PXI™-4

PXI Module Description File Specification

PCI eXtensions for Instrumentation

An Implementation of *CompactPCI™*

Revision 1.1
October 18, 2012

PXI Systems Alliance

IMPORTANT INFORMATION

Copyright

NOTICE

The *PXI Module Description File Description* is authored and copyrighted by the PXI Systems Alliance. The intent of the PXI Systems Alliance is for the *PXI Module Description File 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 trademark 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 Module Description File Specification Revision History

This section is an overview of the revision history of the *PXI Module Description File Specification*.

Revision 1.0, September 25, 2003

This is the first public revision of the *PXI Module Description File Specification*.

Revision 1.1, October 18, 2012

Adds support for systems with multiple PCI root buses.

Mandates a specific installation path for module description files.

Updates examples to comply with `.ini` file rules.

Contents

1. Introduction

This specification is intended to describe PXI module description files. A PXI module description file lists key attributes of a PXI Module. This section lists terminology and documents that are relevant to the PXI Module Description File Specification.

1.1 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.

1.2 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-1: PXI Hardware Specification*
- *PXI-2: PXI Software Specification*
- *VPP-4.3: The VISA Library Specification*

2. Module Description Files

Module description files provide a mechanism for informing the PXI Resource Manager about key attributes of a PXI module. The module description file accomplishes this with two primary features:

- It specifies how software should identify this device and associate it with the VISA implementation.
- It specifies how combination modules with multiple PCI devices and/or multiple PCI functions should be handled by the PXI Resource Manager. (Refer to section 2.7.4 for information about combination modules.)

A module vendor provides a module description file to enable a user to easily configure the product into their PXI system.

OBSERVATION: A module description file is not required for software to correctly identify and interact with every PXI module. Modules with their own drivers (i.e., modules that do not use VISA to communicate with the device) do not require the module description file's ability to associate a device with VISA. Modules which are implemented as a single PCI device/function do not require the module description file's ability to identify combination modules. The result is that many modules will not need either feature of the module description file.

RECOMMENDATION: If the driver software for a module requires VISA, the module vendor SHOULD provide a module description file with VISA registration information for the module.

RECOMMENDATION: If a module consists of multiple PCI devices or multiple PCI functions or both, the module vendor SHOULD provide a module description file for the module.

RECOMMENDATION: Implementations of PXI Resource Managers SHOULD use module description files to detect the described modules in the PXI system. If described modules are detected, the Resource Manager SHOULD incorporate the information found in the module description in the `pxisys.ini` file (refer to the *PXI Software Specification*, section 2.3, *System Description Files*) if appropriate. For example, if the Resource Manager detects a multifunction or multidevice module based on a module description file, it SHOULD create the additional tags and sections in the pxisys.ini file needed to locate all the module's functions and devices in their appropriate slot. Refer to section 2.7, *Module Description File Examples*, for more details. Section 2.7.5, *Incorporating Module Information into PXI System Information*, illustrates how to merge data from a module description file into a system description file.

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

RULE: A module description file name SHALL contain the name of the module vendor to guarantee uniqueness versus module description files from other vendors.

RULE: To maximize backward compatibility, a Resource Manager SHALL be capable of reading module description files with any filename ending with *.ini*.

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

Key: HKEY_LOCAL_MACHINE\SOFTWARE\PXISA\CurrentVersion

Value: ModuleDescriptionFilePath

RULE: The ModuleDescriptionFilePath SHALL be a string value that specifies the complete path of a directory that holds module description files.

RULE: Installation software for a module description file SHALL NOT delete or modify the ModuleDescriptionFilePath registry key if it already exists.

RULE: When creating the ModuleDescriptionFilePath on a system where it did not previously exist, an installer SHALL set the value to the directory `%ALLUSERSAPPDATA%\PXISA\Descriptions\Modules\`, where `%ALLUSERSAPPDATA%` is the user-independent application data folder.

OBSERVATION: Prior versions of this specification did not dictate a specific folder for the module description files, but allowed installers to install a registry key to point to an arbitrary directory. While the above rule was added to simplify installation and removal of software components, the registry key mechanism is maintained for backward compatibility.

