I-9012

I/O Module User Manual

V1.0.0 May 2019

Technical support: service@icpdas.com

Author: Edward Ku

Editor: Anna Huang

Warranty

All products manufactured by ICP DAS are under warranty regarding defective materials for a period of one year, beginning from the date of delivery to the original purchaser.

Warning

ICP DAS assumes no liability for any damage resulting from the use of this product. ICP DAS reserves the right to change this manual at any time without notice. The information furnished by ICP DAS is believed to be accurate and reliable. However, no responsibility is assumed by ICP DAS for its use, nor for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Trademarks

Names are used for identification purposes only and may be registered trademarks of their respective companies.

Contact Us

If you have any problems, please feel free to contact us by email at: service@icpdas.com
You can count on us for a quick response.

Table of Contents

1. Introduction

The I-9012 is a high-performance Analog Input module that provides 16-bit resolution and a sampling rate of up to 200 kS/s per channel. The I-9012 provide eight input channels, and the input range is programmable to ±5 V or ±10 V. The Trigger Output and the Trigger Input Pin cloud allow a multitude of I-9012 modules to be simultaneously sampled. The module also provides 4 kV ESD protection and 2500 Vrms intra-module isolation.

Applicable Platforms

Platform	OS	Module
XPAC	XP-9000-WES7 (WES7)	I-9012
WinPAC	WP-9000-CE7 (CE 7.0)	I-9012
LinPAC	LinPAC-9000 (Linux kernel 3.2/4.4)	I-9012

Features

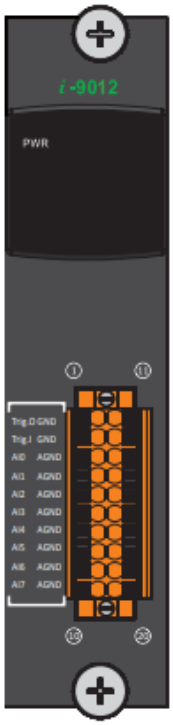
- 16-bit AD Converter
- Simultaneous Sampling
- 200 kHz sample rate for a single channel
- 2500 Vrms intra-module isolation
- 8 KB FIFO
- 4 kV ESD Protection
- Wide Operating Temperature Range: -25 to +75°C

1.1. Specifications

Analog Input		
Channels		8 (single-ended)
Input Range		±5 V, ±10 V
Resolution		16-bit
Sample Rate		Single Channel Pacer Internal Mode: 200 kS/s Single Channel Pacer External Mode: 200 kS/s Single Channel Polling Mode: 40 kS/s
FIFO		8 KB
AD Trigger Mode	Polling	Software Trigger
	Pacer	Internal Clock
		External Clock
Trigger Output		
Frequency Range		16 Hz to 200 kHz
Compatibility		5 V/TTL
Output Type		Pulse
Pulse Width		100 ns/10 MHz
Trigger Input		
Frequency Range		1 Hz to200 kHz
Trigger Edge		Falling Edge
High Logic		< 5 V
Low Logic		> 0.8 V
LED Display		
System LED Indictors		1 LED as Power Indicator
I/O LED Indicators		--

Isolation	
Intra-module Isolation, Field to logic	2500 Vrms
EMS Protection	
ESD (IEC 61000-4-2)	±4 kV Contact for Each Terminal
	±8 kV Air for Random Point
Power	
Power Consumption	1 W Max
Mechanical	
Dimensions (L × W × H)	144 mm × 31 mm × 134 mm
Environment	
Operating Temperature	-25 to +75°C
Storage Temperature	-40 to +85°C
Humidity	10 to 90% RH, Non-condensing

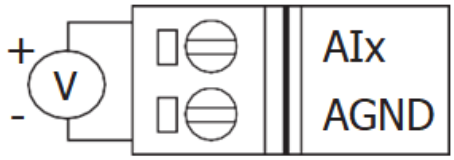
1.2. Pin Assignments



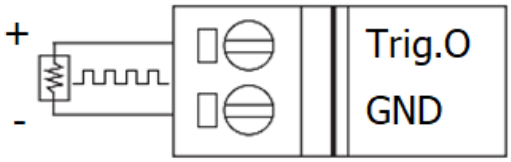
Pin Assignment	Terminal No.		Pin Assignment
Trig. O	01	11	GND
Trig. I	02	12	GND
AI0	03	13	AGND
AI1	04	14	AGND
AI2	05	15	AGND
AI3	06	16	AGND
AI4	07	17	AGND
AI5	08	18	AGND
AI6	09	19	AGND
AI7	10	20	AGND

