PCIS-VEE/NT/98/2000

User Objects for HP VEEâ Windows NT/98/2000

Function Reference

Manual Rev. 3.00 : 10, June 2000

CONTENTS

FUNCTION REFERENCE 1

1.1 AI_9111_CONFIG 2
1.2 AI_9112_CONFIG 3
1.3 AI_9113_CONFIG 4
1.4 AI_9114_CONFIG 4
1.5 AI_9118_CONFIG 5
1.6 AI_9812_CONFIG 6
1.7 AI_CONTREADCHANNEL 8
1.8 AI_CONTSCANCHANNELS 14
1.9 AI_READCHANNEL 23
1.10 AI_VREADCHANNEL 24
1.11 AI_VSCALE 24
1.12 AO_6208A_CONFIG 25
1.13 AO_6308A_CONFIG 26
1.14 AO_6308V_CONFIG 26
1.15 AO_9111_CONFIG 27
1.16 AO_9112_CONFIG 28
1.17 AO_VSCALE 29
1.18 AO_VWRITECHANNEL 30
1.19 AO_WRITECHANNEL 30
1.20 CTR_8554_CK1_CONFIG 31
1.21 CTR_8554_CLKSRC_CONFIG 32

1.22 CTR_8554_DEBOUNCE_CONFIG 32
1.23 CTR_READ 33
1.24 CTR_RESET 34
1.25 CTR_SETUP 34
1.26 DI_7200_CONFIG 38
1.27 DI_7300B_CONFIG 39
1.28 DI_CONTREADPORT 40
1.29 DI_READLINE 41
1.30 DI_READPORT 44
1.31 DIO_PORTCONFIG 45
1.32 DO_7200_CONFIG 46
1.33 DO_7300B_CONFIG 47
1.34 DO_CONTWRITEPORT 48
1.35 DO_READLINE 50
1.36 DO_READPORT 51
1.37 DO_WRITELINE 52
1.38 DO_WRITEPORT 53
1.39 GETSAMPLE 55
1.40 REGISTER_CARD 55
1.41 RELEASE_CARD 57

APPENDIX A ERROR CODES 59

APPENDIX B AI RANGE CODES 61

1

Function Reference

The functionality of PCIS-VEE/NT/98/2000 user objects can be classified to the following capabilities:

1. Card Configuration:

- Initialization & Release: Initialize and release the hardware.
- Operation configuration:
  * Setting clock source, trigger mode, etc before continuous AI/DI/DO operation
  * Setting *Voltage to Current* mode of PCI-6208A/6308A card
  * Setting the direction (Input or output) configuration of the selected port for PCI-7248/7296/7396, cPCI-7249.

2. Analog Input:

- Perform one-shot single-channel analog input operation
- Perform continuous single/multiple-channel analog input operation

3. Analog Output:

- Performs single-channel analog output operation

4. Digital I/O:

- Input/output digital signals
- Perform continuous digital I/O operation on PCI-7200/7300B

5. Timer/Counter:

- timer/counter functions

Appendix "*Function Support*" shows which NuDAQ PCI-bus card each PCIS-VEE/NT/98/2000 user objects supports.

In addition, several sample programs are also included. Thorough understanding of these sample programs will help you understand how to use the library more quickly. And you are welcome to modify the sample programs for your application needs.

1.1 AI_9111_Config

@ Description

Set the trigger mode and trigger source for the PCI-9111 card with card ID *CardNumber*. You must call this function before calling function to perform continuous analog input operation.

@ Cards Support

9111

@ Input Parameter

Int32 **CardNumber** : The card id of the card that want to perform this operation.

Int32 **TrigSource** : The continuous A/D conversion trigger source.
1: on-board Programmable pacer
2: external signal trigger

Int32 **PreTrgEn** : Enable or Disable Pre-Trigger mode.
1: Enable Pre-Trigger mode
0: Disable Pre-Trigger mode

Int32 **TraceCnt** : The number of data will be accessed after a specific trigger event.

Int32 **EDO_Fun** : The mode of EDO ports.
1: EDO channels are used as input channels
2: EDO channels are used as output channels
3: EDO channels are used as channel number output

@ Output Parametet

Int32 **RetValue**: the error status of this user object. Please refer to Appendix A "*Error Codes*"

1.2 AI_9112_Config

@ Description

Set the trigger source for the PCI-9112 with card ID *CardNumber*. You must call this function before calling function to perform continuous analog input operation.

@ Cards Support

9112

@ Input Parameter

Int32 **CardNumber** : The card id of the card that want to perform this operation.

Int32 **TrigSource** : The continuous A/D conversion trigger source.
1: on-board Programmable pacer
2: external signal trigger

@ Output Parametet

Int32 **RetValue**: the error status of this user object. Please refer to Appendix A "*Error Codes*"

1.3 AI_9113_Config

@ Description

Set the trigger source for the PCI-9113 with card ID *CardNumber*. You must call this function before calling function to perform continuous analog input operation.

@ Cards Support

9113

@ Input Parameter

Int32 **CardNumber** :The card id of the card that want to perform this operation.

Int32 **TrigSource** : The continuous A/D conversion trigger source.
1: on-board programmable pacer

@ Output Parametet

Int32 **RetValue**: the error status of this user object. Please refer to Appendix A "*Error Codes*"

1.4 AI_9114_Config

@ Description

Set the trigger source for the PCI-9112 with card ID *CardNumber*. You must call this function before calling function to perform continuous analog input operation.

@ Cards Support

9114

@ Input Parameter

Int32 **CardNumber** : The card id of the card that want to perform this operation.

Int32 **TrigSource** : The continuous A/D conversion trigger source.
1: on-board Programmable pacer

2: external signal trigger

@ Output Parametet

Int32 **RetValue**: the error status of this user object. Please refer to Appendix A "*Error Codes*"

1.5 AI_9118_Config

@ Description

Set the trigger source, trigger mode and trigger properties selected for the PCI-9118 with card ID *CardNumber*. You must call this function before calling function to perform continuous analog input operation.

@ Cards Support

9118

@ Input Parameter

Int32 **CardNumber** :The card id of the card that want to perform this operation.

Int32 **ModeCtrl** : The setting for A/D mode control. This argument is an integer expression formed from one or more of the following constants. There are four groups of constants:

(1) **A/D Polarity Control**
- **0:** BiPolar
- **1:** UniPolar

(2) **A/D Channel Input Mode**
- **0:** Single-ended
- **2:** Differential

(3) **External Gate Enable**
- **4:** 8254 counter is controlled by TGIN pin

(4) **External Trigger Enable**
- **8:** External Hardware Trigger Mode enabled

When two or more modes want to be set, the correspondent constants can be added.

Int32 **FunCtrl** : The setting for A/D Function. This argument is an integer expression formed from one or more of the manifest constants defined in VEE.H. There are four groups of constants:

(1) **Digital Trigger Polarity**
- **0:** Digital trigger negative active
- **16:** Digital trigger positive active

(2) **External Trigger Polarity**
- **0:** External trigger negative active
- **32:** External trigger positive active

(3) **Burst Mode Enable**
- **64:** Burst Mode is enabled

(4) **Burst Mode with Sample and Hold Mode Enable**
- **128:** Burst mode with sample and hold is enabled

(5) **Trigger Mode Enable**
- **256:** Post trigger mode is enabled
- **512:** About trigger mode or Pre-trigger mode is enabled

When two or more modes want to be set, the correspondent constants can be added.

Int32 **BurstCnt** :The burst number

Int32 **PostCnt** :The number of data will be accessed after a specific trigger event

@ Output Parametet

Int32 **RetValue**: the error status of this user object. Please refer to Appendix A "*Error Codes*"

1.6 AI_9812_Config

@ Description

Set the trigger source, trigger mode, and trigger properties selected for the PCI-9812 card with card ID *CardNumber*. You must call this function before calling function to perform analog input operation.

@ Cards Support

9812/10

@ Input Parameter

Int32 **CardNumber** : The card id of the card that want to perform this operation.

Int32 **TrgMode** : The setting for A/D trigger mode. The valid trigger modes are as follows:

- **0:** Software Trigger
- **1:** Post Trigger
- **2:** Pre-Triger Mode
- **3:** Delay Trigger
- **4:** Middle Triger

Int32 **TrgSrc** : The setting for A/D Trigger Source. The valid trigger sources are as follows:

- **0:** Channel 0
- **8:** Channel 1
- **16:** Channel 2
- **24:** Channel 3
- **32:** External Digital Trigger

Int32 **TrgPol** : The setting of Trigger polarity. The valid values are:

- **0:** Positive slope trigger
- **64:** Negative slope trigger

Int32 **ClkSel** : A/D clock source. This argument is an integer expression formed from addition of one or more of the following constants. There are two groups of constants:

(1) **A/D Clock Frequency**

- **128:** Freq. of A/D clock is higher than PCI clock freq.
- **0:** Freq. of A/D clock is lower than PCI clock freq.

(2) **The ADC clock source**

- **0:** Internal clock
- **256:** External sin wave clock
- **512:** External square wave clock

Two constants of different groups can be added to form the needed setting.

Int32 **TrgLevel** :The setting of Trigger level. The relationship between the value of *TrgLevel* and trigger voltage is listed in the following table:

TrgLevel	trigger voltage(±1V)	trigger voltage(±5V)
255	0.992V	4.96V
254	0.984V	4.92V
---	---	---
129	0.008V	0.04V
128	0.000V	0.00V
127	-0.008V	-0.04V
---	---	---
1	-0.992V	-4.96V
0	-1.000V	-5.00V

Int32 **PostCnt**: The post count value setting for Middle Trigger mode or Delay Trigger mode. This argument is expressed as:

Middle trigger mode: the number of data accessed for each selected channel after a specific trigger event

Delay trigger mode: the counter value for deferring to access data after a specific trigger event

@ Output Parametet

Int32 **RetValue**: the error status of this user object. Please refer to Appendix A "*Error Codes*"

1.7 AI_ContReadChannel

@ Description

This function performs continuous A/D conversions on the specified analog input channel at a rate as close to the rate you specified.

