Model CP387-AA11

256 Channel, Digital I/O

User's Manual

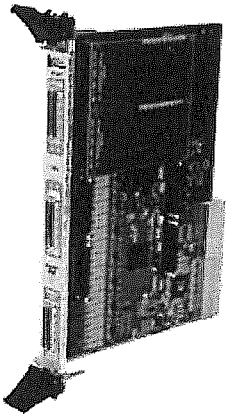
KineticSystems

CP387

Single-width, 6U CompactPCI module

The CP387 is a single-width, 6U, CompactPCI module with up to 256 digital input/output channels.

Four mezzanine card sites can be populated with other forms of digital I/O including isolated input, isolated output, relay output, AC switch output or differential I/O.



KineticSystems' CP387 is a single-width, 6U, CompactPCI module with up to 256 digital input/output channels.

FEATURES

- 256 channel Digital Input/Output Channels
- 128 Base card channels
- 4 mezzanine card sites for added I/O capability including:
  - TTL I/O
  - Differential I/O
  - Isolated Input
  - Isolated Output
  - Form "A" Relay Output
  - Form "C" Relay Output
  - AC switch Output
- Change Of State Recognition
- Pattern Recognition
- Input and Output Strobes
- Programmable contact-bounce suppression on inputs

TYPICAL APPLICATIONS

Automotive test cells

Industrial monitoring and control

Automatic Test Equipment (ATE)

Monitoring and driving TTL-level signals

Driving relays, solenoids or lamps

GENERAL DESCRIPTION

The CP387 is a single-width, 6U, CompactPCI module with up to 256 digital input/output channels. The CP387 base board supports 128 channels of TTL I/O. Four mezzanine card sites can be populated with other forms of digital I/O including isolated input, isolated output, relay output, AC switch output or differential I/O. The mezzanine card concept allows multiple digital I/O types to be configured within a single module to match the application requirements.

Pattern Recognition and Change Of State detection are included. Both operations can be used on the base card as well as span to the mezzanine channels. Input and output strobes are provided for connection to external sources.

The digital inputs and outputs are available on 3 double stacked 68 position high density connectors.

BASIC CIRCUIT OPERATION

The CP387 base-board provides 128 TTL level digital I/O channels. Expansion of up to 128 digital channels is provided by 4 mezzanine card sites. These sites can be populated with an assortment of I/O options to extend the capability of the CP387.

Data transfers to and from the CP387 are under programmed control and support 16 and 32 bit data words. Transfers to and from the base-board channels are executed directly. Transfers to the mezzanine card sites are translated by the base-board logic. The mezzanine card sites use an Industry Pack (IP) form factor and require the translation of PCI bus cycles to IP bus cycles.

Accessing digital output channels is accomplished by writing the selected data values to a register location. Reading digital input channels is also accomplished by simple register read operations.

The CP387 contains flash memory that can be used to restore basic digital input and output configuration parameters on power up. These parameters include the connection of either pull-up or pull-down resistors, the direction of digital I/O channels, and the initial power-up values of digital output channels.

Pattern recognition and change-of-state operations are supported by the CP387 base-board as well as by mezzanine plug on cards. The base-board performs these two operations by latching the digital input data from all sources and then performing a compare for the selected operation. If enabled, an interrupt can be generated at the conclusion of a pattern recognition or change-of-state operation.

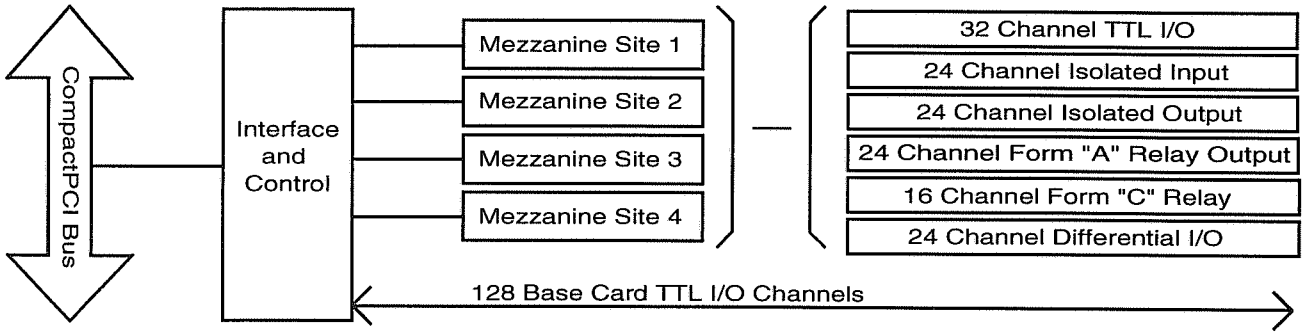
Digital input and output strobes are provided on both the base-board and the mezzanine cards. A strobe signal connected to any of the first 16 channels on the base board can be routed under software control to digital input channels. Programmable contact debounce suppression is provided to prevent false detection of digital input glitches.

SOFTWARE

The CP387 comes with a Plug and Play driver for configuring and using the device and application examples to illustrate its basic functionality.

This and other tools, including their source code, are provided.

CP387 Block Diagram



CP387 Specifications (Base-Board)

Item	Specifications
Number of Channels	128 Base-board Channels, and up to an additional 128 channels through expansion provided by the 4 mezzanine card sites.
I/O Type	Single-ended TTL
Direction Control	Yes, in Groups of 8 channels
Input Switching Threshold "0" Level "1" Level	 0.2 V maximum 2.2V minimum
Output Voltage Threshold "0" Level "1" Level	 0.4 V maximum (Iout = 2.5 mA) 2.7 V minimum (Iout = 2.5 mA)
Low Level Output Current	TBD
High Level Output Current	TBD
Input Current	TBD
Input Strobes Supported	Yes, Maximum of 16 (shared with output strobe count)
Input Strobe Polarity	Programmable
Input Strobe Pulse Width	200 nanoseconds minimum
Output Strobes Supported	Yes, Maximum of 16 (shared with input strobe count)
Output Strobe Polarity	Programmable
Output Strobe Delay	Programmable: 100 nanoseconds and 1 microsecond
Output Strobe Width	200 nanoseconds
Debouncing Rates	Programmable: Disabled, 1 microsecond, 1 millisecond, 10 milliseconds and 100 milliseconds
Input/Output Connector Type	3 Double-stacked 68 position High Density connectors
Power Requirements +5 V +3.3V +12 V -12 V	 TBD TBD TBD TBD
Environmental and Mechanical Temperature range Operational Storage Relative humidity Cooling requirements Dimensions Front-panel potential	 0°C to +50°C -25°C to +75°C 0 to 85%, non-condensing to 40°C 10 CFM 233.35 mm x 160 mm (6U CompactPCI module) Chassis ground

Technical specifications contained within this publication are subject to change without notice.

IP380 Specifications 32 Channel TTL Bi-directional I/O Mezzanine Module

Item	Specifications
Number of Channels	32
I/O Type	Single-ended TTL
Direction Control	Yes, in Groups of 8 channels
Input Switching Threshold "0" Level "1" Level	 0.2 V maximum 2.2V minimum
Output Voltage Threshold "0" Level "1" Level	 0.4 V maximum (Iout = 2.5 mA) 2.7 V minimum (Iout = 2.5 mA)
Low Level Output Current	TBD
High Level Output Current	TBD
Input Current	TBD
Input Strobe Supported	Yes
Input Strobe Polarity	Programmable
Input Strobe Pulse Width	200 nanoseconds minimum
Output Strobe Supported	Yes
Output Strobe Polarity	Programmable
Output Strobe Delay	Programmable: 100 nanoseconds and 1 microsecond
Output Strobe Width	200 nanoseconds
Debouncing Rates	Programmable: Disabled, 1 microsecond, 1 millisecond, 10 milliseconds and 100 milliseconds
Input/Output Connector Type	50 position Industry Pack Connector interface through the CP387 base board and routed to a 68 position high density connector.
Power Requirements +5 V +12V -12 V	 TBD TBD TBD

Technical specifications contained within this publication are subject to change without notice.

IP381 Specifications 24 Channel Isolated Input Mezzanine Module

Item	Specifications
Number of Channels	24
I/O Type	Isolated Input
Direction Control	Input Only
Voltage Levels Supported	5 V dc, 12V dc, 16 V dc, 24 V dc, 28 V dc, 48 V dc, or 120 V ac
Input Isolation	500 V
Input Current	Greater than 5 mA, less than 15 mA
Switching Threshold	Nominally one-half of the rated input
Input Strobe Supported	Yes
Input Strobe Polarity	Programmable
Input Strobe Pulse Width	TBD nanoseconds minimum
Debouncing Rates	Programmable: Disabled, 1 microsecond, 1 millisecond, 10 milliseconds and 100 milliseconds
Input/Output Connector Type	50 position Industry Pack Connector interface through the CP387 base board and routed to a 68 position high density connector.
Power Requirements	
+5 V	TBD
+12 V	TBD
-12 V	TBD

Technical specifications contained within this publication are subject to change without notice.

IP382 Specifications 24 Channel Isolated Output Mezzanine Module

Item	Specifications
Number of Channels	24
I/O Type	Optically Isolated Input
Direction Control	Output Only
Maximum Open Circuit Voltage	30 V
Maximum ON Current	10 mA
ON Voltage Drop	1V maximum
Off Current	1 µA maximum
Output Strobe Supported	Yes
Output Strobe Polarity	Programmable
Output Strobe Pulse Width	TBD nanoseconds minimum
Input/Output Connector Type	50 position Industry Pack Connector interface through the CP387 base board and routed to a 68 position high density connector.
Power Requirements +5 V +12 V -12 V	 TBD TBD TBD

IP383 Specifications 24 Channel Form "A" Relay Output

Item	Specifications
Number of Channels	24
I/O Type	Relay
Direction Control	Output Only
Output Configuration	1 Form "A" Relay
Maximum Open Circuit Voltage	100 V dc
Maximum Current	300 mA
Maximum Switched Load	TBD VA
Life Expectancy	TBD X 10^6 operations (with proper contact protection)
Contact Resistance	TBD mΩ, maximum
Operate Time	TBD mS, maximum
Release Time	TBD mS, maximum
Insulation Resistance	TBD MΩ, minimum
Contact Bounce	TBD mS
Output Strobe Supported	No
Input/Output Connector Type	50 position Industry Pack Connector interface through the CP387 base board and routed to a 68 position high density connector.
Power Requirements +5 V +12 V -12 V	 TBD TBD TBD

Technical specifications contained within this publication are subject to change without notice.

