Revision
ATEasy	Geotest-Marvin Test Systems, Inc.	4.0 or higher
Visual Basic	Microsoft	5.0 or higher
Visual C/C++	Microsoft	5.0 or higher

OBSERVATION: Other system tools MAY be supported in addition to these tools.

3.5 Support for Existing Instrumentation Standards

The challenge for developing PXI instrumentation systems is to provide a platform that extends the capabilities of new test systems, while supporting existing instrumentation standards. The VXI*plug&play* System Alliance provides an industry-standard mechanism for communicating with GPIB, VXI, and serial instrumentation through a series of specifications. These specifications define communications standards (*VISA*), instrument programming interfaces (*instrument drivers*), and interactive user interfaces (*soft front panels*). These specifications also define frameworks for the various Windows operating systems.

RULE: A PXI module that controls a VISA supported interface other than the PXI bus SHALL provide the VISA software as a mechanism for communicating with that interface.

The use of the VISA standard in PXI preserves the user's investment in existing instrumentation software. VISA provides the link from PXI to a VXI chassis and instruments and standalone GPIB and serial instruments.

RECOMMENDATION: Instrumentation class PXI peripheral modules SHOULD provide instrument drivers and soft front panels that are consistent with the VXI*plug&play* instrument driver specifications (VPP-3.*x* and VPP-7).

The goal of supporting VXI*plug&play* instrument drivers and soft front panels is to provide a familiar development environment to test and measurement customers. Test and measurement customers have come to expect VXI and GPIB instruments to have soft front panels and instrument drivers. VXI*plug&play* support for native PXI instruments provides a seamless software path between VXI and PXI based systems.

RECOMMENDATION: All system controller modules SHOULD be supplied with a VISA implementation that supports the PXI bus.

RULE: The above RECOMMENDATION SHALL be made a RULE three (3) months after a VISA for PXI specification is published. At that time, all system controller modules SHALL be required to provide a VISA implementation that supports the PXI bus.

OBSERVATION: The target date for the transition to this requirement is September 30, 2003.