@ Cards Support

9111, 9112, 9113, 9114, 9118, 9812/10

@ Input Parameter

Int32 **CardNumber** : The card id of the card that want to perform this operation.

Int32 **Channel** : Analog input channel number

PCI-9111: 0 through 15
PCI-9112: 0 through 15
PCI-9113: 0 through 31
PCI-9114: 0 through 31
PCI-9118: 0 through 15
PCI-9812/10: 0

Int32 **AdRange** :The analog input range the specified channel is setting. Please refer to the Appendix B, ***AI Range Codes***, for the valid range values.

Int32 **Buffer** : An 32 bit integer array to contain the acquired data. The data format in *Buffer* is described as below:

PCI-9112

Because the *Buffer* is an 32 bit integer array, but **PCI-9112 's** data is on 16 bit integer base, so every item of the *Buffer* contain two 16 bit **PCI-9112 's** integer datas. *Buffer* must has a length equal to or greater than the half value of parameter *ReadCount*.

The upper 16-bit unsigned integer data is described as below:

B11 B10 B9 ..... B1 B0 A3 A2 A1 A0

where B11, B10, ... , B0 : A/D converted data
A3, A2, A1, A0 : converted channel no.

The lower 16-bit unsigned integer data is described as below:

D11 D10 D9 ..... D1 D0 C3 C2 C1 C0

where D11, D10, ... , D0 : A/D converted data
C3, C2, C1, C0 : converted channel no.

PCI-9111DG

Because the *Buffer* is an 32 bit integer array, but **PCI-9111DG 's** data is on 16 bit integer base, so every item of the *Buffer* contain two 16 bit **PCI-9111DG 's** integer datas. *Buffer* must has a length equal to or greater than the half value of parameter *ReadCount*.

The upper 16-bit unsigned integer data is described as below:

B11 B10 B9 ..... B1 B0 A3 A2 A1 A0

where B11, B10, ... , B0 : A/D converted data
A3, A2, A1, A0 : converted channel no.

The lower 16-bit unsigned integer data is described as below:

D11 D10 D9 ..... D1 D0 C3 C2 C1 C0

where D11, D10, ... , D0 : A/D converted data
C3, C2, C1, C0 : converted channel no.

PCI-9111HR

Because the *Buffer* is an 32 bit integer array, but **PCI-9111HR 's** data is on 16 bit integer base, so every item of the *Buffer* contain two 16 bit **PCI-9111HR 's** integer datas. *Buffer* must has a length equal to or greater than the half value of parameter *ReadCount*.

The upper 16-bit unsigned integer data is described as below:

A15 A14 A13 ..... A1 A0

where A15, A14, ... , A0 : A/D converted data

The lower 16-bit unsigned integer data is described as below:

B15 B14 B13 ..... B1 B0

where B15, B14, ... , B0 : A/D converted data

PCI-9113

Because the *Buffer* is an 32 bit integer array, but **PCI-9113 ‘s** data is on 32 bit integer base, so every item of the *Buffer* contain one *32* bit **PCI-9113 ‘s** integer data. *Buffer* must has a length equal to or greater than value of parameter *ReadCount*.

Every 32-bit unsigned integer data (including 12-bit unsigned A/D data):

B31… B21 C4 C3 C2 C1 C0 B15 … B12 D11 D10 ... D1 D0

where D11, D10, ... , D0 : A/D converted data

C3, C2, C1, C0 : converted channel no.

B31 ~ B21 & B15 ~ B12: don’t care

PCI-9114

Because the *Buffer* is an 32 bit integer array, but **PCI-9114 ‘s** data is on 32 bit integer base, so every item of the *Buffer* contain one *32* bit **PCI-9114 ‘s** integer data. *Buffer* must has a length equal to or greater than the value of parameter *ReadCount*.

Every 32-bit unsigned integer data (including 16-bit signed A/D data):

B31 … B21 C4 C3 C2 C1 C0 D15 D14 ... D1 D0

where D15, D14, ... , D0 : A/D converted data

C3, C2, C1, C0 : converted channel no.

B31 ~ B21: don’t care

PCI-9118HR

Because the *Buffer* is an 32 bit integer array, but **PCI-9118HR ‘s** data is on 16 bit integer base, so every item of the *Buffer* contain two 16 bit **PCI-9118HR ‘s** integer datas. *Buffer* must has a length equal to or greater than the half value of parameter *ReadCount*.

The upper 16-bit unsigned integer data is described as below:

A15 A14 A13 ..... A1 A0

where A15, A14, ... , A0 : A/D converted data

The lower 16-bit unsigned integer data is described as below:

D15 D14 D13 ..... D1 D0

where D15, D14, ... , D0 : A/D converted data

PCI-9118DG/HG

Because the *Buffer* is an 32 bit integer array, but **PCI-9118DG/HG 's** data is on 16 bit integer base, so every item of the *Buffer* contain two 16 bit **PCI-9118DG/HG 's** integer datas. *Buffer* must has a length equal to or greater than the half value of parameter *ReadCount*.

The upper 16-bit unsigned integer data is described as below:

B11 B10 B9 ..... B1 B0 A3 A2 A1 A0

where B11, B10, ... , B0 : A/D converted data
A3, A2, A1, A0 : converted channel no.

The lower 16-bit unsigned integer data is described as below:

D11 D10 D9 ..... D1 D0 C3 C2 C1 C0

where D11, D10, ... , D0 : A/D converted data
C3, C2, C1, C0 : converted channel no.

PCI-9812

Because the *Buffer* is an 32 bit integer array, but **PCI-9812 's** data is on 16 bit integer base, so every item of the *Buffer* contain two 16 bit **PCI-9812 's** integer datas. *Buffer* must has a length equal to or greater than the half value of parameter *ReadCount*.

The upper 16-bit unsigned integer data is described as below:

B11 B10 B9 ..... B1 B0 a3 a2 a1 a0

where B11, B10, ... , B0 : A/D converted data
a2, a1, a0 : Digital Input data.
a3: trigger detection flag

The lower 16-bit unsigned integer data is described as below:

D11 D10 D9 ..... D1 D0 c3 c2 c1 c0

where D11, D10, ... , D0 : A/D converted data
c2, c1, c0 : Digital Input data.
c3: trigger detection flag

PCI-9810

Because the *Buffer* is an 32 bit integer array, but **PCI-9810 's** data is on 16 bit integer base, so every item of the *Buffer* contain two 16 bit **PCI-9810 's** integer datas. *Buffer* must has a length equal to or greater than the half value of parameter *ReadCount*.

The upper 16-bit unsigned integer data is described as below:

B9 B8 B7 ..... B1 B0 a5 a4 a3 a2 a1 a0

where B9, B8, ... , B0 : A/D converted data
a4, a3, a2, a1, a0 : Digital Input data.
A5: trigger detection flag

The lower 16-bit unsigned integer data is described as below:

D9 D8 D7 ..... D1 D0 b5 b4 b3 b2 b1 b0

where D9, D8, ... , D0 : A/D converted data
b4, b3, b2, b1, b0 : Digital Input data.
B5: trigger detection flag

Int32 **ReadCount** : the number of A/D conversions to be performed.

Real **SampleRate** :The sampling rate you want for analog input in hertz (samples per second). Your maximum rate depends on the card type and your computer system. If you set A/D trigger mode as external trigger by calling **AI_9111_Config, AI_9112_Config, AI_9113_Config,AI_9114_Config, AI_9812_Config or AI_9118_Config**, the sampling rate is determined by an external

trigger source, you have to set this argument as 10000.

Int32 **TransferCount** :TransferCount is the actual number of A/D conversions performed.

@ Output Parametet

Int32 **RetValue**: the error status of this user object. Please refer to Appendix A "*Error Codes*".

Int32 **Buffer** : The return value of *Buffer* integer array

Int32 **TransferCount** : the actual number of A/D conversions to be performed.

1.8 AI_ContScanChannels

@ Description

This function performs continuous A/D conversions on the specified continuous analog input channels at a rate as close to the rate you specified. This function takes advantage of the hardware auto-scan functionality to perform multi-channel analog input.

@ Cards Support

9111, 9112, 9113, 9114, 9118, 9812/10

@ Input Parameter

Int32 **CardNumber** : The card ID of the card that want to perform this operation.

Int32 **Channel** : The largest channel number of specified continuous analog input channel. The channel order for acquiring data is as follows:

PCI-9111: number of *Channel* must be within 0 and 15. The continuous scan sequence is ascending and the first one must be zero. For example, 0, 1, 2, 3.

PCI-9112: number of *Channel* must be within 0 and 15. The continuous scan

sequence is descending, and the first one must be zero. For example, 3, 2, 1, 0.

PCI-9113: number of *Channel* must be within 0 and 31. The continuous scan sequence is ascending and the first one must be zero. For example, 0, 1, 2, 3.

PCI-9114: number of *Channel* must be within 0 and 31. The continuous scan sequence is ascending and the first one must be zero. For example, 0, 1, 2, 3.

PCI-9118: number of *Channel* must be within 0 and 15. The continuous scan sequence is ascending and the first one must be zero. For example, 0, 1, 2, 3.

PCI-9812/10: number of *Channel* must be 0, 1 or 3. The continuous scan sequence is ascending and the first one must be zero. For example, 0, 1, 2, 3.