IP384 Specifications 16 Channel Form "C" Relay Output

Item	Specifications
Number of Channels	16
I/O Type	Relay
Direction Control	Output Only
Output Configuration	1 Form "C" Relay
Maximum Open Circuit Voltage	100 V
Maximum Current	300 mA
Maximum Switched Load	TBD VA
Life Expectancy	TBD X 10^6 operations (with proper contact protection)
Contact Resistance	TBD mΩ, maximum
Operate Time	TBD mS, maximum
Release Time	TBD mS, maximum
Insulation Resistance	TBD MΩ, minimum
Contact Bounce	TBD mS
Output Strobe Supported	No
Input/Output Connector Type	50 position Industry Pack Connector interface through the CP387 base board and routed to a 68 position high density connector.
Power Requirements	
+5 V	TBD
+12 V	TBD
-12 V	TBD

IP386 Specifications 16 Channel AC Switch Output Mezzanine Module

Item	Specifications
Number of Channels	16
I/O Type	AC Switched Output
Direction Control	Output Only
Zero Voltage Turn-on	10 V peak
Maximum Open Circuit Voltage	130 V RMS (200 V peak)
Maximum ON Current	300 mA, 47-70 Hz
Minimum ON Current	0.01 A
ON Voltage Drop	1.6V RMS, maximum (at rated current)
Turn-on Time (60 Hz)	8.3 mS, maximum
Turn-off Time (60 Hz)	8.4 mS, maximum
Input/Output Connector Type	50 position Industry Pack Connector interface through the CP387 base board and routed to a 68 position high density connector.
Power Requirements	
+5 V	TBD
+12 V	TBD
-12 V	TBD

IP385 Specifications 24 Channel Differential Bi-directional I/O Mezzanine Module

Item	Specifications
Number of Channels	24
I/O Type	Differential I/O conforming to RS-422 signal levels
Direction Control	Yes, in Groups of 8 Channels
Driver Termination	50 Ω in series with each leg
Receiver Termination	100 Ω across the differential path
Maximum Input Voltage	± 7 V
Input Resistance	6 kΩ
Differential Input Sensitivity	+2 V maximum, -2V minimum
Driver Output Voltage "0" Level "1" Level	 +0.5 V maximum, +2.5 V minimum
Driver short Circuit Current	-150 mA maximum, -30 mA minimum
Input Strobe Supported	Yes
Input Strobe Polarity	Programmable
Input Strobe Pulse Width	200 nanoseconds minimum
Output Strobe Supported	Yes
Output Strobe Polarity	Programmable
Output Strobe Delay	Programmable: 100 nanoseconds and 1 microsecond
Output Strobe Width	200 nanoseconds
Debouncing Rates	Programmable: Disabled, 1 microsecond, 1 millisecond, 10 milliseconds and 100 milliseconds
Input/Output Connector Type	50 position Industry Pack Connector interface through the CP387 base board and routed to a 68 position high density connector.
Power Requirements +5 V +12 V -12 V	 TBD TBD TBD

Technical specifications contained within this publication are subject to change without notice.

ORDERING INFORMATION

CP387-ZA11	256 Channel Digital I/O Base Board
IP380-AA11	32 Channel TTL Bi-directional I/O Mezzanine Module
IP381-AA11	24 Channel Isolated Input Mezzanine Module, 5V dc
IP381-AA21	24 Channel Isolated Input Mezzanine Module, 12V dc
IP381-AA31	24 Channel Isolated Input Mezzanine Module, 16V dc
IP381-AA41	24 Channel Isolated Input Mezzanine Module, 24V dc
IP381-AA51	24 Channel Isolated Input Mezzanine Module, 28V dc
IP381-AA61	24 Channel Isolated Input Mezzanine Module, 48V dc
IP381-AA71	24 Channel Isolated Input Mezzanine Module, 120V ac
IP382-AA11	24 Channel Isolated Output Mezzanine Module
IP383-AA11	24 Channel Form "A" Relay Output Mezzanine Module
IP384-AA11	16 Channel Form "C" Relay Output Mezzanine Module
IP385-AA11	24 Channel Differential Bi-directional I/O Mezzanine Module
IP386-AA11	16 Channel AC Switch Output Mezzanine Module

Specifications contained within this data sheet are subject to change without notice.

Updated July 26th, 2005

KineticSystems Company, LLC

900 N. State St.
Lockport, IL 60441-2200

Toll-Free (US and Canada):
phone 1-800-DATA NOW
1-800-328-2669

Direct:
phone +1-815-838-0005
fax +1-815-838-4424

Email:
mkt-info@kscorp.com

To find your local sales representative or distributor or to learn more about KineticSystems' products visit:

www.kscorp.com

Table Of Contents

Chapter 1: Introduction

Description

The CP387 is a single-width, 6U, CompactPCI/PXI module with up to 256 digital input/output (I/O) channels. The CP387 base board supports 128 channels of TTL I/O. Four mezzanine card sites can be populated with other forms of digital I/O including isolated input, isolated output, relay output, AC switch or differential I/O. The mezzanine card concept allows multiple digital I/O type to be configured within a single module to match the application requirements.

Pattern recognition and change-of-state detection are included. These operations are supported on the base board as well as the mezzanine cards. Input and output strobes are provided for connection to external sources.

CP387 Specifications

Item	Specifications
Number of Channels	256 single-ended TTL level signals maximum (128 base board plus up to 128 through 4 Mezzanine cards)
I/O Type	Single-ended TTL
Direction Control	Yes, in groups of 8 channels
Input Termination	Programmable: Disabled, Pulled-up, Pulled-down
Input switching threshold "0" Level "1" Level	 0.8 V maximum 2 V minimum
Output Voltage Level "0" Level "1" Level	 0.4 V maximum (Iout = 2.5mA) 2.7 V minimum (Iout = 2.5mA)
Low level output current	24mA, maximum
High level output current	-24mA, maximum
Input Current	5 mA (with input termination enabled)
Input Strobe Supported	Yes, maximum of 16 (shared with output strobes)
Input Strobe Polarity	Programmable
Input Strobe Pulse Width	200 nanoseconds minimum (debouncing disabled)
Input Debouncing Rate	Programmable: Disabled, 1 microsecond, 1 millisecond, 10 millisecond, 100 millisecond
Output Strobe Supported	Yes, maximum of 16 (shared with input strobes)
Output Strobe Polarity	Programmable
Output Strobe Pulse Width	200 nanoseconds
Output Strobe Delay	Programmable: 1 microsecond, 100 nanosecond
I/O Connector	3 double stacked 68 position Very High Density (VHD) receptacle connectors
Power Requirements +5 V +3.3V +12 V -12 V	 500 mA 300 mA Supplied to mezzanine sites only Supplied to mezzanine sites only
Environmental and Mechanical Temperature range Operational Storage	 0°C to +50°C -25°C to +75°C

Relative humidity	0 to 85%, non-condensing to 40°C
Cooling requirements	10 CFM
Dimensions	233.35 mm x 160 mm (6U CompactPCI module)
Front-panel potential	Chassis ground

Technical specifications contained within this publication are subject to change without notice.

Table 1-1. Specifications

Front Panel

Digital Input/Output Connector(s)

Access to the base board channels is via a single stacked VHD dual 68-position connector (64 channels per connector). The mezzanine channels are available on two-stacked VHD dual 68-position connector (50 contacts used per connector).

Front Panel Connector Pinout - Channels 1-64

Pin Number	Signal Description	Pin Number	Signal Description
1	Channel 1	35	Channel 33
2	Channel 2	36	Channel 34
3	Channel 3	37	Channel 35
4	Channel 4	38	Channel 36
5	Channel 5	39	Channel 37
6	Channel 6	40	Channel 38
7	Channel 7	41	Channel 39
8	Channel 8	42	Channel 40
9	Channel 9	43	Channel 41
10	Channel 10	44	Channel 42
11	Channel 11	45	Channel 43
12	Channel 12	46	Channel 44
13	Channel 13	47	Channel 45
14	Channel 14	48	Channel 46
15	Channel 15	49	Channel 47
16	Channel 16	50	Channel 48
17	Channel 17	51	Channel 49
18	Channel 18	52	Channel 50
19	Channel 19	53	Channel 51
20	Channel 20	54	Channel 52
21	Channel 21	55	Channel 53
22	Channel 22	56	Channel 54
23	Channel 23	57	Channel 55
24	Channel 24	58	Channel 56
25	Channel 25	59	Channel 57
26	Channel 26	60	Channel 58
27	Channel 27	61	Channel 59
28	Channel 28	62	Channel 60
29	Channel 29	63	Channel 61
30	Channel 30	64	Channel 62
31	Channel 31	65	Channel 63
32	Channel 32	66	Channel 64
33	Ground	67	Ground
34	Ground	68	Ground

Table 1-2. Front Panel Connector Pinout –Channels 1-64

Front Panel Connector Pinout – Channels 65-128

Pin Number	Signal Description	Pin Number	Signal Description
1	Channel 65	35	Channel 97
2	Channel 66	36	Channel 98
3	Channel 67	37	Channel 99
4	Channel 68	38	Channel 100
5	Channel 69	39	Channel 101
6	Channel 70	40	Channel 102
7	Channel 71	41	Channel 103
8	Channel 72	42	Channel 104
9	Channel 73	43	Channel 105
10	Channel 74	44	Channel 106
11	Channel 75	45	Channel 107
12	Channel 76	46	Channel 108
13	Channel 77	47	Channel 109
14	Channel 78	48	Channel 110
15	Channel 79	49	Channel 111
16	Channel 80	50	Channel 112
17	Channel 81	51	Channel 113
18	Channel 82	52	Channel 114
19	Channel 83	53	Channel 115
20	Channel 84	54	Channel 116
21	Channel 85	55	Channel 117
22	Channel 86	56	Channel 118
23	Channel 87	57	Channel 119
24	Channel 88	58	Channel 120
25	Channel 89	59	Channel 121
26	Channel 90	60	Channel 122
27	Channel 91	61	Channel 123
28	Channel 92	62	Channel 124
29	Channel 93	63	Channel 125
30	Channel 94	64	Channel 126
31	Channel 95	65	Channel 127
32	Channel 96	66	Channel 128
33	Ground	67	Ground
34	Ground	68	Ground

Table 1-3. Front Panel Connector Pinout – Channels 65-128

Front Panel Connector Pinout – Mezzanine Sites

The mezzanine sites are connected one for one (pins 1 through 50) to the 68 position VHD connectors. Please refer to individual mezzanine card pinouts for signal descriptions.