OBSERVATION: Using the ModuleDescriptionFilePath registry value, module description file installers can copy their module description files to a standard location. In addition, a PXI Resource Manager can use this location to identify the types of modules in the PXI system.

RULE: The module description file SHALL follow the standard text file format for configuration settings used for PXI hardware descriptions. This format defines sections containing key-value pairs (tags), which can be used as descriptors for elements of a module. Refer to the *PXI Software Specification*, section 2.2, *Common File Requirements*, for information about `.ini` files.

RULE: The values of taglines described by this specification as hexadecimal numbers SHALL be expressed with the radix-16 prefix ‘0x’.

OBSERVATION: The above rule is intended to be an exception to a rule in *PXI-2: PXI Software Specification* which states that all numeric constants must be radix-10.

RULE: The module description file SHALL include a version descriptor. Refer to the *PXI Software Specification*, section 2.2.1, *Version Descriptor*, for details about the format of the version descriptor.

2.1 Module Description Definitions

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

- **Module**—A module descriptor corresponds to a physical PXI module. PXI modules typically contain a single PCI device located at PCI function 0 within the module, but can contain PCI-to-PCI bridges to other devices on the module or multiple functions on a single device or both.
  - **Function**—A function descriptor corresponds to a single PCI function on a PCI device within the module. This function can be a PCI-to-PCI bridge, in which case the function provides the connection to the PCI devices on the other side of the bridge. This function can be registered with VISA.
    - **VISA Registration**—A VISA registration descriptor provides the information necessary to register a function of the module with VISA.
  - **Device**—A device descriptor corresponds to a single PCI device on the module. This device must have at least one function (function 0) and may have more than one.

OBSERVATION: Most PXI modules (and most PCI devices of any kind) are implemented with only one function, function 0. For this reason, a PCI bus/device/function address is often referred to as a PCI device. This document attempts to be specific when referring to a function versus a device.

2.2 Module Descriptor

A module descriptor provides a high-level description of a PXI module, including identifying data about the module such as the name and vendor of the module, and information about what PCI functions exist in the module.

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

RULE: The module descriptor SHALL be named "Module."

RULE: Each module descriptor SHALL include the appropriate tag lines described below:

Tag	Valid Values	Description
ModuleName	A text string.	This tag describes the module.
ModuleVendor	A text string.	This tag gives the module vendor.
FunctionList OR The tags for a function descriptor (defined below, section 2.3)	A list of comma-separated numbers.	The FunctionList tag lists the PCI function numbers rooted at this module's bus/device address, and is used to determine which function descriptors are present in the module specification file. If omitted, FunctionList of "0" is implied and the tags from the function descriptor must be present here instead.

Module Descriptor Example

```
# This example describes a module containing a multifunction device
[Module]
ModuleName = "Multifunction Module"
ModuleVendor = "PXISA"
FunctionList = "0,1"
```

RULE: A FunctionList SHALL include all of the functions of the module (that is, of the PCI device at the root of the module), including PCI-to-PCI bridge functions.

OBSERVATION: The module descriptor is a device descriptor (defined below in section 2.5) with the addition of the ModuleName and ModuleVendor tags.

2.3 Function Descriptor

A function descriptor characterizes a single PCI function on the module. All PXI modules have at least one PCI function. The function descriptor specifies whether the function is a normal PCI device or a PCI-to-PCI bridge, gives the PCI vendor and device IDs for the function (including PCI subsystem IDs if appropriate), and may include information about how to register the function with VISA.

RULE: Except as noted below for cases where only function 0 exists, a module description file SHALL contain a distinct function descriptor for each function of the module.

RULE: A function descriptor for a function located at the root of the module and enumerated in the FunctionList of a module descriptor SHALL be named "Function*F*", where *F* is the function number.

RULE: A function descriptor for a function located on a bridged device on the module and enumerated in the FunctionList of a device descriptor SHALL be named "*deviceName*Function*F*" where *F* is the function number and *deviceName* is the name of the device descriptor.

RULE: Function numbers in function descriptor names SHALL be those listed in the FunctionList tag of a module or device descriptor.