Int32 **AdRange** : The analog input range the continuous specified channel is setting. Please refer to the Appendix B for the valid range values.

Int32 **Buffer** : An 32 bit integer array to contain the acquired data. The data format in *Buffer* is described as below:

PCI-9111DG

The length of *Buffer* must be equal to or greater than the half value of parameter *ReadCount*. The acquired data is stored in interleaved sequence. For example, if the value of *Channel* is 3, and the scanned channel numbers is descending (e.g. **PCI-9111DG**), then this function input data from channel 2, then channel 1, then channel 0, then channel 2, then channel 1, ... The data acquired is put to *Buffer* by order. Because

the *Buffer* is an 32 bit integer array, but **PCI-9111DG 's** data is on 16 bit integer base, so every item of the *Buffer* contain two 16 bit integer data. So the data read from channel 2 is stored in the upper part of *Buffer*[0], The data from channel 1 is stored in the lower part of *Buffer*[0], The data from channel 0 is stored in the upper part of *Buffer*[1], the next data read from channel 2 is stored in the lower part of *Buffer*[1], The next data from channel 1 is stored in the upper part of *Buffer*[2], The data from channel 0 is stored in the lower part of *Buffer*[2], ... If double-buffered mode is enabled, this buffer is of no use, you can ignore this argument.
Every 16-bit signed integer data:

D11 D10 D9 ..... D1 D0 C3 C2 C1 C0

where D11, D10, ... , D0 : A/D converted data
C3, C2, C1, C0 : converted channel no.

PCI-9111HR

The length of *Buffer* must be equal to or greater than the half value of parameter *ReadCount*. The acquired data is stored in interleaved sequence. For example, if the value of *Channel* is 3, and the scanned channel numbers is descending (e.g. **PCI-9111HR**), then this function input data from channel 2, then channel 1, then channel 0, then channel 2, then channel 1, ... The data acquired is put to *Buffer* by order. Because the *Buffer* is an 32 bit integer array, but **PCI-9111HR 's** data is on 16 bit integer base, so every item of the *Buffer* contain two 16 bit integer data. So the data read from channel 2 is stored in the upper part of *Buffer*[0], The data from channel 1 is stored in the lower part of *Buffer*[0], The data from channel 0 is stored in the upper part of *Buffer*[1], the next data read from channel 2 is stored in the lower part

of *Buffer*[1], The next data from channel 1 is stored in the upperpart of *Buffer*[2], The data from channel 0 is stored in the lower part of *Buffer*[2], ... If double-buffered mode is enabled, this buffer is of no use, you can ignore this argument.
Every 16-bit signed integer data:

D15 D14 D13 ..... D1 D0

where D15, D14, ... , D0 : A/D converted data

PCI-9112

The length of *Buffer* must be equal to or greater than the half value of parameter *ReadCount*. The acquired data is stored in interleaved sequence. For example, if the value of *Channel* is 3, and the scanned channel numbers is descending (e.g. **PCI-9112**), then this function input data from channel 2, then channel 1, then channel 0, then channel 2, then channel 1, ... The data acquired is put to *Buffer* by order. Because the *Buffer* is an 32 bit integer array, but **PCI-9112 's** data is on 16 bit integer base, so every item of the *Buffer* contain two 16 bit integer data. So the data read from channel 2 is stored in the upper part of *Buffer*[0], The data from channel 1 is stored in the lower part of *Buffer*[0], The data from channel 0 is stored in the upper part of *Buffer*[1], the next data read from channel 2 is stored in the lower part of *Buffer*[1], The next data from channel 1 is stored in the upper part of *Buffer*[2], The data from channel 0 is stored in the lower part of *Buffer*[2], ... If double-buffered mode is enabled, this buffer is of no use, you can ignore this argument.
Every 16-bit unsigned integer data:

D11 D10 D9 ..... D1 D0 C3 C2 C1 C0

where D11, D10, ... , D0 : A/D converted data

C3, C2, C1, C0 : converted channel no.

PCI-9113

The length of *Buffer* must be equal to or greater than the value of parameter *ReadCount*. The acquired data is stored in interleaved sequence. For example, if the value of *Channel* is 3, and the scanned channel numbers is descending (e.g. **PCI-9113**), then this function input data from channel 2, then channel 1, then channel 0, then channel 2, then channel 1, ... The data acquired is put to *Buffer* by order. Because the *Buffer* is an 32 bit integer array, and **PCI-9113 's** data is also on 32 bit integer base, so every item of the *Buffer* contain one *32* bit integer data. So the data read from channel 2 is stored in *Buffer*[0], The data from channel 1 is stored in *Buffer*[1], The data from channel 0 is stored in *Buffer*[2], the next data read from channel 2 is stored in *Buffer*[3], The next data from channel 1 is stored in *Buffer*[4], The data from channel 0 is stored in *Buffer*[5], ... If double-buffered mode is enabled, this buffer is of no use, you can ignore this argument.
Every 32-bit unsigned integer data (including 12-bit unsigned A/D data):

B31 … B21 C4 C3 C2 C1 C0 B15 … B12 D11 D10 … D1 D0

where D11, D10, ... , D0 : A/D converted data
C3, C2, C1, C0 : converted channel no.
B31 ~ B21 & B15 ~ B12: don't care

PCI-9114

The length of *Buffer* must be equal to or greater than the value of parameter *ReadCount*. The acquired data is stored in interleaved sequence. For example, if the value of *Channel* is 3, and the scanned channel numbers is descending (e.g. **PCI-9114**), then this function input data from

channel 2, then channel 1, then channel 0, then channel 2, then channel 1, ... The data acquired is put to *Buffer* by order. Because the *Buffer* is an 32 bit integer array, and **PCI-9114 's** data is also on 32 bit integer base, so every item of the *Buffer* contain one *32* bit integer data. So the data read from channel 2 is stored in *Buffer*[0], The data from channel 1 is stored in *Buffer*[1], The data from channel 0 is stored in *Buffer*[2], the next data read from channel 2 is stored in *Buffer*[3], The next data from channel 1 is stored in *Buffer*[4], The data from channel 0 is stored in *Buffer*[5], ... If double-buffered mode is enabled, this buffer is of no use, you can ignore this argument.
Every 32-bit unsigned integer data (including 16-bit signed A/D data):

B31 … B21 C4 C3 C2 C1 C0 D15 D14 ... D1 D0

where D15, D14, ... , D0 : A/D converted data
C3, C2, C1, C0 : converted channel no.
B31 ~ B21: don't care

PCI-9118HR

The length of *Buffer* must be equal to or greater than the half value of parameter *ReadCount*. The acquired data is stored in interleaved sequence. For example, if the value of *Channel* is 3, and the scanned channel numbers is descending (e.g. **PCI-9118HR**), then this function input data from channel 2, then channel 1, then channel 0, then channel 2, then channel 1, ... The data acquired is put to *Buffer* by order. Because the *Buffer* is an 32 bit integer array, but **PCI-9118HR 's** data is on 16 bit integer base, so every item of the *Buffer* contain two 16 bit integer data. So the data read from channel 2 is stored in the upper part of *Buffer*[0], The data from channel 1 is stored in the lower part of *Buffer*[0], The data from channel 0 is stored

in the upper part of *Buffer*[1], the next data read from channel 2 is stored in the lower part of *Buffer*[1], The next data from channel 1 is stored in the upper part of *Buffer*[2], The data from channel 0 is stored in the lower part of *Buffer*[2], ... If double-buffered mode is enabled, this buffer is of no use, you can ignore this argument.
Every 16-bit signed integer data:

D15 D14 D13 ..... D1 D0

where D15, D14, ... , D0 : A/D converted data

PCI-9118DG/HG

The length of *Buffer* must be equal to or greater than the half value of parameter *ReadCount*. The acquired data is stored in interleaved sequence. For example, if the value of *Channel* is 3, and the scanned channel numbers is descending (e.g. **PCI-9118DG/HG**), then this function input data from channel 2, then channel 1, then channel 0, then channel 2, then channel 1, ... The data acquired is put to *Buffer* by order. Because the *Buffer* is an 32 bit integer array, but **PCI-9118DG/HG 's** data is on 16 bit integer base, so every item of the *Buffer* contain two 16 bit integer data. So the data read from channel 2 is stored in the upper part of *Buffer*[0], The data from channel 1 is stored in the lower part of *Buffer*[0], The data from channel 0 is stored in the upper part of *Buffer*[1], the next data read from channel 2 is stored in the lower part of *Buffer*[1], The next data from channel 1 is stored in the upper part of *Buffer*[2], The data from channel 0 is stored in the lower part of *Buffer*[2], ... If double-buffered mode is enabled, this buffer is of no use, you can ignore this argument.
Every 16-bit unsigned integer data:

D11 D10 D9 ..... D1 D0 C3 C2 C1 C0

where D11, D10, ... , D0 : A/D converted data
C3, C2, C1, C0 : converted channel no.