Product Ordering Information

CP387-ZA11	256 Channel Digital I/O Module
CP387-ZA21	256 Channel Digital I/O Module to Support High Current

Mezzanine Cards:

IP380-AA11	32 Channel TTL Bi-directional I/O Module
IP381-AA11	24 Channel Isolated Input Module, 5V DC
IP381-AA21	24 Channel Isolated Input Module, 12V DC
IP381-AA31	24 Channel Isolated Input Module, 16V DC
IP381-AA41	24 Channel Isolated Input Module, 24V DC
IP381-AA51	24 Channel Isolated Input Module, 28V DC
IP381-AA61	24 Channel Isolated Input Module, 48V DC
IP381-AA71	24 Channel Isolated Input Module, 120V AC
IP382-AA11	24 Channel Isolated Output Module
IP383-AA11	24 Channel Form "A" Relay Output Module
IP384-AA11	16 Channel Form "C" Relay Output Module
IP385-AA11	24 Channel Differential Bi-directional I/O Module
IP386-AA11	16 Channel AC Switch Output Module

Related Products

V765-ZA11	Rack Mount Termination Panel
5868-Txyz	Cable, 68P High Density SCSI to 68P VHD (For use with V765 Rack Mount Termination Panel)
5868-Sxyz	Cable, 68P VHD to unterminated
?????????	KineticSystems VISA layer Software
?????????	KineticSystems CPCI VISA Plug-in Software
?????????	KineticSystems 387 VXI/PXI Interworking Library Software

Chapter 2: Installation

⚠ Do not install hardware before installing accompanying software. Installing the software before the hardware ensures that the information in the module description file is available to the operating system when it needs to identify the hardware. A brief overview of the installation steps is as follows:

1. Install software.
2. Run the *Resource Manager* to register the module type with VISA.
3. Power the system down.
4. Install the module.
5. Power the system up. The operating system will automatically identify the new hardware and install kernel mode drivers.

Software Installation

The CP387 Plug and Play driver depends on an installed VISA layer. This procedure assumes that VISA has already been installed.

1. Insert the accompanying CD into your system and run setup.exe. This will install the Plug and Play driver code and libraries, as well as the module.ini file.
2. Run the VISA *Resource Manager* tool. The *Resource Manager* will identify the newly installed module.ini file and register the module type with VISA and build appropriate kernel mode driver files for the operating system.

Directory Structure

Software installation will place files as described below. `<VXIPNP>` denotes where VISA is installed (e.g., by default `C:\vxipnp\winnt` on a Windows based machine).

- `<VXIPNP>\include`: `ks387.h` (API header file)
- `<VXIPNP>\include`: `ksipapi.h` (IP API header file)
- `<VXIPNP>\include`: `ksp387.bas` (VB header file)
- `<VXIPNP>\bin`: `ksp387.dll` (API library)
- `<VXIPNP>\lib\<format>`: (API lib file `ksp387.lib`, in various formats)
- `<VXIPNP\ksp387`: `ksp387.c` (API source code)
- `<VXIPNP\ksp387`: `ks387.fp` (Function Panel)
- `<VXIPNP\ksp387`: `ks387.hlp` (API help file)

In addition, the module_cp387.ini file will be installed in the directory specified by registry setting `HKEY_LOCAL_MACHINE\SOFTWARE\PXISA\CURRENT_VERSION`, value `ModuleDescriptionFilePath`, or simply `<VXIPNP>` if the registry value is not set or does not exist.

Manual Registration with VISA

In the event that your VISA *Resource Manager* does not or cannot automatically register the CP387 with VISA via the module.ini file, you will probably need to manually register it with VISA. This will probably be accomplished by running a tool or wizard distributed with your VISA; consult your VISA documentation for details.

To manually register the CP387 with VISA, you will need the following information:

- Module Name: "CP387"
- Module Vendor: "KineticSystems Company, LLC"
- Model Code: 0x387
- Manufacturer Code: 0x11f4
- Interrupt Detect and Quiesce: The CP387 generates 3 sources of interrupt.
  - Detect: true if a 32 bit read of space BAR2, offset 0xd8 returns bits 0, 1, or 2 on.
  - Quiesce: To deassert the interrupt, perform a 32 bit write of value 0x7 to space BAR2, offset 0xd8

  In PXI terms:

```
NumDetectSequences = 3
InterruptDetect0 = "C32 BAR2 0xd8 0x1 0x1;"
InterruptDetect1 = "C32 BAR2 0xd8 0x2 0x2;"
InterruptDetect2 = "C32 BAR2 0xd8 0x4 0x4;"
InterruptQuiesce = "W32 BAR2 0xd8 0x7;"
```

Unpacking the CP387

The CP387 comes in an anti-static bag to avoid electrostatic damage to the module. Please take the following precautions when unpacking the module:

- Ground yourself with a grounding strap or by touching a grounded object.
- Touch the anti-static package to a metal part of your CompactPCI chassis before removing the module from the package.
- Remove the module from the package and inspect the module for damage.
- Do not install the module into the CompactPCI chassis until you are satisfied that the module exhibits no obvious mechanical damage and is configured to conform to the desiring operating environment.

Module Insertion

Chapter 3: Device Operation

Overview

The CP387 is a single width, 6U, CompactPCI module with 128 TTL level digital I/O channels. Expansion of up to 128 digital channels is provided by 4 mezzanine card sites. These sites can be populated with an assortment of I/O options to extend the capability of the CP387.

Data transfers to and from the CP387 are under programmed control and support 16 and 32 bit data words. Transfers to and from the base board channels are executed directly. Transfers to the mezzanine card sites are translated by base board logic. The mezzanine card sites use an Industry Pack (IP) form factor and require the translation of PCI bus cycles to IP bus cycles.

Accessing digital output channels is accomplished by writing the selected data values to a register location. Reading digital input channels is also accomplished by simple register read operations.

The CP387 contains flash memory that can be used to restore basic digital input and output configuration parameters on power up. These parameters include the connection of either pull-up or pull-down resistors, the direction of digital I/O channels, and the initial power-up values of digital output channels.

Pattern recognition and change-of-state operations are supported by the CP387 base board as well as by mezzanine cards. The base board performs these two operations by latching the digital input data from all sources and then performing a compare for the selected operation. If enabled, an interrupt can be generated at the detection of a pattern recognition or change-of-state operation.

Digital input and output strobes are provided on both the base board and the mezzanine cards. A strobe signal connected to any of the first 16 channels on the base board can be routed under software control to digital input channels. Programmable contact debounce suppression is provided to prevent false detection of digital input glitches.

Below is a simplified block diagram of the CP387.

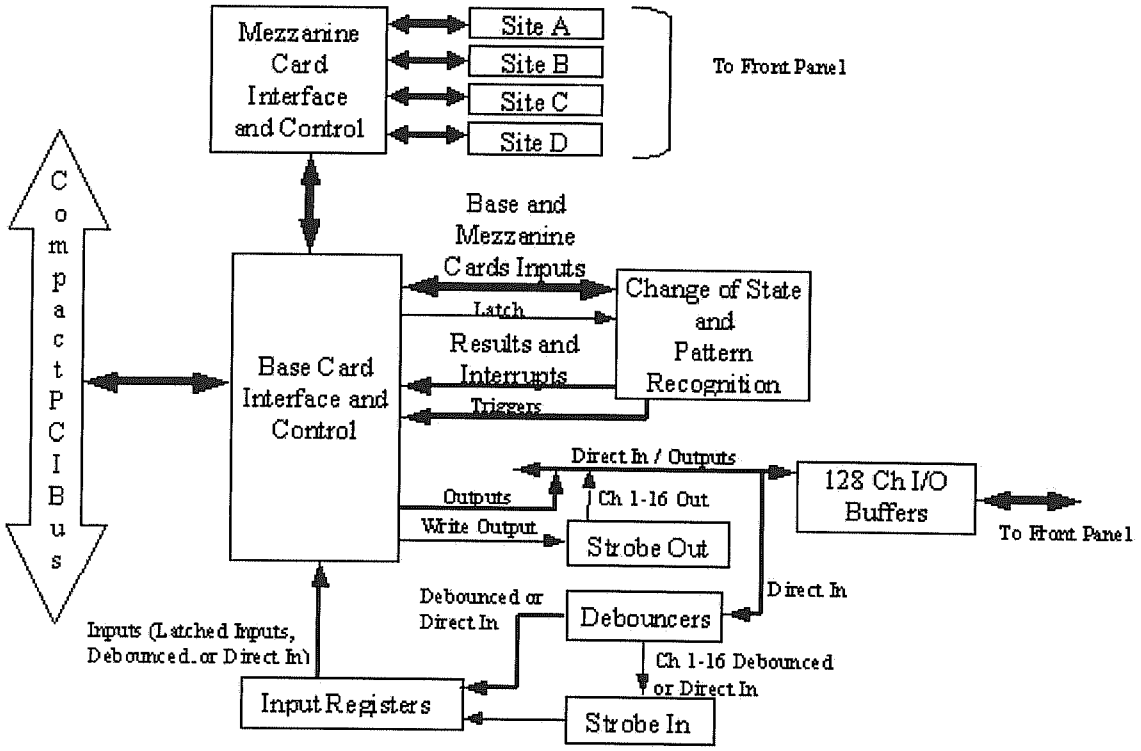


Figure 3-1. CP387 Block Diagram

Basic Circuit Operation

The CP387 can accept and source 128 TTL level signals. The 128 channels are divided into 8-channel groups. Each 8-channel group can be designated as an input or an output. Inputs can be connected to pull-up or pull-down resistors, or be left unterminated. This allows for unused inputs to be at a known value. Power-up/reset default for the CP387 is all channels configured as inputs with no input termination.