PERMISSION: If only function 0 exists in a module or device, the FunctionList tag MAY be omitted from the module or device descriptor. If this is done, the function descriptor tags take the place of the FunctionList tag in the module or device descriptor.

RULE: Each function descriptor SHALL contain the following tag line types:

Tag	Valid Values	Description
Type (OPTIONAL)	The exact string “Device”. The exact string “InternalBridge”.	This tag defines which other tags are expected. “Device” is used when the specified function acts as a normal PCI device. “InternalBridge” is used when the specified function is a bridge to additional devices on the same module. If Type is omitted, the function is assumed to be of type “Device”.
ModelCode	A 16-bit hexadecimal number.	This tag represents the PCI device ID of this function.
ManufCode	A 16-bit hexadecimal number.	This tag represents the PCI vendor ID of this function.
SubsystemModelCode (OPTIONAL)	A 16-bit hexadecimal number.	This tag represents the PCI subsystem ID of this function, if appropriate. This tag is REQUIRED if SubsystemManufCode is present. If specified, it implies that VISA should use the subsystem to identify the device.
SubsystemManufCode (OPTIONAL)	A 16-bit hexadecimal number.	This tag represents the PCI subsystem vendor ID of this function, if appropriate. This tag is REQUIRED if SubsystemModelCode is present.

Tag	Valid Values	Description
VISARegistration (OPTIONAL if Type=Device, not applicable otherwise)	The exact string “None”. The exact string “Simple”. The exact string name of a VISA device registration information section.	This tag describes how to register this function with VISA. If not present, the value “None” is assumed, which means this function will not be registered with VISA (for example, because it already has a separate driver). If the value “Simple” is used, the device is registered using the default settings for the tags described in the VISA registration descriptor. Otherwise, it is registered using the information provided in the specified section.
DeviceList (if Type=InternalBridge, not applicable otherwise)	A list of comma-separated numbers.	This tag lists the PCI device numbers of devices found on this module behind this bridge. The device numbers in this list are fixed for a given module since the IDSEL lines between the bridge and its devices are fixed.

Module Descriptor Example

```
# This example describes a function which is a normal PCI device at function 0 and
# which should be registered with VISA
[Function0]
Type = "Device"
ModelCode = 0x1234
ManufCode = 0xABCD
VISARegistration = "Simple"
```

RULE: For functions of type InternalBridge, the DeviceList tag SHALL list all devices bridged by this function.

PERMISSION: The VISARegistration tag MAY be other strings besides those listed in order to allow vendor-specific extensions.

RULE: If a vendor’s PXI Resource Manager, VISA, or other tool does not recognize the VISARegistration tag and cannot find a corresponding VISA registration descriptor in the module description file, it SHALL treat the tag as if it had the value “None”.

RULE: If a VISARegistration tag specifies a name that conflicts with a vendor-specific extension, tools which use that vendor-specific extension SHALL perform the specified behavior of the VISARegistration tag and not the vendor-specific behavior.

OBSERVATION: A module containing multiple PCI-to-PCI bridges would be exceedingly rare. However, such modules are fully supported because each function descriptor which represents a PCI-to-PCI bridge can contain a list of device descriptors which can contain further function descriptors representing more bridges.

2.4 VISA Registration Descriptor

The VISA registration descriptor provides information needed to register the device with VISA. If a function descriptor specifies a VISARegistration tag for that function, it can provide as that tag's value a string specifying a required separate descriptor of VISA registration information.

RULE: A module description file SHALL contain a unique VISA registration descriptor for each function descriptor that includes a custom VISARegistration tag.

RULE: A VISA registration descriptor SHALL be named according to the string given in the function descriptor's VISARegistration tag. For example, if a function descriptor lists "VISARegistration=*MyFunctionRegistration*", the module description file SHALL include a VISA registration descriptor named "*MyFunctionRegistration*".