PCI-9812

The length of *Buffer* must be equal to or greater than the half value of parameter *ReadCount*. The acquired data is stored in interleaved sequence. For example, if the value of *Channel* is 3, and the scanned channel numbers is descending (e.g. **PCI-9812**), then this function input data from channel 2, then channel 1, then channel 0, then channel 2, then channel 1, ... The data acquired is put to *Buffer* by order. Because the *Buffer* is an 32 bit integer array, but **PCI-9812 's** data is on 16 bit integer base, so every item of the *Buffer* contain two 16 bit integer data. So the data read from channel 2 is stored in the upper part of *Buffer*[0], The data from channel 1 is stored in the lower part of *Buffer*[0], The data from channel 0 is stored in the upper part of *Buffer*[1], the next data read from channel 2 is stored in the lower part of *Buffer*[1], The next data from channel 1 is stored in the upper part of *Buffer*[2], The data from channel 0 is stored in the lower part of *Buffer*[2], ... If double-buffered mode is enabled, this buffer is of no use, you can ignore this argument.
Every 16-bit signed integer data:

D11 D10 D9 ..... D1 D0 b3 b2 b1 b0

where D11, D10, ... , D0 : A/D converted data
b2, b1, b0 : Digital Input data.
b3: trigger detection flag

PCI-9810

The length of *Buffer* must be equal to or greater than the half value of parameter *ReadCount*. The acquired data is stored in

interleaved sequence. For example, if the value of *Channel* is 3, and the scanned channel numbers is descending (e.g. **PCI-9810**), then this function input data from channel 2, then channel 1, then channel 0, then channel 2, then channel 1, ... The data acquired is put to *Buffer* by order. Because the *Buffer* is an 32 bit integer array, but **PCI-9810 's** data is on 16 bit integer base, so every item of the *Buffer* contain two 16 bit integer data. So the data read from channel 2 is stored in the upper part of *Buffer*[0], The data from channel 1 is stored in the lower part of *Buffer*[0], The data from channel 0 is stored in the upper part of *Buffer*[1], the next data read from channel 2 is stored in the lower part of *Buffer*[1], The next data from channel 1 is stored in the upper part of *Buffer*[2], The data from channel 0 is stored in the lower part of *Buffer*[2], ... If double-buffered mode is enabled, this buffer is of no use, you can ignore this argument.

Every 16-bit signed integer data:

D9 D8 D7 ..... D1 D0 b5 b4 b3 b2 b1 b0

where D9, D8, ... , D0 : A/D converted data
b2, b1, b0 : Digital Input data.
b3: trigger detection flag

Int32 **ReadCount** : the number of A/D conversions to be performed.

Real **SampleRate** : The sampling rate you want for analog input in hertz (samples per second). The maximum rate depends on the card type and your computer system. If you set A/D trigger source as external trigger by calling **AI_9111_Config, AI_9112_Config, AI_9113_Config,AI_9114_Config, AI_9812_Config or AI_9118_Config**, the sampling rate is

determined by an external trigger source, you have to set this argument as 10000.

Int32 **TransferCount** : the actual number of A/D conversions to be performed.

@ Output Parametet

Int32 **RetValue**: the error status of this user object. Please refer to Appendix A "*Error Codes*".

Int32 **TransferCount** :the actual number of A/D conversions performed.

Int32 **Buffer** : The return value of *Buffer* integer array

1.9 AI_ReadChannel

@ Description

This function performs a software triggered A/D conversion (analog input) on an analog input channel and returns the value converted.

@ Cards Support

9111, 9112, 9113, 9114, 9118

@ Input Parameter

Int32 **CardNumber** : The card id of the card that want to perform this operation.

Int32 **Channel** : Analog input channel number.
Range:0 through 15 for PCI-9112, PCI-9111, PCI-9118
Range: 0 through 31 for PCI-9113, PCI-9114

Int32 **AdRange** :The analog input range the specified channel is setting. Please refer to the Appendix B for the valid range values.

@ Output Parametet

Int32 **RetValue**: the error status of this user object. Please refer to Appendix A "*Error Codes*".

Int32 **Value** : The return value of A/D converted value

1.10 AI_VReadChannel

@ Description

This function performs a software triggered A/D conversion (analog input) on an analog input channel and returns the value scaled to a voltage in units of volts.

@ Cards Support

9111, 9112, 9113, 9114, 9118, 9812/10

@ Input Parameter

Int32 **CardNumber** :The card id of the card that want to perform this operation.

Int32 **Channel** :Analog input channel number.
Range: 0 through 15 for PCI-9112/cPCI-9112, PCI-9111, PCI-9118
Range: 0 through 31 for PCI-9113, PCI-9114
Range: 0 for PCI-9812/10 and cPCI-9812/10

Int32 **AdRange** :The analog input range the specified channel is setting. Please refer to the Appendix B for the valid range values.

@ Output Parameter

Int32 **RetValue**: the error status of this user object. Please refer to Appendix A "*Error Codes*".

Int32 **voltage** :The measured voltage value returned and scaled to units of voltage.

1.11 AI_VScale

@ Description

This function converts the result from an AI_ReadChannel call to the actual input voltage.

@ Cards Support

9111, 9112, 9113, 9114, 9118, 9812/10

@ Input Parameter

Int32 **CardNumber** :The card id of the card that want to perform this operation.

Int32 **AdRange** :The analog input range the specified channel is setting. Please refer to the Appendix B for the valid range values.

Int32 **reading** :The result of the AD Conversion.

@ Output Parameter

Int32 **RetValue**: the error status of this user object. Please refer to Appendix A "*Error Codes*".

Int32 **voltage** :Computed voltage value.

1.12 AO_6208A_Config

@ Description

Set the Voltage to Current mode of PCI-6208A.

@ Cards Support

6208A

@ Input Parameter

Int32 **CardNumber** : The card id of the card that want to perform this operation.

Int32 **V2AMode** : The voltage to current mode.

0: 0~20mA
1: 5~25mA
3: 4~20mA

@ Output Parametet

Int32 **RetValue**: the error status of this user object. Please refer to Appendix A "*Error Codes*".

1.13 AO_6308A_Config

@ Description

Sets the *Voltage to Current* Mode of PCI-6308A.

@ Cards Support

6308A

@ Input Parameter

Int32 **CardNumber** :The card id of the card that want to perform this operation.

Int32 **V2AMode** :The voltage to current mode. The valid V2Amode are:

P6308_CURRENT_0_20MA
P6308_CURRENT_5_25MA
P6308_CURRENT_4_20MA

@ Output Parametet

Int32 **RetValue**: the error status of this user object. Please refer to Appendix A "*Error Codes*".

1.14 AO_6308V_Config

@ Description

Informs PCIS-DASK library of the polarity (unipolar or bipolar) that the output channel is configured for the analog output and the reference voltage value selected for an analog output channel of PCI-6308V. You can configure each channel to use an internal reference of 10V or an external reference (0V ~ +10V) by setting related jumpers. You must call this function before calling function to perform voltage output operation.

@ Cards Support

6308V

@ Input Parameter

Int32 **CardNumber** :The card id of the card that want to perform this operation.

Int32 **Channel** :The AO channel number configured. The valid values are:
P6308V_AO_CH0_3
P6308V_AO_CH4_7

Int32 **OutputPolarity** :The polarity (unipolar or bipolar) of the output channel. The valid values are:
P6308V_AO_UNIPOLAR
P6308V_AO_BIPOLAR

Int32 **refVoltage** :Voltage reference value.
If the D/A reference voltage source your device use is internal reference, the valid values for *refVoltage* is 10.
If the D/A reference voltage source your device use is external reference, the valid range for *refVoltage* is 0 to +10.

@ Output Parametet

Int32 **RetValue**: the error status of this user object. Please refer to Appendix A "*Error Codes*".

Note : If the 10V D/A reference voltage is selected, the D/A output range is 0V~10V.

1.15 AO_9111_Config

@ Description

Informs PCIS-DASK library of the polarity (unipolar or bipolar) that the output channel is configured for the analog output of PCI9111. You must call this function before calling function to perform voltage output operation.

@ Cards Support

9111

@ Input Parameter

Int32 **CardNumber** :The card id of the card that want to perform this operation.

Int32 **OutputPolarity** :The polarity (unipolar or bipolar) of the output channel. The valid values are:
P9111_AO_UNIPOLAR
P9111_AO_BIPOLAR

@ Output Parametet

Int32 **RetValue**: the error status of this user object. Please refer to Appendix A "*Error Codes*".

1.16 AO_9112_Config

@ Description

Informs PCIS-DASK library of the reference voltage value selected for an analog output channel of PCI9112. You can configure each channel to use an internal reference of –5V (default) or –10V or an external reference (-10V ~ +10V) by setting related jumpers. You must call this function before calling function to perform voltage output operation.

@ Cards Support

9112

@ Input Parameter

Int32 **CardNumber** :The card id of the card that want to perform this operation.

Int32 **Channel** :The AO channel number configured.

Int32 **refVoltage** :Voltage reference value.
If the D/A reference voltage source your device use is internal reference, the valid values for *refVoltage* is –5 and –10.
If the D/A reference voltage source your device use is external reference, the valid range for *refVoltage* is –10 to +10.

@ Output Parametet

Int32 **RetValue**: the error status of this user object. Please refer to Appendix A "*Error Codes*".

Note : If the -10V D/A reference voltage is selected, the D/A output range is 0V~10V. On the other hand, if the +10V is selected, the D/A output range is -10V~0V.

1.17 AO_VScale

@ Description

Scales a voltage (or a current value) to a binary value.

@ Cards Support

9111, 9112, 9118, 6208V/16V/08A, 6308V/08A

@ Input Parameter

Int32 **CardNumber** :The card id of the card that want to perform this operation.

Int32 **Channel** :The analog output channel number.
Range: 0 or 1 for PCI-9112/cPCI-9112
Range: 0 for PCI-9111
Range: 0 or 1 for PCI-9118
Range: 0 through 7 for PCI-6208V/08A and PCI-6308V/08A
Range: 0 through 15 for PCI-6216V

@ Output Parametet

Int32 **RetValue**: the error status of this user object. Please refer to Appendix A "*Error Codes*".

Int32 **Voltage** :Voltage, in volts, to be converted to a binary value

Int32 **binValue** :the converted binary value returned

1.18 AO_VWriteChannel

@ Description

Accepts a voltage value (or a current value), scales it to the proper binary value and writes a binary value to the specified analog output channel.