Input data comes from one of three sources depending on how the module is configured. The default input data source is Direct Input. Direct Input is the raw TTL data from the external source. The second source for input data is Debounced Data. Debouncing is a method used to clean input signals of unwanted glitches. A programmed debounce time defines how long an input signal must be at a valid (high or low) value until the signal is made available for reading. The time can be programmed for 1µS, 1mS, 10mS or 100mS. The third source for input data is Strobed Data. The first 16 channels can be programmed to be input strobes, one for each of the 8-channel groups. This allows an external source to latch or control the input of data to the CP387. The polarity (whether inputs are latched on the rising edge or falling edge) of the strobe can also be programmed via the *Plug and Play* (PNP) driver. Output strobes can also be produced to latch data into an external device. The output strobe is produced when the output data is written and can be

delayed by either 100nS or 1μS and is approximately 200nS wide. Each of the 8-channel groups can be programmed to route out the strobe to any of the first 16 channels. The output strobe polarity can also be programmed based on the requirements of the external device (rising or falling edge).

The CP387 has the ability to store configuration information in flash memory so that on power-up/reset the configured parameters can be restored. This allows the CP387 to power-up/reset to a user-defined configuration instead of the factory default. Parameters such as I/O direction (input or output), initial output value (high or low), and input termination (pull-up, pull-down, or unterminated) are stored.

The CP387 has the ability to perform several operations on the digital inputs of the module. These operations include pattern recognition and change-of-state. Each channel can be individually programmed to enable these functions. All 128 channels on the base board, as well as the additional 128 channels on the mezzanine card (if supported), can be programmed for pattern recognition and change-of-state. The CP387 produces a signal for both pattern recognition and change-of-state to latch the input data from all sources. For pattern recognition all enabled inputs are compared to the stored pattern data. When there is a match, an event notification and an interrupt, if enabled, will be generated. For change-of-state, a present value is initially stored for all enabled inputs. The CP387 continues to monitor the inputs until a change from the present value is detected. When detected, an event notification and an interrupt, if enabled, will be generated.

Chapter 4: Programming

The following functions are provided with the CP387 *Plug and Play* instrument driver.

Required Functions

ks387_init

Syntax:

```
ViStatus ks387_init (ViRsrc resourceName, ViBoolean IDQuery,
                     ViBoolean resetDevice,
                     ViPSession instrumentHandle);
```

Description:

ks387_init establishes communication with a CP387 and optionally resets and queries the module. Once these operations are complete, an instrument handle is returned for subsequent communication with the module. For each instrument handle returned via ks387_init, a corresponding ks387_close should be executed prior to termination of the application program.

Parameters:

resourceName

Variable Type ViRsrc

Instrument description.

IDQuery

Variable Type ViBoolean

If (VI_TRUE) perform in-system verificationIf (VI_FALSE) do not perform in-system verification.

resetDevice

Variable Type ViBoolean

If (VI_TRUE) perform reset operation; the reset will be a soft reset (see ks387_resetEx)
If (VI_FALSE) do not perform reset operation

instrumentHandle

Variable Type ViSession (passed by reference)

Instrument handle

Return Values:

Standard function status codes

ks387_reset

Syntax:

```
ViStatus ks387_reset (ViSession instrumentHandle);
```

Description:

`ks387_reset` performs a reset operation on the instrument, returning it to its power-up state. The reset is a soft reset, as described under `ks387_resetEx`; see the description for `ks387_resetEx` for details on the distinction between soft and hard resets.

Parameters:

instrumentHandle

Variable Type ViSession

Instrument handle

Return Values:

Standard function status codes

ks387_self_test

Syntax:

```
ViStatus ks387_self_test (ViSession instrumentHandle,
                          ViPInt16 selfTestResults,
                          ViChar _VI_FAR selfTestMessage[]);
```

Description:

`ks387_self_test` runs the module self-test and returns the result.

Parameters:

instrumentHandle

Variable Type ViSession

Instrument handle

selfTestResults

Variable Type ViInt16 (passed by reference)

Numeric result from self-test operation0 = no error (test passed)

selfTestMessage

Variable Type ViChar []

Self-test status message

Return Values:

Standard function status codes

ks387_error_query

Syntax:

```
ViStatus ks387_error_query (ViSession instrumentHandle,
                            ViPInt16 errorCode,
                            ViChar _VI_FAR errorMessage[]);
```

Description:

`ks387_error_query` is not supported on the CP387.

Parameters:

instrumentHandle

Variable Type ViSession

Instrument handle

errorCode

Variable Type ViInt16 (passed by reference)

Instrument error code

errorMessage

Variable Type ViChar []

Error message; must be at least 256 characters

Return Values:

Standard function status codes.

ks387_error_message

Syntax:

```
ViStatus ks387_error_message (ViSession instrumentHandle,
                              ViInt16 statusCode,
                              ViChar _VI_FAR message[]);
```

Description:

ks387_error_message takes an error code returned from some other CP387 Instrument Driver API function and translates it to a readable text message.

Parameters:

instrumentHandle

Variable Type ViSession

instrument handle

statusCode

Variable Type ViInt16

instrument error code returned from another API function.

message

Variable Type ViChar []

Error message; must be at least 256 characters

Return Values

Standard function status codes

ks387_revision_query

Syntax:

```
ViStatus ks387_revision_query (
                        ViSession instrumentHandle,
                        ViChar _VI_FAR instrumentDriverRevision[],
                        ViChar _VI_FAR firmwareRevision[]);
```

Description:

`ks387_revision_query` returns the revision of the instrument driver and the firmware revision of the instrument.

Parameters:

instrumentHandle

Variable Type ViSession

Instrument handle

instrumentDriverRevision

Variable Type ViChar[]

Instrument driver revision; must be at least 256 characters.

firmwareRevision

Variable Type ViChar []

Instrument firmware revision; must be at least 256 characters.

Return Values:

Standard function status codes

ks387_close

Syntax:

```
ViStatus ks387_close (ViSession instrumentHandle);
```

Description

`ks387_close` terminates the software connection to the instrument and deallocates system resources associated with that instrument.

Parameters:

instrumentHandle

Variable Type ViSession

Instrument handle

Return Values

Standard function status codes

Optional Functions

ks387_autoConnectToFirst

Syntax:

```
ViStatus ks387_autoConnectToFirst (ViPSession instrumentHandle);
```

Description:

`ks387_autoConnectToFirst` searches the system for a CP387 module, opens communication with the module, and returns a session handle. The module selected is arbitrary see `ks387_autoConnectToAll` to obtain sessions to more than 1 instrument.

Parameters:

instrumentHandle

Variable Type ViSession (passed by reference)

Instrument handle

Return Values:

Standard function status codes

ks387_autoConnectToAll

Syntax:

```
ViStatus ks387_autoConnectToAll (
                        ViSession _VI_FAR instrumentHandles[],
                        ViInt16 inputArray_length,
                        ViPInt16 numberFound);
```

Description:

`ks387_autoConnectToAll` attempts to find up to arrayLength CP387 module(s) modules in the system and returns open sessions to them.

Parameters:

instrumentHandles

Variable Type ViSession []

Array of session handles to be populated; must have at least arrayLength elements.

inputArray_length

Variable Type ViInt16

Number of elements to populate in vi array.

numberFound

Variable Type ViInt16 (passed by reference)

Number of instruments handled populated in vi array.

Return Values:

Standard function status codes

Configuration Functions

ks387_autoRst

Syntax:

```
ViStatus ks387_autoRst (ViSession instrumentHandle,
                        ViBoolean restoreEnable);
```

Description:

`ks387_autoRst` specifies the power-up state of the instrument. By default, the instrument powers up with all channels set as unterminated inputs. This function allows the user to save the current I/O direction (input/output) of each channel, the output level (high/low) for output channels, and the input termination (unterminated/pulled up/pulled down) for input channels. These current settings will by automatically applied on the next power cycle (or reset; see `ks387_resetEx`). Output levels are not saved from the current settings, but are explicitly set via `ks387_setPowerupOutput`).

Parameters:

instrumentHandle

Variable Type ViSession

Instrument handle

restoreEnable

Variable Type ViBoolean

If set to VI_FALSE, any currently saved power-up settings are discarded, and all channels will be set to unterminated inputs on next power-up/reset. Current settings are not affected. If set to VI_TRUE, all current I/O direction, output level, and input termination settings are saved, and will automatically be reapplied on the next power-cycle/reset operation.

Return Values:

Standard function status codes

ks387_autoRstStatus

Syntax:

```
ViStatus ks387_autoRstStatus (ViSession instrumentHandle,
                              ViPBoolean restoreStatus);
```

Description:

`ks387_autoRstStatus` returns the power-up state of the instrument. See `ks387_autoRst()` for details.

Parameters:

instrumentHandle

Variable Type ViSession

Instrument handle

restoreStatus

Variable Type ViBoolean (passed by reference)

Returns VI_TRUE if the device restored a saved configuration on the last powerup or reset. Returns VI_FALSE if the device did not restore a saved configuration on last powerup or reset.

Return Values

Standard function status codes

ks387_resetEx

Syntax:

```
ViStatus ks387_resetEx (ViSession instrumentHandle,
                        ViUInt16 resetType);
```

Description:

`ks387_resetEx` sets the power-up mode of the instrument and then performs a reset. The CP387 has 2 different types of reset: soft and hard. Soft reset returns the settings of all channels to the saved configuration. (see `ks387_autoRst` for details). Hard reset removes any stored configuration and then does the reset procedure; all channels are configured as unterminated inputs. This represents the state of the instrument when shipped from the factory. The standard `ks387_reset` function does the equivalent of a soft reset.