RULE: Each VISA registration descriptor SHALL contain the following tags:

Tag	Valid Values	Description
NumDetectSequences (OPTIONAL)	A non-negative integer.	This tag indicates whether the given function of the module can generate interrupts that should be handled by VISA, and if so, how many different detection mechanisms are required to determine if the function is asserting an interrupt condition. If this tag is omitted it is assumed to be 0, indicating that the function does not generate interrupts that are handled by VISA.
InterruptDetect*X* (if NumDetectSequences > 0)	A quoted string in interrupt detect/quiesce format (see section 2.4.1).	This tag is required for each *X* in the range [0…NumDetectSequences-1]. Each value is a quoted text string describing a series of register reads, writes, and compares that will allow VISA to identify whether an interrupt condition is caused by the given function of the module. At least one sequence must be true if this function is interrupting.
InterruptQuiesce (OPTIONAL if NumDetectSequences > 0)	A quoted string in interrupt detect/quiesce format.	This tag describes a series of register reads, writes, and compares that will cause the given PCI function to stop asserting the interrupt condition detected by an InterruptDetect*X* operation. If this tag is omitted and NumDetectSequences > 0, an empty string is used.

Tag	Valid Values	Description
ManufName (OPTIONAL)	A text string.	This tag gives the name of the vendor of this function, which will be used to identify the function to VISA (that is, VI_ATTR_MANF_NAME) where appropriate. If this tag is omitted, the ModuleVendor value from the module description section (see section 2.2 above) is used instead.
ModelName (OPTIONAL)	A text string.	This tag gives the name of this function, which will be used to identify the function to VISA (that is, VI_ATTR_MODEL_NAME) where appropriate. If this tag is omitted, the ModuleName value from the module description section (see section 2.2 above) is used instead.

VISA Registration Descriptor Example

```
# This example registers a VISA device that can cause interrupts
[MyFunctionRegistration]
NumDetectSequences = 1
InterruptDetect0 = "C8 BAR0 0x1000 0x1 0x1;"
InterruptQuiesce = "W8 BAR0 0x1000 0x0;"
ModelName = "Interrupting Function"
ManufName = "PXISA"
```

PERMISSION: The InterruptQuiesce operation MAY be an empty string as long as the InterruptDetect*X* operation(s) inhibit(s) the interrupt condition as a side effect of detecting it (as for a Release On Register Access, or RORA, implementation).

RULE: The detect operation is always required for a function that may cause interrupts that will be handled by VISA, and SHALL NOT be empty if present.

OBSERVATION: The VISA device registration also requires the use of the PCI device and vendor IDs and possibly the PCI subsystem ID and subsystem vendor ID. These values are found in the function descriptor (section 2.3) which is the parent of the VISA registration descriptor.

OBSERVATION: All tags in the VISA registration descriptor are optional.

RULE: If a VISA registration descriptor is present, it SHALL contain at least one of the specified tags. If all of the default VISARegistration descriptor tag values are acceptable, the function descriptor SHALL use “Simple” as the VISARegistration tag value, and not provide a VISA registration descriptor.

2.4.1 Interrupt Detect/Quiesce String Format

The InterruptDetect*X* and InterruptQuiesce tags have values that specify PCI register reads, writes, and compares that allow VISA to determine if a particular interrupt condition is caused by this function and if so, to correctly cause the function to stop asserting the interrupt condition.

The interrupt detect/quiesce string format is a compact, human and program-readable list of simple PCI operations. Each sequence string is made up of one or more operations, which are defined as follows:

Operation Syntax	Meaning
W*width space offset value*	Perform a register write of *width* bits (8, 16, or 32) into PCI address space *space* (CFG, BAR*n*) at offset *offset* with the value *value*.
R*width space offset*	Perform a register read of *width* bits (8, 16, or 32) from PCI address space *space* (CFG, BAR*n*) at offset *offset*.
C*width space offset mask value*	Perform a register read of *width* bits (8, 16, or 32) from PCI address space *space* (CFG, BAR*n*) at offset *offset*, mask the read value with *mask*, and compare the result to *value*.

For the purpose of the InterruptDetect*X* tag, the entire sequence is considered “true” if and only if all compare steps in the sequence are true.