@ Cards Support

9111, 9112, 9118, 6208V/16V/08A, 6308V/08A

@ Input Parameter

Int32 **CardNumber** :The card id of the card that want to perform this operation.

Int32 **Channel** :The analog output channel number.
Range: 0 or 1 for PCI-9112/cPCI-9112
Range: 0 for PCI-9111
Range: 0 or 1 for PCI-9118
Range: 0 through 7 for PCI-6208V/08A and PCI-6308V/08A
Range: 0 through 15 for PCI-6216V

Int32 **Voltage** :The value to be scaled and written to the analog output channel. The range of voltages depends on the type of device, on the output polarity, and on the voltage reference (external or internal).

@ Output Parametet

Int32 **RetValue**: the error status of this user object. Please refer to Appendix A "*Error Codes*".

1.19 AO_WriteChannel

@ Description

Writes a binary value to the specified analog output channel.

@ Cards Support

9111, 9112, 9118, 6208V/16V/08A

@ Input Parameter

Int32 **CardNumber** : The card id of the card that want to perform this operation.

Int32 **Channel** : The analog output channel number.
PCI-9112: 0 or 1
PCI-9111: 0
PCI-9118: 0 or 1
PCI-6208V/08A: 0 through 7
PCI-6216V: 0 through 15

Int32 **Value** : The value to be written to the analog output channel.
PCI-9111, PCI-9112, PCI-9118: 0 through 4095
PCI-6208A: 0 though 32767
PCI-6208V/16V: -32768 through 32767

@ Output Parametet

Int32 **RetValue**: the error status of this user object. Please refer to Appendix A "*Error Codes*".

1.20 CTR_8554_CK1_Config

@ Description

Selects the source of CK1.

@ Cards Support

8554

@ Input Parameter

Int32 **CardNumber** :The card id of the card that want to perform this operation.

Int32 **ClockSource** :The source of CK1.
0: C8M
1: COUT11

@ Output Parametet

Int32 **RetValue**: the error status of this user object. Please refer to Appendix A "*Error Codes*".

1.21 CTR_8554_ClkSrc_Config

@ Description

Selects PCI-8554 counter #1 ~ #10 clock source. (Clock source of counter #11 is 8MHz and clock source of counter #12 is from COUT11**,** both are fixed.)

@ Cards Support

8554

@ Input Parameter

Int32 **CardNumber** :The card id of the card that want to perform this operation.

Int32 **Ctr** : The counter number.
Range: 1~10

Int32 **ClockSource** :The clock source of the specified counter.
0: external clock source
1: the cascaded counter output (COUT n-1)
2: internal clock source CK1
3: output of the counter 10 (COUT10)

@ Output Parametet

Int32 **RetValue**: the error status of this user object. Please refer to Appendix A "*Error Codes*".

1.22 CTR_8554_Debounce_Config

@ Description

Selects debounce clock.

@ Cards Support

8554

@ Input Parameter

Int32 **CardNumber** :The card id of the card that want to perform this operation.

Int32 **DebounceClock** : 0: output of counter 11
1: 2MHz

@ Output Parametet

Int32 **RetValue**: the error status of this user object. Please refer to Appendix A "*Error Codes*".

1.23 CTR_Read

@ Description

Reads the current contents of the selected counter without disturbing the counting process.

@ Cards Support

9111, 9112, 9113, 9114, 9118, 7248, 7249, 7296, 7396, 8554

@Input Parameter

Int32 **CardNumber** :The card id of the card that want to perform this operation.

Int32 **Ctr** : The counter number.
Range: 0 for PCI-9111, PCI-9112/cPCI-9112, PCI-9113, PCI-9114, PCI-9118.
0, 1, 2 for PCI-7248/cPCI-7248, cPCI-7249R, PCI-7296.
1~12 for PCI-8554.

Int32 **Value** : Returns the current count of the specified counter.
Range: 0 through 65536 for binary mode (default).
0 through 9999 for BCD counting mode.

@ Output Parametet

Int32 **RetValue**: the error status of this user object. Please refer to Appendix A "*Error Codes*".

1.24 CTR_Reset

@ Description

Sets the output of the selected counter to the specified state.

@ Cards Support

9111, 9112, 9113, 9114, 9118, 7248, 7249, 7296, 7396, 8554

@Input Parameter

Int32 **CardNumber** :The card id of the card that want to perform this operation.

Int32 **Ctr** : The counter number.
Range: 0 for PCI-9111, PCI-9112/cPCI-9112, PCI-9113, PCI-9114, PCI-9118.
0, 1, 2 for PCI-7248/cPCI-7248, cPCI-7249R, PCI-7296.
1~12 for PCI-8554

Int32 **state** : The logic state to which the counter is to be reset.
Range: 0 or 1.

@ Output Parametet

Int32 **RetValue**: the error status of this user object. Please refer to Appendix A "*Error Codes*".

1.25 CTR_Setup

@ Description

Configures the selected counter to operate in the specified mode.

@ Cards Support

9111, 9112, 9113, 9114, 9118, 7248, 7249, 7296, 7396, 8554

@Input Parameter

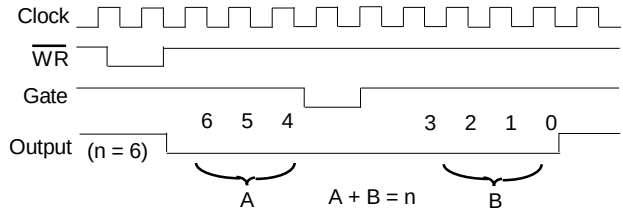
Int32 **CardNumber** :The card id of the card that want to perform this operation.

Int32 **Ctr** : The counter number.
Range: 0 for PCI-9111, PCI-9112/cPCI-9112, PCI-9113, PCI-9114, PCI-9118.
0, 1, 2 for PCI-7248/cPCI-7248, cPCI-7249R, PCI-7296.
1~12 for PCI-8554

Int32 **Mode** : The mode in which the counter is to operate.
Valid value:
TOGGLE_OUTPUT
PROG_ONE_SHOT
RATE_GENERATOR
SQ_WAVE_RATE_GENERATOR
SOFT_TRIG
HARD_TRIG

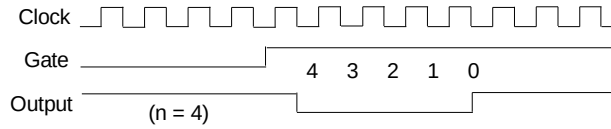
TOGGLE_OUTPUT:Toggle output from low to high on terminal count

In this mode, the output goes low after the mode set operation, and the counter begins to count down while the gate input is high. When terminal count is reached, the output goes high and remains high until the selected counter is set to a different mode. The following diagram shows the TOGGLE_OUTPUT mode timing diagram.



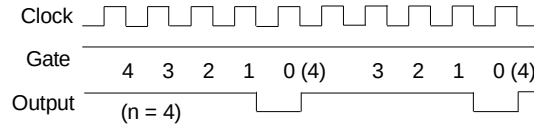
PROG_ONE_SHOT:Programmable one-shot

In this mode, the output goes low on the cofollowing the rising edge of the gate input and goes high on terminal count. The following diagram shows the PROG_ONE_SHOT mode timing diagram.



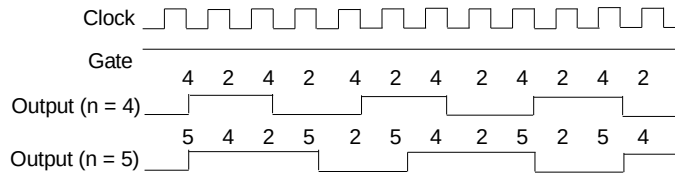
RATE_GENERATOR:Rate generator

In this mode, the output goes low for one period of the clock input. *count* indicates the period from one output pulse to the next. The following diagram shows the RATE_GENERATOR mode timing diagram.



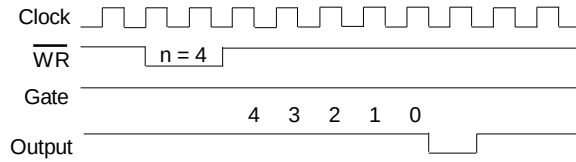
SQ_WAVE_RATE_GENERATOR:Square wave rate generator

In this mode, the output stays high for one half of the *count* clock pulses and stays low for the other half. The following diagram shows the SQ_WAVE_RATE_GENERATOR mode timing diagram.



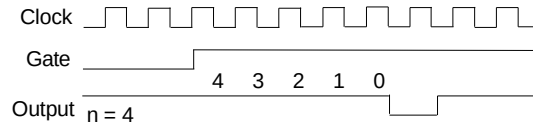
SOFT_TRIG:Software-triggered strobe

In this mode, the output is initially high, and the counter begins to count down while the gate input is high. On terminal count, the output goes low for one clock pulse, then goes high again. The following diagram shows the SOFT_TRIG mode timing diagram.



HARD_TRIG:Hardware-triggered strobe

This mode is similar to SOFT_TRIG mode except that the gate input is used as a trigger to start counting. The following diagram shows the HARD_TRIG mode timing diagram.



Int32 **Count** : The period from one output pulse to the next.

Int32 **BinBcd** : Whether the counter operates as a 16-bit binary counter or as a 4-decade binary-coded decimal (BCD) counter.
Valid value:
BIN: 16-bit binary counter.
BCD: 4-decade BCD counter.

Int32 **RetValue**: the error status of this user object. Please refer to Appendix A "*Error Codes*".

1.26 DI_7200_Config

@ Description

Set the trigger source, and input mode for PCI-7200 with card ID *CardNumber*. You must call this function before calling function to perform continuous digital input operation.