Parameters:

instrumentHandle

Variable Type ViSession

Instrument handle

resetType

Variable Type ViUInt16

If set to KS387RESETTYPESOFT, all channels will be configured to the saved configuration (if any). See `ks387_autoRst`. If set to KS387RESETTYPEHARD, any saved configuration will be removed before the reset is applied; all channels are configured as unterminated inputs.

Return Values:

Standard function status codes

ks387_getNumberSites

Syntax:

```
ViStatus ks387_getNumberSites (ViSession instrumentHandle,
                               ViPUInt16 numberSites);
```

Description:

`ks387_getNumberSites` returns the number of mezzanine sites supported by this 387. The number returned is the total supported, not the number actually occupied.

Parameters:

instrumentHandle

Variable Type ViSession

Instrument handle

numberSites

Variable Type ViUInt16 (passed by reference)

Variable to be populated with the number of sites supported.

Return Values:

Standard function status codes

ks387_getSiteChannels

Syntax:

```
ViStatus ks387_getSiteChannels (ViSession instrumentHandle,
                                ViUInt16 site,
                                ViPUInt16 firstChannel,
                                ViPUInt16 lastChannel);
```

Description:

`ks387_getSiteChannels` returns the first and last channel on a given mezzanine site. No that site 0 is the 387 baseboard itself. Sites 1 through N (where N is the number of supported sites; see `ks387_getNumberSites`) refer to the mezzanine sites.

Parameters:

instrumentHandle

Variable Type ViSession

Instrument handle site

Variable Type ViUInt16

The mezzanine site to query. Site '0' corresponds to the 387 board itself. Sites '1' through 'N' are the mezzanine sites proper.

firstChannel

Variable Type ViUInt16 (passed by reference)

The first channel located on this site.

lastChannel

Variable Type ViUInt16 (passed by reference)

The last channel located on this site.

Return Values:

Standard function status codes. In addition to standard return codes, `VI_ERROR_INV_SITE` is returned if an out of range site value is given. `VI_ERROR_UNOCCUPIED_SITE` is returned if a valid but unoccupied site is given.

ks387_getSites

Syntax:

```
ViStatus ks387_getSites (ViSession instrumentHandle,
                         ViPUInt32 occupiedSites);
```

Description:

Return a bitmap of all the occupied mezzanine sites.

Parameters:

instrumentHandle

Variable Type ViSession

Instrument Handle

occupiedSites

Variable Type ViUInt32 (passed by reference)

Bitmap of occupied mezzanine sites, where each bit is populated with a '1' if the site is occupied, and '0' if not occupied. The API header file (ks387.h) contains macro KS387MEZZANINESITE() used to test the bitmap to determine if a given site is occupied.

Return Values:

Standard function status codes

ks387_setDebounceTime

Syntax:

```
ViStatus ks387_setDebounceTime (ViSession instrumentHandle,
                                ViUInt16 firstChannel,
                                ViUInt16 lastChannel,
                                ViUInt32 timeRequested,
                                ViPUInt32 actualTime);
```

Description:

`ks387_setDebounceTime` sets the debounce time of a range of channels to a given value. The Debounce time is the minimum amount of time a transition on an input must persist before it is 'detected'. Noise in input channels can be removed by setting the debounce time to a value greater than the duration of the bursts of noise. Debouncing minimizes the effects of contact closures. Because of limitations in the hardware, only certain time durations are supported. The closest supported duration will be used. The 'pTimeActual' value is populated with the exact value the hardware was set to.

Parameters:

instrumentHandle

Variable Type ViSession

Instrument Handle

firstChannel

Variable Type ViUInt16

The first channel in a range to set the given debounce time to.

lastChannel

Variable Type ViUInt16

The last channel in a range to set the given debounce time to.

timeRequested

Variable Type ViUInt32

The debounce time in microseconds. Time of 0 is disable. Since hardware may not be able to support rate requested, time actually set is returned (if pTimeActual is not NULL).

actualTime

Variable Type ViUInt32 (passed by reference)

Supported time value hardware actually set to. This pointer can be NULL, in which case no value is returned.

Return Values:

Standard function status codes. In addition to standard return codes, `VI_WARN_BEST_TIME` is returned to indicate that the input time is not supported, and a closest value was substituted. `VI_ERROR_CHAN_ALIGN` is returned if range of input channels violates the channel grouping restrictions.

ks387_getDebounceTime

Syntax:

```
ViStatus ks387_getDebounceTime (ViSession instrumentHandle,
                                ViUInt16 firstChannel,
                                ViUInt16 lastChannel,
                                ViPUInt32 debounceTimeArray);
```

Description:

ks387_getDebounceTime retrieves the debounce times for a block of channels. Time is in microseconds. Time of 0 is disable. Each channel in requested range populates an element in pDebounceTime array. pDebounceTime must be sized to accommodate the number of channels.

Parameters:

instrumentHandle

Variable Type ViSession

Instrument Handle

firstChannel

Variable Type ViUInt16

The first channel in a range to set the given debounce time to.

lastChannel

Variable Type ViUInt16

The last channel in a range to set the given debounce time to.

debounceTimeArray

Variable Type ViUInt32 (passed by reference)

An array to be populated with each channel's current debounce time setting.

Return Values:

Standard function status codes.

ks387_setIoDirection

Syntax:

```
ViStatus ks387_setIoDirection (ViSession instrumentHandle,
                               ViUInt16 firstChannel,
                               ViUInt16 lastChannel,
                               ViUInt16 direction);
```

Description:

ks387_setIoDirection sets the direction of a block of channels to either input or output. Note that not all channels are I/O direction programmable; consult the hardware manual for your 387 and/or installed mezzanine cards.

Parameters:

instrumentHandle

Variable Type ViSession

Instrument Handle

firstChannel

Variable Type ViUInt16

The first channel in a range to set the given Input/Output direction to.

lastChannel

Variable Type ViUInt16

The last channel in a range to set the given Input/Output direction to.

direction

Variable Type ViUInt16

Set to either KS387IODIRINPUT to set the block of channels to read input or KS387IODIROUTPUT to set the block of channels to drive an output.

Return Values:

Standard function status codes In addition to standard return codes, VI_ERROR_CHAN_ALIGN is returned if range of input channels violates the channel grouping restrictions.

ks387_getIoDirection

Syntax:

```
ViStatus ks387_getIoDirection (ViSession instrumentHandle,
                               ViUInt16 firstChannel,
                               ViUInt16 lastChannel,
                               void *bitmap_pointer);
```

Description:

`ks387_getIoDirection` retrieves the Input/Output settings of a block of channels. The result is returned as a bitmap, where bit 0 of the first byte of the pDir array represents the first channel, bit 1 represents the second channel, etc. Each bit is set according to the binary equivalent of KS387IODIRINPUT or KS387IODIROUTPUT

Parameters:

instrumentHandle

Variable Type ViSession

Instrument Handle

firstChannel

Variable Type ViUInt16

The first channel in a range to get the Input/Output direction of.

lastChannel

Variable Type ViUInt16

The last channel in a range to get the Input/Output direction of.

bitmap_pointer

Variable Type void (passed by reference)

Pointer to a user-allocated array. The array must be sized appropriately to accommodate 1 bit per channel requested.

Return Values:

Standard function status codes

ks387_setPowerupOutput

Syntax:

```
ViStatus ks387_setPowerupOutput (ViSession instrumentHandle,
                                 ViUInt16 firstChannel,
                                 ViUInt16 lastChannel,
                                 ViUInt16 powerupState);
```

Description:

ks387_setPowerupOutput sets the powerup state of a block of channels. This function has no immediate effect. When the 387 resets or powers up, however, output channels will automatically be placed in the state specified by this call if auto restoration is activated (see ks387_autoRst).

Parameters:

instrumentHandle

Variable Type ViSession

Instrument Handle

firstChannel

Variable Type ViUInt16

The first channel in a range to specify the powerup state.

lastChannel

Variable Type ViUInt16

The last channel in a range to specify the powerup state.

powerupState

Variable Type ViUInt16

Set to KS387PWRCFGLOW to powerup to the low state, or KS387PWRCFGHIGH to powerup to the high state.

Return Values:

Standard function status codes. In addition to standard return codes, VI_ERROR_CHAN_ALIGN is returned if range of input channels violates the channel grouping restrictions.

ks387_getPowerupOutput

Syntax:

```
ViStatus ks387_getPowerupOutput (ViSession instrumentHandle,
                                 ViUInt16 firstChannel,
                                 ViUInt16 lastChannel,
                                 void *bitmap_pointer);
```

Description:

`ks387_getPowerupState` retrieves the powerup state of a block of channels. The result is returned as a bitmap, where bit 0 of the first byte of the pState array represents the first channel, bit 1 represents the second channel, etc. Each bit is set according to the binary equivalent of KS387PWRCFGLOW or KS387PWRCFGHIGH.

Parameters:

instrumentHandle

Variable Type ViSession

Instrument Handle

firstChannel

Variable Type ViUInt16

The first channel in a range to get the powerup state of.

lastChannel

Variable Type ViUInt16

The last channel in a range to get the powerup state of.

bitmap_pointer

Variable Type void (passed by reference)

Pointer to a user-allocated array. The array must be sized appropriately to accommodate 1 bit per channel requested.

Return Values:

Standard function status codes

ks387_setResistorConfig

Syntax:

```
ViStatus ks387_setResistorConfig (ViSession instrumentHandle,
                                  ViUInt16 firstChannel,
                                  ViUInt16 lastChannel,
                                  ViUInt16 inputTermination);
```

Description:

`ks387_setResistorConfig` sets how input channels are terminated. Each input channel can be pulled up to high, pulled down to low, or left unconnected.

Parameters:

instrumentHandle

Variable Type ViSession

Instrument Handle

firstChannel

Variable Type ViUInt16

The first channel in a range to specify the termination.

lastChannel

Variable Type ViUInt16

The last channel in a range to specify the termination.

inputTermination

Variable Type ViUInt16

Set to KS387TERMUP to pull up to high, KS387TERMDOWN to pull down to low, or KS387TERMNONE to leave unterminated.