Interrupt Detect Example

An example of an interrupt detect sequence string could be `"W32 BAR0 0x00001830 0x00000000;C8 BAR0 0x00001002 0x01 0x01;"`. This would write offset 0x1830 in PCI base address space 0 with the 32-bit value zero, then read the 8-bit register at offset 0x1002 and check if the bit masked by 0x1 is set. If so, the sequence is true.

Interrupt Quiesce Example

An example of an interrupt quiesce sequence string could be `"W8 BAR0 0x00001002 0x02;"`. This would simply write the 8-bit register at offset 0x1002 in base address space 0 with the value 0x02, which must stop the associated device from interrupting. Note that the interrupt condition need not be completely handled at this point—a VISA callback can be installed to do further handling once the interrupt is squelched.

RULE: Operations in a sequence SHALL be separated by semicolons and the list SHALL be terminated by a semicolon. Tokens within the operations SHALL be separated by whitespace.

PERMISSION: Operations in a sequence MAY be separated by whitespace in addition to the semicolon.

2.5 Device Descriptor

A device descriptor describes an individual PCI device on the module. The attribute of a device that is of interest is the list of functions found at that device.

RULE: A module description file SHALL include a distinct device descriptor for each device listed in an InternalBridge function descriptor’s DeviceList.

RULE: A device descriptor SHALL be named "*functionName*Device*D*", where *D* is the device number of the device and the DeviceList tag in a function descriptor, and *functionName* is the name of the function descriptor.

PERMISSION: If there is no ambiguity about which function's DeviceList is being defined—for example, because there is only one InternalBridge function and therefore only one DeviceList in the module description file—the device descriptor name MAY omit the *functionName* portion of the device descriptor name. The name must include this qualifier if there would otherwise be multiple device descriptors named Device*D*, however.

RULE: Each device descriptor SHALL contain the following tags:

Tag	Valid Values	Description
FunctionList OR The tags for a function descriptor (defined above, section 2.3)	A list of comma-separated numbers.	The FunctionList tag lists the PCI function numbers rooted at this module's bus/device address, and is used to determine which function descriptors are present in the module specification file. If omitted, FunctionList of "0" is implied and the tags from the function descriptor must be present here instead.

Device Descriptor Example

```
# This example describes a multifunction PCI device
[Function0Device4]
FunctionList = "0,1"
```

RULE: A FunctionList SHALL include all of the functions of the device, including PCI-to-PCI bridge functions.

OBSERVATION: A device descriptor's tags are a subset of the high-level module descriptor's tags, described above in section 2.2.

2.6 Backwards Compatibility

The module description file specification is intended to extend, not obsolete, previously specified PXI hardware description files. Refer to the *PXI Software Specification*, section 2.2.2, *Backwards Compatibility with Previous PXI Specifications*, for information about the backwards-compatibility goals of this specification.

2.7 Module Description File Examples

PERMISSION: A vendor MAY place descriptors or tags in a module description file other than those described in this section.

OBSERVATION: The above permission may be useful to store supplemental information about a module that is useful for advanced vendor-specific functionality.

RECOMMENDATION: Any vendor-specific descriptors or tags in a module description file SHOULD be named such that they are unlikely to collide with tags or descriptors added in a future version of any PXISA specification.

OBSERVATION: The above recommendation can be accomplished by incorporating the vendor name into the descriptor or tag name.

2.7.1 Registering a Single-Function Module with VISA

The simplest module description file satisfies the need for an easy way to register a PXI module with VISA. For a module which does not generate interrupts, the vendor could supply a very short file such as the following example.

2.7.1.1 Example Single-Function Module Description

```
[Module]
ModuleName = "Basic Module"
ModuleVendor = "PXISA"
; note that function 0 of type Device is implied
ModelCode = 0xABCD
ManufCode = 0x1234
VISARegistration = "Simple"
```

In this case, the module contains only one device, with no internal bridges, and only one function on that device; therefore, this module requires no special handling in the PXI Resource Manager or the `pxisys.ini` system information file. Since no FunctionList is specified in the module descriptor, function 0 is assumed, and the required tags from the function descriptor are included directly in the module descriptor. These tags specify the vendor and model code, which are then used as the default values in the VISA device registration. Because the default values for the VISA registration are acceptable, the keyword Simple is used for that value instead of creating a separate VISA registration descriptor. VISA uses this registration information to create a Windows setup information (`.inf`) file like the following example.