@ Cards Support

7200

@ Input Parameter

Int32 **CardNumber** : The card id of the card that want to perform this operation.

Int32 **TriSource** : The trigger mode for continuous digital input.
1: on-board programmable pacer
2: external signal trigger
3: handshaking

Int32 **ExtTrigEn** : External Trigger Enable
0: input sampling starts immediately
2: digital input sampling waits rising or falling edge of I_TRG to start DI

Int32 **TrigPol** : Trigger Polarity, the valid values are:
0: I_TRG is falling edge active
4: I_TRG is rising edge active

Int32 **I_REQ_Pol** :I_REQ Polarity (for handshaking mode)
0: I_REQ is falling edge active
8: I_REQ is rising edge active

@ Output Parametet

Int32 **RetValue**: the error status of this user object. Please refer to Appendix A "*Error Codes*".

1.27 DI_7300B_Config

@ Description

Set the trigger source, port width, etc. for PCI-7300A Rev.B board with card ID *CardNumber*. You must call this function before calling function to perform continuous digital input operation.

@ Cards Support

7300A Rev.B

@ Input Parameter

Int32 **CardNumber** : The card id of the card that want to perform this operation.

Int32 **PortWidth** : The width of digital input port (PORT A). The valid value is 0, 8, 16, or 32.

Int32 **TrigSource** : The trigger mode for continuous digital input.
- **1:** on-board programmable pacer timer0
- **2:** external signal trigger
- **3:** handshaking
- **4:** 10MHz clock
- **5:** 20MHz clock

Int32 **WaitStatus** : DI Wait Trigger Status, the valid values are:
- **0:** input sampling starts immediately
- **1:** digital input sampling waits rising or falling edge of I_TRG to start DI

Int32 **Terminator** : PortA Terminator On/Off, the valid values are:
- **1:** terminator on
- **0:** terminator off

Int32 **I_Cntrl_Pol** : The polarity configuration. This argument is an integer expression formed from one or more of the manifest constants defined in DASK.H. There are three groups of constants:

(1) DIREQ
0: DIREQ signal is rising edge active
1: DIREQ signal is falling edge active
(2) DIACK
0: DIACK signal is rising edge active
2: DIACK signal is falling edge active
(3) DITRIG
0: DITRIG signal is rising edge active
4: DITRIG signal is falling edge active

Int32 **ClearFifo** :FALSE: retain the FIFO data
TRUE: clear FIFO data before perform digital input

Int32 **DisableDI** :FALSE: digital input operation still active after DMA transfer complete. The input data still put into FIFO
TRUE: disable digital input operation immediately when DMA transfer complete

@Return Codes
NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport

1.28 DI_ContReadPort

@ Description

This function performs continuous digital input on the specified digital input port at a rate as close to the rate you specified.

@ Cards Support

7200, 7300A Rev.B

@ Input Parameter

Int32 **CardNumber** : The card id of the card that want to perform this operation.

Int32 **Port** : Digital input port number. For PCI-7200 and PCI-7300A, this argument must be set to 0.

Int32 **Buffer** : The starting address of the memory to contain the input data. This memory must have been allocated for enough space to store input data.

Int32 **ReadCount** : the number of input operation to be performed.

Real **SampleRate** : The sampling rate you want for digital input in hertz (samples per second). Your maximum rate depends on the card type and your computer system. This argument is only useful if the DI trigger mode was set as internal programmable pacer by calling **`DI_7200_Config`** or **`DI_7300B_Config`**. For the other settings, you have to set this argument as 10000.

Int32 **TransferCount** :TransferCount is the actual number of digital data input.

@ Output Parametet

Int32 **RetValue**: the error status of this user object. Please refer to Appendix A "*Error Codes*".

Int32 **Buffer** : The user buffer to which the data is to be copied.

Int32 **TransferCount** :the actual number of digital data input.

1.29 DI_ReadLine

@ Description

Read the digital logic state of the specified digital line in the specified port.

@ Cards Support

6208V/16V/08A, 6308V/08A, 7200, 7230, 7233, 7248, 7249, 7250/51, 7252, 7296, 7300A, 7396, 7432, 7433, 8554, 9111, 9112, 9114, 9118

@ Input Parameter

Int32 **CardNumber** :The card id of the card that want to perform this operation.

Int32 **Port** : Digital input port number. The valid value:
PCI-6208V/16V/08A: 0
PCI-6308V/08A: 0

PCI-7200 : 0
cPCI-7200: 0, 1 (auxiliary input port)
PCI-7230/cPCI-7230: 0
PCI-7233: 0
PCI-7248/cPCI-7248:
Channel_P1A, Channel_P1B,
Channel_P1C, Channel_P1CL,
Channel_P1CH, Channel_P2A,
Channel_P2B, Channel_P2C,
Channel_P2CL, Channel_P2CH
cPCI-7249R:
Channel_P1A, Channel_P1B,
Channel_P1C, Channel_P1CL,
Channel_P1CH, Channel_P1AE,
Channel_P1BE, Channel_P1CE,
Channel_P2A, Channel_P2B,
Channel_P2C, Channel_P2CL,
Channel_P2CH, Channel_P2AE,
Channel_P2BE, Channel_P2CE,
PCI-7250/51: 0 through 3
cPCI-7252: 0
PCI-7296:
Channel_P1A, Channel_P1B,
Channel_P1C, Channel_P1CL,
Channel_P1CH, Channel_P2A,
Channel_P2B, Channel_P2C,
Channel_P2CL, Channel_P2CH,
Channel_P3A, Channel_P3B,
Channel_P3C, Channel_P3CL,
Channel_P3CH, Channel_P4A,
Channel_P4B, Channel_P4C,
Channel_P4CL, Channel_P4CH
PCI-7396:
Channel_P1A, Channel_P1B,
Channel_P1C,
Channel_P2A, Channel_P2B,
Channel_P2C,
Channel_P3A, Channel_P3B,
Channel_P3C,
Channel_P4A, Channel_P4B,

Channel_P4C
PCI-7300A/cPCI-7300A: 1 (auxiliary input port)
PCI-7432/cPCI-7432: 0
PCI-7433/cPCI-7433: PORT_DI_LOW, PORT_DI_HIGH
PCI-8554: 0
PCI-9111: P9111_CHANNEL_DI, P9111_CHANNEL_EDI
PCI-9112/cPCI-9112: 0
PCI-9114: 0
PCI-9118: 0

Int32 **Line** : The digital line to be read. The valid value:
PCI-6208V/16V/08A: 0 through 3
PCI-6308V/08A: 0 through 3
PCI-7200/cPCI-7200: 0 through 31 (for port 0)
0 through 3 (for auxiliary input port of cPCI7200)
PCI-7230/cPCI-7230: 0 through 15
PCI-7233: 0 through 31
PCI-7248/cPCI-7248: 0 through 7
cPCI-7249R: 0 through 7
PCI-7250/51: 0 through 7
cPCI-7252: 0 through 15
PCI-7296: 0 through 7
PCI-7300A/cPCI-7300A: 0 through 3
PCI-7396: 0 through 7
PCI-7432/cPCI-7432/cPCI-7432R: 0 through 31
PCI-7433/cPCI-7433/cPCI-7433R: 0 through 31
PCI-8554: 0 through 7
PCI-9111: 0 through 15
PCI-9112/cPCI-9112: 0 through 15
PCI-9114: 0 through 15
PCI-9118: 0 through 3

@ Output Parametet

Int32 **RetValue**: the error status of this user object. Please refer to Appendix A "*Error Codes*".

Int32 **State** : Returns the digital logic state, 0 or 1, of the specified line.

1.30 DI_ReadPort

@ Description

Read digital data from the specified digital input port.

@ Cards Support

6208V/16V/08A, 7200, 7230, 7248, 7250/51, 7296, 7300A Rev.B, 7432, 7433, 7434, 9111, 9112, 9114, 9118

@ Input Parameter

Int32 **CardNumber** : The card id of the card that want to perform this operation.

Int32 **Port** : Digital input port number. The valid value:

PCI-6208V/16V/08A: 0
PCI-7200: 0
PCI-7230: 0
PCI-7250/51: 0 through 3
PCI-7248:
P1A: 0, P1B: 1, P1C: 2, P1C Lower: 3, P1C Upper: 4
P2A: 5, P2B:6, P2C: 7, P2C Lower:8, P2C Upper: 9
PCI-7296:
P1A: 0, P1B: 1, P1C: 2, P1C Lower: 3, P1C Upper: 4
P2A: 5, P2B:6, P2C: 7, P2C Lower:8, P2C Upper: 9
P3A: 10, P3B: 11, P3C: 12, P3C Lower: 13, P3C Upper: 14
P2A: 15, P2B: 16, P2C: 17, P2C Lower: 18, P2C Upper: 19
PCI-7300A: 1 (auxiliary input port)
PCI-7432: 0

PCI-7433: 0: PORT_DI_LOW, 1: PORT_DI_HIGH
PCI-9111: 0
PCI-9112: 0
PCI-9114: 0
PCI-9118: 0

@ Output Parametet

Int32 **RetValue**: the error status of this user object. Please refer to Appendix A "*Error Codes*".

Int32 **Value** : Returns the digital data read from the specified port.
PCI-6208V/16V/08A: 4-bit data
PCI-7200: 32-bit data
PCI-7230: 16-bit data
PCI-7248: 8-bit data
PCI-7250/51: 8-bit data
PCI-7296: 8-bit data
PCI-7300A: 4-bit data
PCI-7432: 32-bit data
PCI-7433: 32-bit data
PCI-9111: 16-bit data
PCI-9112: 16-bit data
PCI-9114: 16-bit data
PCI-9118: 4-bit data

1.31 DIO_PortConfig

@ Description

Set the direction configuration (Input or output) of the selected port.