Return Values:

Standard function status codes. In addition to standard return codes, `VI_ERROR_CHAN_ALIGN` is returned if range of input channels violates the channel grouping restrictions.

ks387_getResistorConfig

Syntax:

```
ViStatus ks387_getResistorConfig (ViSession instrumentHandle,
                                  ViUInt16 firstChannel,
                                  ViUInt16 lastChannel,
                                  ViPUInt16 arrayPointer);
```

Description:

`ks387_getResistorConfig` returns the input termination states of a block of channels. Each element in the pState array will be populated with either KS387TERMNONE, KS387TERMUP or KS387TERMDOWN. The user allocated pState array must be appropriately sized for 1 16 bit value for each channel requested.

Parameters:

instrumentHandle

Variable Type ViSession

Instrument Handle

firstChannel

Variable Type ViUInt16

The first channel in a range to get input termination configuration of.

lastChannel

Variable Type ViUInt16

The last channel in a range to get input termination configuration of.

arrayPointer

Variable Type ViUInt16 (passed by reference)

Pointer to a user-allocated array. The array must be sized appropriately to accommodate 1 16 bit value per channel requested.

Return Values:

Standard function status codes

ks387_readPatternRecognition

Syntax:

```
ViStatus ks387_readPatternRecognition(ViSession InstrumentHandle,
                                      ViUInt16 firstChannel,
                                      ViUInt16 lastChannel,
                                      void *bitmap_pointer);
```

Description:

ks387_readPatternRecognition retrieves the matching pattern for a block of channels. The result is returned as a bitmap, where bit 0 of the first byte of the pData array represents the first channel, bit 1 represents the second channel, etc.

Parameters:

instrumentHandle

Variable Type ViSession

Instrument Handle

firstChannel

Variable Type ViUInt16

The first channel in a range to get the match pattern of.

lastChannel

Variable Type ViUInt16

The last channel in a range to get the match pattern of.

bitmap_pointer

Variable Type void (passed by reference)

Pointer to a user-allocated array. The array must be sized appropriately to accommodate 1 bit per channel requested.

Return Values:

Standard function status codes

ks387_writePatternRecognition

Syntax:

```
ViStatus ks387_writePatternRecognition(
                                        ViSession instrumentHandle,
                                        ViUInt16 firstChannel,
                                        ViUInt16 lastChannel,
                                        void matchPatternArray);
```

Description:

ks387_writePatternRecognition sets the match pattern for a block of channels.

Parameters:

instrumentHandle

Variable Type ViSession

Instrument Handle

firstChannel

Variable Type ViUInt16

The first channel in a range to specify the match pattern.

lastChannel

Variable Type ViUInt16

The last channel in a range to specify the match pattern.

matchPatternArray

Variable Type void

A bitmap of match pattern bits. Bit 0 of the first byte is the pattern that will match the first channel in the block, the second bit is the pattern the second channel will match, etc.

Return Values:

Standard function status codes

ks387_readPatternRecognitionMask

Syntax:

```
ViStatus ks387_readPatternRecognitionMask (ViSession instrumentHandle,
                                           ViUInt16 firstChannel,
                                           ViUInt16 lastChannel,
                                           void *bitmap_pointer);
```

Description:

ks387_readPatternRecognitionMask retrieves the matching pattern mask for a block of channels. The result is returned as a bitmap, where bit 0 of the first byte of the pData array represents the first channel, bit 1 represents the second channel, etc. A '1' bit in the mask indicates that the corresponding channel will participate in the pattern match operation, whereas a '0' bit in the mask indicates that the channel will not participate (input state is 'don't care').

Parameters:

instrumentHandle

Variable Type ViSession

Instrument Handle

firstChannel

Variable Type ViUInt16

The first channel in a range to get the match pattern mask of.

lastChannel

Variable Type ViUInt16

The last channel in a range to get the match pattern mask of.

bitmap_pointer

Variable Type void (passed by reference)

Pointer to a user-allocated array. The array must be sized appropriately to accommodate 1 bit per channel requested.

Return Values:

Standard function status codes

ks387_writePatternRecognitionMask

Syntax:

```
ViStatus ks387_writePatternRecognitionMask (ViSession instrumentHandle,
                                            ViUInt16 firstChannel,
                                            ViUInt16 lastChannel,
                                            void matchPatternMaskArray);
```

Description:

`ks387_writePatternRecognitionMask` sets the match pattern mask for a block of channels. A bitmap is input, where bit 0 of the first byte of the pData array represents the first channel, bit 1 represents the second channel, etc. A '1' bit in the mask indicates that the corresponding channel will participate in the pattern match operation, whereas a '0' bit in the mask indicates that the channel will not participate (input state is 'don't care').

Parameters:

instrumentHandle

Variable Type ViSession

Instrument Handle

firstChannel

Variable Type ViUInt16

The first channel in a range to specify the match pattern mask.

lastChannel

Variable Type ViUInt16

The last channel in a range to specify the match pattern mask.

matchPatternMaskArray

Variable Type void (passed by reference)

A bitmap of mask bits. Bit 0 of the first byte is the mask of the first channel in the block, the second bit is the mask the second channel will match, etc. A '1' bit in the mask indicates that the corresponding channel will participate in the pattern match operation, whereas a '0' bit in the mask indicates that the channel will not participate (input state is 'don't care').

Return Values:

Standard function status codes

ks387_startPatternRecognition

Syntax:

```
ViStatus ks387_startPatternRecognition (ViSession instrumentHandle);
```

Description:

`ks387_startPatternRecognition` directs the 387 to begin scanning for the pattern specified by `ks387_writePatternRecognition`. Only channels masked 'on' ('1') via `ks387_writePatternRecognitionMask` are scanned. The status of the scanning operation can be polled via `ks387_getPatternRecognitionStatus` or `ks387_enableEvent` can be used to enable the 387 to generate a VISA event when the pattern is matched.

Parameters:

instrumentHandle

Variable Type ViSession

Instrument Handle

Return Values:

Standard function status codes. In addition to standard return codes, Returns `VI_ERROR_RSRC_BUSY` if recognition operation is already is progress. Returns `VI_ERROR_INV_SETUP` if strobing is enabled on any unmasked channel.

ks387_stopPatternRecognition

Syntax:

```
ViStatus ks387_stopPatternRecognition (ViSession instrumentHandle);
```

Description:

`ks387_stopPatternRecognition` stops the 387 to stop scanning for the pattern specified by `ks387_writePatternRecognition`.

Parameters:

instrumentHandle

Variable Type ViSession

Instrument Handle

Return Values:

Function status code In addition to standard return codes, Returns `VI_ERROR_ABORT` if no recognition operation is in progress.

ks387_getPatternRecognitionStatus

Syntax:

```
ViStatus ks387_getPatternRecognitionStatus (ViSession instrumentHandle);
```

Description:

`ks387_getPatternRecognitionStatus` returns the status of the pattern match operation initiated via `ks387_startPatternRecognition`.

Parameters:

instrumentHandle

Variable Type ViSession

Instrument Handle

Return Values:

Function status code `VI_SUCCESS` is returned if the operation is complete and the pattern was matched. `VI_SUCCESS_QUEUE_EMPTY` is returned if this function has already returned `VI_SUCCESS`, but no start operation has been subsequently performed.
`VI_SUCCESS_EVENT_EN` is returned if a pattern match operation has started and is in progress.
`VI_SUCCESS_EVENT_DIS` is returned if no pattern match operation is in progress or successfully completed (i.e., none ever started or last one was stopped).

ks387_readChangeOfStateResult

Syntax:

```
ViStatus ks387_readChangeOfStateResult (ViSession instrumentHandle,
                                        ViUInt16 firstChannel,
                                        ViUInt16 lastChannel,
                                        void *bitmap_pointer);
```

Description:

`ks387_readChangeOfStateResult` retrieves the input pattern which triggered a change of state. The result is returned as a bitmap, where bit 0 of the first byte of the pData array represents the state of the first channel, bit 1 represents the state of the second channel, etc.

Parameters:

instrumentHandle

Variable Type ViSession

Instrument Handle

firstChannel

Variable Type ViUInt16

The first channel in a range to get the change of state result.

lastChannel

Variable Type ViUInt16

The last channel in a range to get the change of state result.

bitmap_pointer

Variable Type void (passed by reference)

Pointer to a user-allocated array. The array must be sized appropriately to accommodate 1 bit per channel requested.

Return Values:

Standard function status codes

ks387_readChangeOfStateMask

Syntax:

```
ViStatus ks387_readChangeOfStateMask (ViSession instrumentHandle,
                                      ViUInt16 firstChannel,
                                      ViUInt16 lastChannel,
                                      void *bitmap_pointer);
```

Description:

`ks387_readChangeOfStateMask` retrieves the change of state mask for a block of channels. The result is returned as a bitmap, where bit 0 of the first byte of the pData array represents the first channel, bit 1 represents the second channel, etc. A '1' bit in the mask indicates that the corresponding channel will participate in the change of state operation, whereas a '0' bit in the mask indicates that the channel will not participate (input state is 'don't care').

Parameters:

instrumentHandle

Variable Type ViSession

Instrument Handle

firstChannel

Variable Type ViUInt16

The first channel in a range to get the change of state mask of.

lastChannel

Variable Type ViUInt16

The last channel in a range to get the change of state mask of.

bitmap_pointer

Variable Type void (passed by reference)

Pointer to a user-allocated array. The array must be sized appropriately to accommodate 1 bit per channel requested.

Return Values:

Standard function status codes

ks387_writeChangeOfStateMask

Syntax:

```
ViStatus ks387_writeChangeOfStateMask (ViSession instrumentHandle,
                                       ViUInt16 firstChannel,
                                       ViUInt16 lastChannel,
                                       void change_ofStateMaskArray);
```

Description:

`ks387_writeChangeOfStateMask` sets the change of state mask for a block of channels. A bitmap is input, where bit 0 of the first byte of the pData array represents the first channel, bit 1 represents the second channel, etc. A '1' bit in the mask indicates that the corresponding channel will participate in the change of state operation, whereas a '0' bit in the mask indicates that the channel will not participate (input state is 'don't care').