2.7.1.2 .inf File Example

```
[Version]
Signature=$WINDOWS NT$
Class=visaPxiDevice
ClassGUID={E1590550-9B9C-11d3-A250-0040055732CC}
ClassName="visaPxiDevice"
Provider=%Vendor0%

;=================================================================================

[ClassInstall32]
AddReg=AddClassToRegistry

[ClassInstall]
AddReg=AddClassToRegistry

;=================================================================================

[ControlFlags]
ExcludeFromSelect=PCI\VEN_1234&DEV_ABCD
ExcludeFromSelect=PCI\VEN_1234&DEV_ABCD&SUBSYS_00000000&REV_00

;=================================================================================
```

```
[Manufacturer]
%Vendor1%=PCIList

[PCIList]
%BasicModule0.DeviceDesc%=BasicModule0.Cfg,PCI\VEN_1234&DEV_ABCD
%BasicModule0.DeviceDescN%=BasicModule0.Cfg,PCI\VEN_1234&DEV_ABCD&SUBSYS_0000000
0&REV_00

;=============================================================================

[BasicModule0.Cfg]
AddReg=BasicModule0.AddReg

[BasicModule0.Cfg.Services]
AddService=sample, 0x00000002, sample_Service_Inst

[BasicModule0.AddReg]
HKR,,BridgeType,1,01,00,00,00
;=============================================================================

[AddClassToRegistry]
HKR,,,0,%DeviceClassString%

[sample_Service_Inst]
DisplayName   = %sample.SvcDesc%
ServiceType   = 1         ;SERVICE_KERNEL_DRIVER
StartType     = 0         ;SERVICE_BOOT_START
ErrorControl  = 1         ;SERVICE_ERROR_NORMAL
ServiceBinary = %12%\sample.sys
;=============================================================================

[Strings]
Vendor0="PXISA"
Vendor1="PXISA"
BasicModule0.DeviceDesc="Basic Module"
BasicModule0.DeviceDescN="Basic Module"
DeviceClassString="Sample PXI Devices"
sample.SvcDesc="Sample Driver"
```

2.7.2 Registering a Module which Generates Interrupts

The module description file specification section 2.4 describes how to register a function with VISA when that function can cause interrupts that must be handled by VISA. In this case the `module.ini` file requires a VISA registration descriptor. Again, the function description for this single-function module is collapsed into the module descriptor, but this time the VISARegistration value specifies the separate descriptor needed to elaborate on the registration information.

2.7.2.1 Example Module Description with Interrupt Handling

```
[Module]
ModuleName = "Basic Module"
ModuleVendor = "PXISA"
ModelCode = 0xABCD
ManufCode = 0x1234
```

```
VISARegistration = "MyModuleRegistration"

[MyModuleRegistration]
NumDetectSequences = 1
InterruptDetect0 = "C8 BAR0 0x00001002 0x01 0x01;"
InterruptQuiesce = "W8 BAR0 0x00001002 0x02;"
```

2.7.3 Identifying a Multifunction Module

The module description file allows vendors to associate multiple PCI functions of a single device so that they can be properly identified by the PXI Resource Manager. Since there are multiple functions, separate and explicit function descriptors are required for this module.

2.7.3.1 Example Module Description with Multiple Functions

```
[Module]
ModuleName = "Sample Multifunction Module"
ModuleVendor = "PXISA"
FunctionList = "0,1"

[Function0]
; note that Type = Device is implied if omitted
Type = "Device"
ModelCode = 0xABCD
ManufCode = 0x1234
SubsystemModelCode = 0x0001
SubsystemManufCode = 0x1234
VISARegistration = "FirstFunction"

[Function1]
ModelCode = 0xABCE
ManufCode = 0x1234
SubsystemModelCode = 0x0002
SubsystemManufCode = 0x1234
VISARegistration = "SecondFunction"

[FirstFunction]
ManufName = "PXISA"
ModelName = "Multifunction Module Function 0"

[SecondFunction]
ManufName = "PXISA"
ModelName = "Multifunction Module Function 1"
```

2.7.4 Identifying a Combination Module with Multiple Devices

The most complex kinds of PXI modules from a PCI standpoint are modules with multiple separate PCI devices connected to a bridge device. These are referred to as *combination modules*.