@ Cards Support

7248, 7296

@ Input Parameter

Int32 **CardNumber** : The card id of the card that want to perform this operation.

Int32 **Port** : The port selected. The valid value:
PCI-7248:

P1A: 0, P1B: 1, P1C: 2, P1C Lower: 3, P1C Upper: 4
P2A: 5, P2B:6, P2C: 7, P2C Lower:8, P2C Upper: 9

PCI-7296:

P1A: 0, P1B: 1, P1C: 2, P1C Lower: 3, P1C Upper: 4
P2A: 5, P2B:6, P2C: 7, P2C Lower:8, P2C Upper: 9
P3A: 10, P3B: 11, P3C: 12, P3C Lower: 13, P3C Upper: 14
P2A: 15, P2B: 16, P2C: 17, P2C Lower: 18, P2C Upper: 19

Int32 **Direction** :The port direction of PIO port.
1: Input
2: Output

@ Output Parametet

Int32 **RetValue**: the error status of this user object. Please refer to Appendix A "*Error Codes*".

1.32 DO_7200_Config

@ Description

Set the trigger source and output mode for PCI-7200 with card ID *CardNumber*. You must call this function before calling function to perform continuous digital output operation.

@ Cards Support

7200

@ Input Parameter

Int32 **CardNumber** : The card id of the card that want to perform this operation.

Int32 **TrigSource** :The trigger source for continuous digital input.
1: on-board programmable pacer

3: handshaking

Int32 **OutReqEn** : Output REQ Enable
0: output REQ is disable
16: output REQ is enabled, an O_REQ strobe is generated after output data is strobe

Int32 **OutTrigSig** : Output Trigger Signal
0: O_TRIG signal goes low
32: O_TRIG signal goes high

@ Output Parameter

Int32 **RetValue**: the error status of this user object. Please refer to Appendix A "*Error Codes*".

1.33 DO_7300B_Config

@ Description

Set the trigger source, port width, etc. for PCI7300 with card ID *CardNumber*. You must call this function before calling function to perform continuous digital output operation.

@ Cards Support

7300A Rev.B

@ Input Parameter

Int32 **CardNumber** : The card id of the card that want to perform this operation.

Int32 **PortWidth** :The width of digital output port (PORT B). The valid value is 0, 8, 16, or 32.

Int32 **TrigSource** :The trigger mode for continuous digital output.
Valid values are:
1: on-board programmable pacer timer1
2: external signal trigger
4: 10MHz clock
5: 20MHz clock
6: burst handshaking mode by using timer1 output as output clock

7: burst handshaking mode by using 10MHz clock as output clock
8: burst handshaking mode by using 20MHz clock as output clock

Int32 **WaitStatus** : DO Wait Status
0: digital output starts immediately
1: digital output waits rising or falling edge of O_TRG to start
2: delay output data until FIFO is not almost empty
3: delay output data until O_TRG active and FIFO is not almost empty

Int32 **Terminator** : PortB Terminator On/Off
1: terminator on
0: terminator off

O_Cntrl_Pol : The polarity configuration. This argument is an integer expression formed from one or more of the manifest constants defined in DASK.H. There are three groups of constants:
(1) DOREQ
0: DOREQ signal is rising edge active
8: DOREQ signal is falling edge active
(2) DOACK
0: DOACK signal is rising edge active
16: DOACK signal is falling edge active
(3) DOTRIG
0: DOTRIG signal is rising edge active
32: DOTRIG signal is falling edge active

FifoThreshold : programmable almost empty threshold of both PORTB FIFO and PORTA FIFO (if output port width is 32).

@ Output Parametet

Int32 **RetValue**: the error status of this user object. Please refer to Appendix A "*Error Codes*".

1.34 DO_ContWritePort

@ Description

This function performs continuous digital output on the specified digital output port at a rate as close to the rate you specified.

@ Cards Support

7200, 7300A Rev.B

@ Input Parameter

Int32 **CardNumber** : The card id of the card that want to perform this operation.

Int32 **Port** : Digital input port number. For PCI-7200 and PCI-7300A, this argument must be set to 0.

Int32 **Buffer** : The starting address of the memory containing the output data. This memory must has been allocated for enough space to store output data.

Int32 **WriteCount** : *WriteCount* is the number of output operation to be performed.

Int32 **Iterations** : The repeat time of performs continuous digital output on the specified digital output port. It will output forever with *Iterations* as 0.

Real **SampleRate** : The sampling rate you want for digital output in hertz (samples per second). Your maximum rate depends on the card type and your computer system. This argument is only useful if the DO trigger mode was set as internal programmable pacer by calling **`DO_7200_Config`** or **`DO_7300B_Config`**. For the other settings, you have to set this argument as 10000.

Int32 **TransferCount** :TransferCount is the actual number of data output.

@ Output Parametet

Int32 **RetValue**: the error status of this user object. Please refer to Appendix A "*Error Codes*".

Int32 **Buffer** : The starting address of the memory containing the output data.

Int32 **TransferCount** :the actual number of data output.

1.35 DO_ReadLine

@ Description

Read back the digital logic state of the specified digital output line in the specified port.

@ Cards Support

6208, 6308, 7200, 7248, c7249R, 7296, 7300A, 7396, 7250/51, 7252, 9118

@ Input Parameter

Int32 **CardNumber** :The card id of the card that want to perform this operation.

Int32 **Port** : Digital output port number. The valid value:
PCI-6208V/16V/08A: 0
PCI-6308V/08A: 0
PCI-7200: 0
cPCI-7200: 0, 1 (auxiliary output port)
PCI-7250/51: 0 through 3
cPCI-7252: 0
PCI-9118DG/HG/HR: 0
PCI-7300A/cPCI-7300A: 1 (auxiliary output port)
PCI-7248/96, cPCI-7249R, PCI-7396: refer to the function *DI_ReadLine* section.

Int32 **Line** : The digital line to be accessed. The valid value:
PCI-6208V/16V/08A: 0 through 3
PCI-6308V/08A: 0 through 3
PCI-7200/cPCI-7200: 0 through 31 (for port 0)
0 through 3 (auxiliary output port of cPCI-7200)
PCI-7250/51: 0 through 7
cPCI-7252: 0 through 7

PCI-7300A/cPCI-7300A: 0 through 3
PCI-9118DG/HG/HR: 0 through 3
PCI-7248/96, cPCI-7249R, PCI-7396: refer to the function *DI_ReadLine* section.

@ Output Parametet

Int32 **RetValue**: the error status of this user object. Please refer to Appendix A "*Error Codes*".

Int32 **State** : Returns the digital logic state, 0 or 1, of the specified line.

1.36 DO_ReadPort

@ Description

Read back the output digital data from the specified digital output port.

@ Cards Support

6208, 6308, 7200, 7248, c7249R, 7296, 7300A, 7396, 7250/51, 7252, 9118

@ Parameter

Int32 **CardNumber** :The card id of the card that want to perform this operation.

Int32 **Port** : Digital output port number. The valid value:
PCI-6208V/16V/08A: 0
PCI-6308V/08A: 0
PCI-7200: 0
cPCI-7200: 0, 1 (auxiliary output port)
PCI-7250/51: 0 through 3
cPCI-7252: 0
PCI-9118DG/HG/HR: 0
PCI-7300A/cPCI-7300A: 1 (auxiliary output port)
PCI-7248/96, cPCI-7249R, PCI-7396: refer to the function *DI_ReadPort* section.

@ Output Parametet

Int32 **RetValue**: the error status of this user object. Please refer to Appendix A "*Error Codes*".

Int32 **Value** : Returns the digital data read from the specified output port.
PCI-6208V/16V/08A: 4-bit data
PCI-6308V/08A: 4-bit data
PCI-7200/cPCI-7200: 32-bit data (for port 0)
4-bit data (for auxiliary output port of cPCI-7200)
PCI-7250/51: 8-bit data
cPCI-7252: 8-bit data
PCI-7300A/cPCI-7300A: 4-bit data
PCI-9118DG/HG/HR: 4-bit data
PCI-7248/96, cPCI-7249R, PCI-7396: refer to the function *DI_ReadPort* section.

1.37 DO_WriteLine

@ Description

Sets the specified digital output line in the specified digital port to the specified state. This function is only available for these cards that support digital output read-back functionality.

@ Cards Support

6208, 6308, 7200, 7248, c7249R, 7296, 7300A, 7396, 7250/51, 7252, 9118

@ Parameter

Int32 **CardNumber** :The card id of the card that want to perform this operation.

Int32 **Port** : Digital output port number. The valid value:
PCI-6208V/16V/08A: 0
PCI-6308V/08A: 0
PCI-7200: 0
cPCI-7200: 0, 1 (auxiliary output port)
PCI-7250/51: 0 through 3

cPCI-7252: 0
PCI-9118DG/HG/HR: 0
PCI-7300A/cPCI-7300A: 1 (auxiliary output port)
PCI-7248/96, cPCI-7249R, PCI-7396: refer to the function *DI_ReadLine* section.

Int32 **Line** : The digital line to write to. The valid value:
PCI-6208V/16V/08A: 0 through 3
PCI-6308V/08A: 0 through 3
PCI-7200/cPCI-7200: 0 through 31(for port 0)
: 0 through 3 (auxiliary output port of cPCI-7200)
PCI-7250/51: 0 through 7
cPCI-7252: 0 through 7
PCI-7300A/cPCI-7300A: 0 through 3
PCI-9118DG/HG/HR: 0 through 3
PCI-7248/96, cPCI-7249R, PCI-7396: refer to the function *DI_ReadLine* section.