Parameters:

instrumentHandle

Variable Type ViSession

Instrument Handle

firstChannel

Variable Type ViUInt16

The first channel in a range to specify the change of state mask.

lastChannel

Variable Type ViUInt16

The last channel in a range to specify the change of state mask.

change_ofStateMaskArray

Variable Type void (passed by reference)

A bitmap of mask bits. Bit 0 of the first byte is the mask of the first channel in the block, the second bit is the mask the second channel will match, etc. A '1' bit in the mask indicates that the corresponding channel will participate in the pattern match operation, whereas a '0' bit in the mask indicates that the channel will not participate (input state is 'don't care').

Return Values:

Standard function status codes

ks387_startChangeOfState

Syntax:

```
ViStatus ks387_startChangeOfState (ViSession instrumentHandle);
```

Description:

`ks387_startChangeOfState` directs the 387 to begin scanning for a change in any input channel masked 'on' ('1') via `ks387_writeChangeOfStateMask`. The status of the scanning operation can be polled via `ks387_getChangeOfStateStatus` or `ks387_enableEvent` can be used to enable the 387 to generate a VISA event when the change of state occurs.

Parameters:

instrumentHandle

Variable Type ViSession

Instrument Handle

Return Values:

Standard function status codes. In addition to standard return codes, Returns `VI_ERROR_RSRC_BUSY` if change of state operation is already is progress.

ks387_stopChangeOfState

Syntax:

```
ViStatus ks387_stopChangeOfState (ViSession instrumentHandle);
```

Description:

`ks387_stopChangeOfState` stops the 387 to stop scanning for a change of state.

Parameters:

instrumentHandle

Variable Type ViSession

Instrument Handle

Return Values:

Standard function status codes. In addition to standard return codes, Returns `VI_ERROR_ABORT` if no change of state operation is in progress.

ks387_getChangeOfStateStatus

Syntax:

```
ViStatus ks387_getChangeOfStateStatus (ViSession instrumentHandle);
```

Description:

`ks387_getChangeOfStateStatus` returns the status of the state scanning operation initiated via `ks387_startChangeOfState`.

Parameters:

instrumentHandle

Variable Type ViSession

Instrument Handle

Return Values:

`VI_SUCCESS` is returned if the operation is complete and a change of state was detected. `VI_SUCCESS_QUEUE_EMPTY` is returned if this function has already returned `VI_SUCCESS`, but no start operation has been subsequently performed. `VI_SUCCESS_EVENT_EN` is returned if a change of state operation has started and is in progress. `VI_SUCCESS_EVENT_DIS` is returned if no change of state operation is in progress or successfully completed (i.e., none ever started or last one was stopped).

ks387_enableEvent

Syntax:

```
ViStatus ks387_enableEvent (ViSession instrumentHandle);
```

Description:

`ks387_enableEvent` allows the Completion of either COS or PAT to generate a VISA event. Note that events _must_ be enabled first in VISA via viEnableEvent() _before_ this function (`ks387_enableEvent`) is called. Enabling events via the Pnp driver before enabling in VISA may cause a system hang! (This is a PXI/cPCI issue, not a VISA implementation issue.)

Parameters:

instrumentHandle

Variable Type ViSession

Instrument Handle

Return Values:

Standard function status codes

ks387_disableEvent

Syntax:

```
ViStatus ks387_disableEvent (ViSession instrumentHandle);
```

Description:

`ks387_disableEvent` disables VISA event generation.

Parameters:

instrumentHandle

Variable Type ViSession

Instrument Handle

Return Values:

Standard function status codes

ks387_enableEventStatus

Syntax:

```
ViStatus ks387_enableEventStatus (ViSession instrumentHandle,
                                  ViPBoolean enableStatus);
```

Description:

`ks387_enableEventStatus` determines if events are enabled or not.

Parameters:

instrumentHandle

Variable Type ViSession

Instrument Handle

enableStatus

Variable Type ViBoolean (passed by reference)

Variable to be populated `VI_TRUE or VI_FALSE`, depending on if VISA events are enabled.

Return Values:

Standard function status codes

ks387_setOutputStrobeDelay

Syntax:

```
ViStatus ks387_setOutputStrobeDelay (ViSession instrumentHandle,
                                     ViUInt16 firstChannel,
                                     ViUInt16 lastChannel,
                                     ViUInt32 timeRequested,
                                     ViPUInt32 actualTime);
```

Description:

`ks387_setOutputStrobeDelay` sets the time an output strobe is delayed after changing an output channel.

Parameters:

instrumentHandle

Variable Type ViSession

Instrument Handle

firstChannel

Variable Type ViUInt16

The first channel in a range to specify the output strobe delay.

lastChannel

Variable Type ViUInt16

The last channel in a range to specify the output strobe delay.

timeRequested

Variable Type ViUInt32

The delay time in nanoseconds. Since hardware may not be able to support time requested, time actually set is returned (if pDelayActual is not NULL).

actualTime

Variable Type ViUInt32 (passed by reference)

Supported time value hardware actually set to. This pointer can be NULL, in which case no value is returned.

Return Values:

Standard function status codes. In addition to standard return codes, `VI_ERROR_CHAN_ALIGN` is returned if range of input channels violates the channel grouping restrictions.

ks387_getOutputStrobeDelay

Syntax:

```
ViStatus ks387_getOutputStrobeDelay (ViSession instrumentHandle,
                                     ViUInt16 firstChannel,
                                     ViUInt16 lastChannel,
                                     ViPUInt32 actualTime);\
```

Description:

`ks387_getOutputStrobeDelay` gets the time an output strobe is delayed after changing an output channel. `ks387_getOutputStrobeDelay` retrieves the delay times for a block of channels. Time is in nanoseconds. Each channel in requested range populates an element in `pDelayTime` array. `pDelayTime` must be sized to accommodate the number of channels.

Parameters:

instrumentHandle

Variable Type ViSession

Instrument Handle

firstChannel

Variable Type ViUInt16

The first channel in a range to get the output strobe delay.

lastChannel

Variable Type ViUInt16

The last channel in a range to get the output strobe delay.

actualTime

Variable Type ViUInt32 (passed by reference)

Supported time value hardware actually set to. This pointer can be NULL, in which case no value is returned.

Return Values:

Standard function status codes.

ks387_setStrobeEnable

Syntax:

```
ViStatus ks387_setStrobeEnable (ViSession instrumentHandle,
                                ViUInt16 firstChannel,
                                ViUInt16 lastChannel,
                                ViUInt16 strobeChannel);
```

Description:

ks387_setStrobeEnable assigns a strobe channel to a block of channels and enables strobing OR disables strobing. All other attributes of the strobe (polarity, direction, etc) must be configured before enabling. To enable the strobe and then manipulate any other configuration settings will produce unpredictable results.

Parameters:

instrumentHandle

Variable Type ViSession

Instrument Handle

firstChannel

Variable Type ViUInt16

The first channel in a range to specify the strobe channel.

lastChannel

Variable Type ViUInt16

The last channel in a range to specify the strobe channel.

strobeChannel

Variable Type ViUInt16

The channel channel. It must be less than or equal to KS387STROBEMAXCHAN. To disable strobing for a block of channels, specify KS387NOSTROBE.

Return Values:

Standard function status codes. In addition to standard return codes, VI_ERROR_CHAN_ALIGN is returned if range of input channels violates the channel grouping restrictions. VI_ERROR_INV_SETUP is returned if either a pattern recognition or change of state operation is in progress.

ks387_getStrobeEnable

Syntax:

```
ViStatus ks387_getStrobeEnable (ViSession instrumentHandle,
                                ViUInt16 firstChannel,
                                ViUInt16 lastChannel,
                                ViPUInt16 strobeChannelArray);
```

Description:

`ks387_getStrobeEnable` returns an array of strobe channels, 1 per channel in the input range.

Parameters:

instrumentHandle

Variable Type ViSession

Instrument Handle

firstChannel

Variable Type ViUInt16

The first channel in a range to get the strobe channel.

lastChannel

Variable Type ViUInt16

The last channel in a range to get the strobe channel.

strobeChannelArray

Variable Type ViUInt16 (passed by reference)

A user supplied array to be populated with the set of strobes associated with the channel input request. The array must be sized to accommodate 1 entry per channel in input range. The value KS387NOSTROBE is returned for any channel which does not have strobing enabled.

Return Values:

Standard function status codes.

ks387_setStrobePolarity

Syntax:

```
ViStatus ks387_setStrobePolarity (ViSession instrumentHandle,
                                  ViUInt16 firstChannel,
                                  ViUInt16 lastChannel,
                                  ViUInt16 strobePolarity);
```

Description:

`ks387_setStrobePolarity` assigns a strobe polarity to a block of channels. Polarity can be either a high pulse or a low pulse.

Parameters:

instrumentHandle

Variable Type ViSession

Instrument Handle

firstChannel

Variable Type ViUInt16

The first channel in a range to specify the strobe polarity.

lastChannel

Variable Type ViUInt16

The last channel in a range to specify the strobe polarity.

strobePolarity

Variable Type ViUInt16

Strobe polarity. Set to KS387STROBEPULSEHIGH for a normally low signal which pulses high, or KS387STROBEPULSELOW for a normally high signal which pulses low.

Return Values:

Standard function status codes. In addition to standard return codes, `VI_ERROR_CHAN_ALIGN` is returned if range of input channels violates the channel grouping restrictions.

ks387_getStrobePolarity

Syntax:

```
ViStatus ks387_getStrobePolarity (ViSession instrumentHandle,
                                  ViUInt16 firstChannel,
                                  ViUInt16 lastChannel,
                                  ViPUInt16 polarityArray);
```

Description:

`ks387_getStrobePolarity` returns an array of strobe polarity values, 1 per channel in the input range.

Parameters:

instrumentHandle

Variable Type ViSession

Instrument Handle

firstChannel

Variable Type ViUInt16

The first channel in a range to get the strobe polarity.

lastChannel

Variable Type ViUInt16

The last channel in a range to get the strobe polarity.

polarityArray

Variable Type ViUInt16 (passed by reference)

A user supplied array to be populated with the set of polarities associated with the channel input request. The array must be sized to accommodate 1 entry per channel in input range. The value KS387STROBEPULSEHIGH is returned for a low to high pulse, and KS387STROBEPULSELOW for a high to low pulse.