2.7.4.1 Example Module Description with Multiple Devices

```
[Module]
ModuleName = "Sample Bridged Module"
VendorName = "PXISA"
```

```
Type = "InternalBridge"
DeviceList = "4,5"

[Device4]
ModelCode = 0xABCF
ManufCode = 0x1234
VISARegistration = "None"

[Device5]
ModelCode = 0xABD0
ManufCode = 0x1234
VISARegistration = "None"
```

This example also shows how FunctionList tags and function description sections are unnecessary when the only function in a FunctionList would be function 0. This applies to both the module descriptor and the device descriptors. Because there is no ambiguity about which function each device descriptor refers to, the function is not specified in the device descriptor name. For comparison, the above example is exactly equivalent to the following example, which lists all the implied tags and descriptors explicitly:

2.7.4.2 Expanded Example Module Description with Multiple Devices

This listing is provided to clarify the effect of the implied/default tags and descriptors.

```
[Module]
ModuleName = "Sample Bridged Module"
VendorName = "PXISA"
FunctionList = "0"

[Function0]
Type = "InternalBridge"
DeviceList = "4,5"

[Function0Device4]
FunctionList = "0"

[Function0Device4Function0]
Type = "Device"
ModelCode = 0xABCF
ManufCode = 0x1234
VISARegistration = "None"

[Function0Device5]
FunctionList = "0"

[Function0Device5Function0]
Type = "Device"
ModelCode = 0xABD0
ManufCode = 0x1234
VISARegistration = "None"
```

2.7.5 Incorporating Module Information into PXI System Information

The PXI Resource Manager uses the `module.ini` to provide extended information in the `pxisys.ini` file about the combination module. The presence of the extended information sections is indicated by the inclusion of the FunctionList tag in the slot description section. If FunctionList is present, the PCISlotPath,

PCISlotPathRootBus, PCIBusNumber, and PCIDeviceNumber fields of the slot section *apply to the module's root device's PCI function 0*. Tools parsing the `pxisys.ini` file can find those fields for any device/function (including the root function 0) in the function-specific sections indicated by the FunctionList. In the case of the multidevice module illustrated above, the resulting `pxisys.ini` file would contain extended information for the appropriate slot (slot 5 shown in this example) as shown in example 2.7.5.1.
Note that the `pxisys.ini` representation is a verbose version that does not use the "shortcuts" (such as the implied function 0) since it is intended to be parsed programmatically, whereas the `module.ini` file itself is intended for both programmatic parsing and human reading.

For more information about the PCISlotPath and PCISlotPathRootBus tags, refer to *PXI-2: PXI Software Specification*.

OBSERVATION: All functions of a given device will always be in the same physical PXI slot.

2.7.5.1 Example pxisys.ini with Module Information

```
[Chassis1Slot5]
PCISlotPathRootBus = 0
PCISlotPath = "60,88"
PCIBusNumber = 2
PCIDeviceNumber = 12
LocalBusLeft = "Slot4"
LocalBusRight = "Slot6"
ExternalBackplaneInterface = "None"
DescriptionFile = "PXISAModuleDescFile.ini"
FunctionList = "0"

[Chassis1Slot5Function0]
PCISlotPath = "60,88"
PCIBusNumber = 2
PCIDeviceNumber = 12
Type = "InternalBridge"
DeviceList = "4,5"

[Chassis1Slot5Function0Device4]
FunctionList = "0"

[Chassis1Slot5Function0Device4Function0]
Type = "Device"
PCISlotPath = "20,60,88"
PCIBusNumber = 3
PCIDeviceNumber = 4

[Chassis1Slot5Function0Device5]
FunctionList = "0"

[Chassis1Slot5Function0Device5Function0]
Type = "Device"
PCISlotPath = "18,60,88"
PCIBusNumber = 3
PCIDeviceNumber = 5
```