Int32 **State** : The new digital logic state, 0 or 1.

@ Output Parametet

Int32 **RetValue**: the error status of this user object. Please refer to Appendix A "*Error Codes*".

1.38 DO_WritePort

@ Description

Writes digital data to the specified digital output port.

@ Cards Support

6208V/16V/08A, 7200, 7230, 7248, 7250/51, 7296, 7300A Rev.B, 7432, 7434, 9111, 9112, 9118

@ Input Parameter

Int32 **CardNumber** :The card id of the card that want to perform this operation.

Int32 **Port** : Digital output port number. The cards that support this function and their corresponding valid value are as follows:

PCI-6208V/16V/08A: 0
PCI-7200: 0
PCI-7230: 0
PCI-7250/51: 0 through 3
PCI-7248:
P1A: 0, P1B: 1, P1C: 2, P1C Lower: 3, P1C Upper: 4
P2A: 5, P2B:6, P2C: 7, P2C Lower:8, P2C Upper: 9
PCI-7296:
P1A: 0, P1B: 1, P1C: 2, P1C Lower: 3, P1C Upper: 4
P2A: 5, P2B:6, P2C: 7, P2C Lower:8, P2C Upper: 9
P3A: 10, P3B: 11, P3C: 12, P3C Lower: 13, P3C Upper: 14
P2A: 15, P2B: 16, P2C: 17, P2C Lower: 18, P2C Upper: 19
PCI-7300A: 1 (auxiliary output port)
PCI-7432: 0
PCI-7434: 0:PORT_DO_LOW, 1:PORT_DO_HIGH
PCI-9111: 0
PCI-9112: 0
PCI-9114: 0
PCI-9118: 0

Int32 **Value** : Digital data that is written to the specified port.

PCI-6208V/16V/08A: 4-bit data
PCI-7200: 32-bit data
PCI-7230: 16-bit data
PCI-7248: 8-bit data
PCI-7250/51: 8-bit data
PCI-7296: 8-bit data
PCI-7300A: 4-bit data
PCI-7432: 0 through 31
PCI-7434: 0 through 31
PCI-9111: 16-bit data
PCI-9112: 16-bit data

PCI-9114: 16-bit data
PCI-9118: 4-bit data

@ Output Parametet

Int32 **RetValue**: the error status of this user object. Please refer to Appendix A "*Error Codes*".

1.39 GetSample

@ Description

This function uses 16-bit as a item to divide the buffer, then get a 16 bit unsigned integer of the index i in the *buffer* (a memory block).

@ Cards Support

9111, 9112, 9118, 9812/10

@ Input Parameter

Int32 **buffer** : An data array that want to use 16-bit as a item to be divided.

Int32 **index** : The index-th 16 bit unsigned integer data want to get from *buffer*.

@ Output Parametet

Int32 **RetValue**: The value of the Index-th 16 bit unsigned integer data get from *buffer*.

Int32 **buffer** : An data array that want to use 16-bit as a item to be divided.

1.40 Register_Card

@ Description

Initializes the hardware and software states of an NuDAQ PCI-bus data acquisition card, and then returns a numeric card ID that corresponds to the card initialized. **`Register_Card`** must be called before any other PCIS-VEE/NT user object can be

called for that card. The function initializes the card and variables internal to PCIS-VEE/NT. Because NuDAQ PCI-bus data acquisition cards meets the plug-and-play design, the base address (pass-through address) and IRQ level are assigned by system BIOS directly.

@ Cards Support

6208V/6216V, 6208A, 7200, 7230, 7248, 7250/51, 7296, 7300A Rev.B, 7432, 7433, 7434, 9111, 9112, 9113, 9114, 9118, 9812/10

@ Input Parameter

Int32 **CardType** : The type of card to be initialized.

PCI-6208V: 1
PCI-6208A: 2
PCI-6308V: 3
PCI-6308A: 4
PCI-7200: 5
PCI-7230: 6
PCI-7233: 7
PCI-7234: 8
PCI-7248: 9
PCI-7249: 10
PCI-7250: 11
PCI-7252: 12
PCI-7296: 13
PCI-7300A_RevA: 14
PCI-7300A_RevB: 15
PCI-7432: 16
PCI-7433: 17
PCI-7434: 18
PCI-8554: 19
PCI-9111DG: 20
PCI-9111HR: 21

PCI-9112: 22
PCI-9113: 23
PCI-9114DG: 24
PCI-9114HG: 25
PCI-9118DG: 26
PCI-9118HG: 27
PCI-9118HR: 28

PCI-9810: 29
PCI-9812: 30

Int32 **card_num** : The sequence number of card with the same card type. The first card (in the most prior slot) is with card_num=0. For example, if there are two PCI-7200 cards and one PCI-9112 card plugged on your PC, the PCI-7200 card in prior slot should be registered with card_num=0, the other PCI-7200 card with card_num=1. The PCI-9112 card should be registered with card_num=0.

@ Output Parametet

Int32 **Card_Number** :This function returns a numeric card id for the card initialized. The range of card id is between 0 and 31. If there is any error occurs, it will return negative error code. Please refer to Appendix A "*Error Codes*".

1.41 Release_Card

@ Description

There are at most 32 cards that can be registered simultaneously. This function is used to tell PCIS-VEE/NT that this registered card is not used currently and can be released. This would make room for new card to register. Also by the end of a program, you need to use this function to release all cards that were registered.

@ Cards Support

6208V/6216V, 6208A, 7200, 7230, 7248, 7250/51, 7296, 7300A Rev.B, 7432, 7433, 7434, 9111, 9112, 9113, 9114, 9118, 9812/10

@ Input Parameter

Int32 **CardNumber** : The card id of the card that want to perform this operation.

@ Output Parameter

NoOutput Parameter.

Appendix A Error Codes

This appendix lists the status codes returned by PCIS-VEE/NT, including the name and description.

Each PCIS-VEE/NT function returns a status code that indicates whether the function was performed successfully. When an PCIS-VEE/NT function returns a negative number, it means that an error occurred while executing the function.

Status Code	Status Name	Description
0	NoError	No error occurred
-1	ErrorUnknownCardType	The C*ardType* argument is not valid
-2	ErrorInvalidCardNumber	The *CardNumber* argument is out of range (larger than 15).
-3	ErrorTooManyCardRegistered	There have been 16 cards that were registered.
-4	ErrorCardNotRegistered	No card registered as id *CardNumber*.
-5	ErrorFuncNotSupport	The function called is not supported by this type of card..
-6	ErrorInvalidIoChannel	The specified *Channel* or *Port* argument is out of range..
-7	ErrorInvalidAdRange	The specified analog input range is invalid.
-8	ErrorContIoNotAllowed	The specified continuous IO operation is not supported by this type of card.
-13	ErrorOpenDriverFailed	Failed to open the device driver.
-15	ErrorTransferCountTooLarge	The size of transfer is larger than the size of Initially allocated memory in driver.
-16	ErrorNotDoubleBufferMode	Double buffer mode is disabled.

-18	ErrorInvalidCounterMode	The value of the *Mode* argument is invalid.
-19	ErrorInvalidCounter	The value of the *Ctr* argument is out of range.
-20	ErrorInvalidCounterState	The value of the *State* argument is out of range.
-21	ErrorInvalidBinBcdParam	The value of the *BinBcd* argument is invalid.
-22	ErrorBadCardType	The value of *CardType* argument is invalid
-29	ErrorNotInputPort	The value of AI/DI port argument is invalid
-30	ErrorNotOutputPort	The value of AO/DO argument is invalid
-31	ErrorInvalidDioPort	The value of DI/O port argument is invalid
-32	ErrorInvalidDioLine	The value of DI/O line argument is invalid
-33	ErrorContIoActive	Continuous IO operation is not active
-35	ErrorConfigFailed	The specified function configuration is failed
-36	ErrorInvalidPortDirection	The value of DIO port direction argument is invalid

Appendix B AI Range Codes

The ***Analog Input Range*** of NuDAQ PCI-bus Cards

Constant Value	*Analog Input Range*
1	Bipolar -10V to +10V
2	Bipolar -5V to +5V
3	Bipolar -2.5V to +2.5V
4	Bipolar -1.25V to +1.25V
5	Bipolar -0.625V to +0.625V
6	Bipolar -0.3125V to +0.3125V
7	Bipolar -0.5V to +0.5V
8	Bipolar -0.05V to +0.05V
9	Bipolar -0.005V to +0.005V
10	Bipolar -1V to +1V
11	Bipolar -0.1V to +0.1V
12	Bipolar -0.01V to +0.01V
13	Bipolar -0.01V to +0.001V
14	Unipolar 0 to +20V
15	Unipolar 0 to +10V
16	Unipolar 0 to +5V
17	Unipolar 0 to +2.5V
18	Unipolar 0 to +1.25V
19	Unipolar 0 to +1V
20	Unipolar 0 to +0.1V
21	Unipolar 0 to +0.01V
22	Unipolar 0 to +0.001V

Valid values for each card:

PCI-9111 DG/HR	: 1, 2, 3, 4, 5
PCI-9112	: 1, 2, 3, 4, 5, 15, 16, 17, 18
PCI-9113	: 1, 10, 11, 2, 7, 8, 15, 19, 20
PCI-9114 HG	: 1, 10, 11, 12
PCI-9114 DG	: 1, 2, 3, 4
PCI-9118 DG/HR	: 2, 3, 4, 5, 15, 16, 17, 18
PCI-9118 HG	: 2, 7, 8, 9, 15, 19, 20, 21
PCI-9812/10	: 10, 2