Return Values:

Standard function status codes.

ks387_setStrobeDirection

Syntax:

```
ViStatus ks387_setStrobeDirection (ViSession instrumentHandle,
                                   ViUInt16 firstChannel,
                                   ViUInt16 lastChannel,
                                   ViUInt16 direction);
```

Description:

`ks387_setStrobeDirection` assigns a strobe direction to a block of channels. Direction can be either KS387IODIRINPUT or KS387IODIROUTPUT. This function is only used for mezzanine channels; the strobe direction of base channels it specified by the `ks387_setIoDirection` function.

Parameters:

instrumentHandle

Variable Type ViSession

Instrument Handle

firstChannel

Variable Type ViUInt16

The first channel in a range to specify the strobe direction.

lastChannel

Variable Type ViUInt16

The last channel in a range to specify the strobe direction.

direction

Variable Type ViUInt16

Set to either KS387IODIRINPUT to set the block of channels to strobe on input or KS387IODIROUTPUT to set the block of channels to strobe on output.

Return Values:

Standard function status codes. In addition to standard return codes, `VI_ERROR_CHAN_ALIGN` is returned if range of input channels violates the channel grouping restrictions. `VI_ERROR_NSUP_OPER` is returned if the input range includes channels that are not mezzanine channels.

ks387_getStrobeDirection

Syntax:

```
ViStatus ks387_getStrobeDirection (ViSession instrumentHandle,
                                   ViUInt16 firstChannel,
                                   ViUInt16 lastChannel,
                                   void *bitmap_pointer);
```

Description:

`ks387_getStrobeDirection` retrieves the strobe direction settings of a block of channels. The result is returned as a bitmap, where bit 0 of the first byte of the pDir array represents the first channel, bit 1 represents the second channel, etc. Each bit is set according to the binary equivalent of KS387IODIRINPUT or KS387IODIROUTPUT

Parameters:

instrumentHandle

Variable Type ViSession

Instrument Handle

firstChannel

Variable Type ViUInt16

The first channel in a range to get the strobe direction of.

lastChannel

Variable Type ViUInt16

The last channel in a range to get the strobe direction of.

bitmap_pointer

Variable Type void (passed by reference)

Pointer to a user-allocated array. The array must be sized appropriately to accommodate 1 bit per channel requested.

Return Values:

Standard function status codes. In addition to standard return codes, `VI_ERROR_NSUP_OPER` is returned if the input range includes channels that are not mezzanine channels.

ks387_readROMConfig

Syntax:

```
ViStatus ks387_readROMConfig (ViSession instrumentHandle,
                              ViUInt16 site,
                              ViUInt16 offset,
                              ViUInt16 numberBytes,
                              void *buffer);
```

Description:

Given a site and offset, `ks387_readROMConfig` reads numBytes of configuration data. This is a convenience function to read mezzanine ROM information without a session to a mezzanine card.

Parameters:

instrumentHandle

Variable Type ViSession

Instrument Handle

site

Variable Type ViUInt16

The mezzanine site for which configuration data is to be read. Site should be in the range of 1 to N, where N is the total number of sites supported (see ks387_getNumberSites to determine the number of sites supported).

offset

Variable Type ViUInt16

Offset from the start of configuration space, in bytes.

numberBytes

Variable Type ViUInt16

Number of bytes to read from configuration space.

buffer

Variable Type void (passed by reference)

User-allocated array into which confuration data will be transfered.

Return Values:

Standard function status codes. In addition to standard return codes, VI_ERROR_NSUP_OPER is returned if the base board (site == 0) is specified, VI_ERROR_UNOCCUPIED_SITE if an unoccupied site is specified, or VI_ERRROR_INV_SITE if an invalid site is specified that are not mezzanine channels.

ks387_createIPSession

Syntax:

```
ViStatus ks387_createIPSession (ViSession instrumentHandle,
                                ViUInt16 site,
                                ViPSession session_to_site);
```

Description:

ks387_createIPSession returns a VISA session opened directly to the specified mezzanine site. The session can be passed into any of the instrument driver driver functions for that specific mezzanine card type. This function is provided because VISA cannot directly address mezzanine cards on the 387. The session created is not a general VISA session, and should only be used to call mezzanine card instrument driver functions. When no longer needed, it should be passed into viClose().

Parameters:

instrumentHandle

Variable Type ViSession

Instrument Handle

site

Variable Type ViUInt16

The mezzanine site for which a session should be created. Site should be in the range of 1 to N, where N is the total number of sites supported (see ks387_getNumberSites to determine the number of sites supported).

session_to_site

Variable Type ViSession (passed by reference)

Returned session.

Return Values:

Standard function status codes. In addition to standard return codes, VI_ERROR_NSUP_OPER is returned if the base board (site = 0) is specified, VI_ERROR_UNOCCUPIED_SITE if an unoccupied site is specified, or VI_ERRROR_INV_SITE if an invalid site is specified that are not mezzanine channels.

Data Functions

ks387_readIoData

Syntax:

```
ViStatus ks387_readIoData (ViSession instrumentHandle,
                           ViUInt16 firstChannel,
                           ViUInt16 lastChannel,
                           void *bitmap_pointer);
```

Description:

`ks387_readIoData` reads data for a block of channels. The result is returned as a bitmap, where bit 0 of the first byte of the pData array represents the first channel, bit 1 represents the second channel, etc. Each bit represents the state of the channel. For output channels, the value read is the last value that was written out; for input channels, the value is the value sensed at the input.

Parameters:

instrumentHandle

Variable Type ViSession

Instrument Handle

firstChannel

Variable Type ViUInt16

The first channel in a range to read data from.

lastChannel

Variable Type ViUInt16

The last channel in a range to read data from.

bitmap_pointer

Variable Type void (passed by reference)

Pointer to a user-allocated array. The array must be sized appropriately to accommodate 1 bit per channel requested.

Return Values:

Standard function status codes

ks387_writeIoData

Syntax:

```
ViStatus ks387_writeIoData (ViSession instrumentHandle,
                            ViUInt16 firstChannel,
                            ViUInt16 lastChannel,
                            void *outputLevelArray);
```

Description:

`ks387_writeIoData` sets the levels of a block of channels. There is no effect on any channels in the range which are configured as inputs.

Parameters:

instrumentHandle

Variable Type ViSession

Instrument Handle

firstChannel

Variable Type ViUInt16

The first channel in a range to write output levels to.

lastChannel

Variable Type ViUInt16

The last channel in a range to write output levels to.

outputLevelArray

Variable Type void (passed by reference)

A bitmap of level bits. Bit 0 of the first byte is the level of the first channel in the block, the second bit is the level the second channel will match, etc.

Return Values:

Standard function status codes.

Appendix A

Technical Support and Warranty

KineticSystems warrants its standard hardware products to be free of defects in workmanship and materials for a period of one year from the date of shipment to the original end user. KineticSystems warrants its software products to conform to the software description applicable at the time of purchase for a period of ninety days from the date of shipment. Products purchased for resale by KineticSystems carry the original equipment manufacturer's warranty.

KineticSystems will, at its option, either repair or replace products that prove to be defective in materials or workmanship during the warranty period.

Transportation charges for shipping products to KineticSystems are prepaid by the purchaser, while charges for returning the repaired product to the purchaser, if located in the United States, are paid by KineticSystems. Return shipments are made by UPS, where available, unless the purchaser requests a premium method of shipment at his expense. The selected carrier is not the agent of KineticSystems, and KineticSystems assumes no liability relating to the services provided by the carrier.

The product warranty may vary outside the United States and does not include shipping, customs clearance or any other charges. Consult your local authorized representative for more information regarding specific warranty coverage and shipping details.

Product specifications and descriptions in this document subject to change without notice.
KineticSystems specifically makes no warranty of fitness for a particular purpose or any other warranty either expressed or implied, except as is expressly set forth herein. This warranty does not cover product failures created by unauthorized modifications, product misuse or improper installation.

Products are not accepted for credit or exchange without prior written approval. If it is necessary to return a product for repair replacement or exchange, a Return Authorization (RA) Number must first be obtained from the Repair Service Center before shipping the product to KineticSystems.

Please take the following steps if you are having a problem and feel you may need to return a product for service:

Contact KineticSystems and discuss the problem with a Technical Service Engineer.
Obtain a Return Authorization (RA) Number.
Initiate a purchase order for the estimated repair charge if the product is out of warranty.
Include with the product a description of the problem and the name of the technical contact person at your facility.
Ship the product prepaid with the RA Number marked on the outside of the package to:

KineticSystems Company, LLC
Repair Service Center
900 North State Street
Lockport, IL 60441

Telephone: (815) 838-0005
Fax: (815) 838-4424

Ways to contact us:

KineticSystems Company, LLC
900 N. State Street
Lockport, IL 60441-2200

Phone: (800) DATA NOW (1-800-328-2669)
(815) 838-0005
Fax: (815) 838-4424

E-mail: mkt-info@kscorp.com
tech-serv@kscorp.com
sales@kscorp.com
Web: http://www.kscorp.com

Feedback

The purpose of this manual is to provide you with the information you need to make the CP387 as easy as possible to understand and use. It is very important that the information is accurate, understandable and accessible. To help us continue to make this manual as "user friendly" as possible, we hope you will fill out this form and Fax it back to us at (815) 838 0095. Or mail a copy to KineticSystems Company, LLC 900 N. State, Lockport, IL 60441. Your input is very valuable.

Please rate each of the following.

The information in this manual is:

	Yes								No
Accurate	10	9	8	7	6	5	4	3	2 1
Readable	10	9	8	7	6	5	4	3	2 1
Easy to find	10	9	8	7	6	5	4	3	2 1
Well organized	10	9	8	7	6	5	4	3	2 1
Sufficient	10	9	8	7	6	5	4	3	2 1

We would appreciate receiving any thoughts you have about how we can improve this user's manual:

(Include additional sheets if needed)
Name Phone
Company