1.3. Wiring Connections

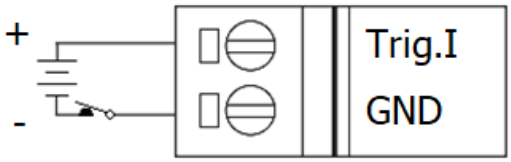
Voltage Input Wiring



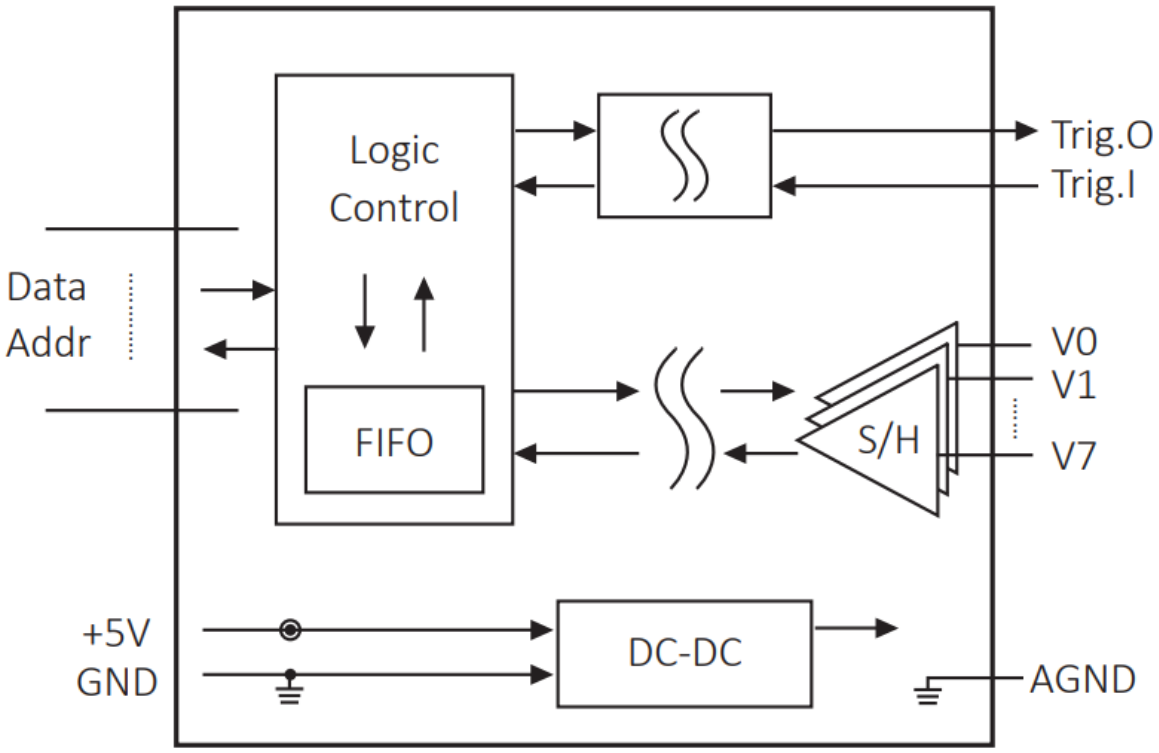
Trig Output Wiring



Trig Input Wiring



1.4. Block Diagram



2. Quick Start

This section provides details of the functions for the I-9012, together with demos for Windows-based platforms. Demos are provided for both C# and C++. To download the demos, refer to Chapter 3 “Demo Programs”.

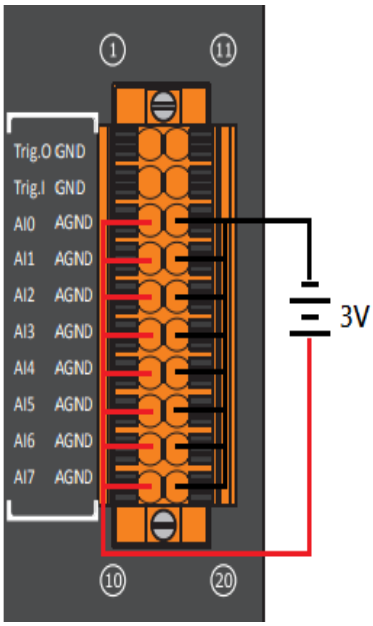
The following provides the name of the demos for each function:

Language / Function	C++	C#
Read Analog Input	pac_i8012W_Basic_Info	pac_i8012W_Basic_Info
Magic Scan Block	pac_i8012W_MagicScan_Block	pac_i8012W_MagicScan_Block
Magic Scan NonBlock	pac_i8012W_MagicScan_NonBlock	pac_i8012W_MagicScan_NonBlock
Magic Scan ISR	pac_i8012W_MagicScan_ISR	NA
Multiple Module Scan	pac_i8012W_Multi_Modules_Scan	pac_i8012W_Multi_Modules_Scan

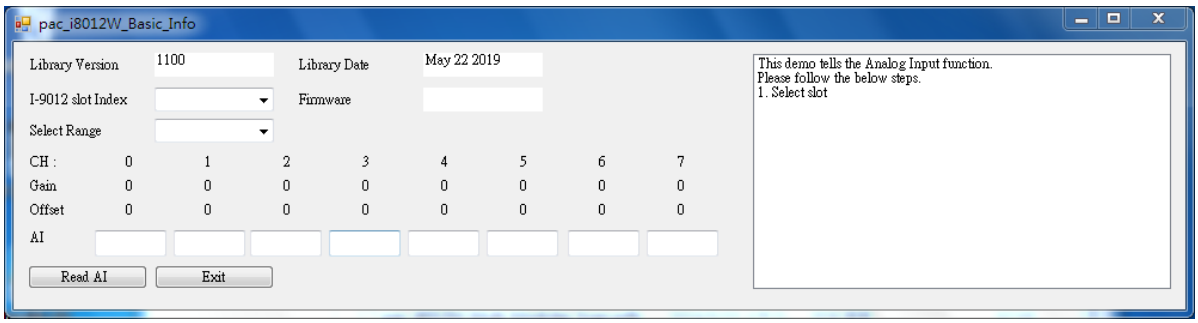
2.1. Reading the Analog Input

This function is used to read the analog signal. If there is an unused channel, connect it to the AGND pin to avoid creating floating data.

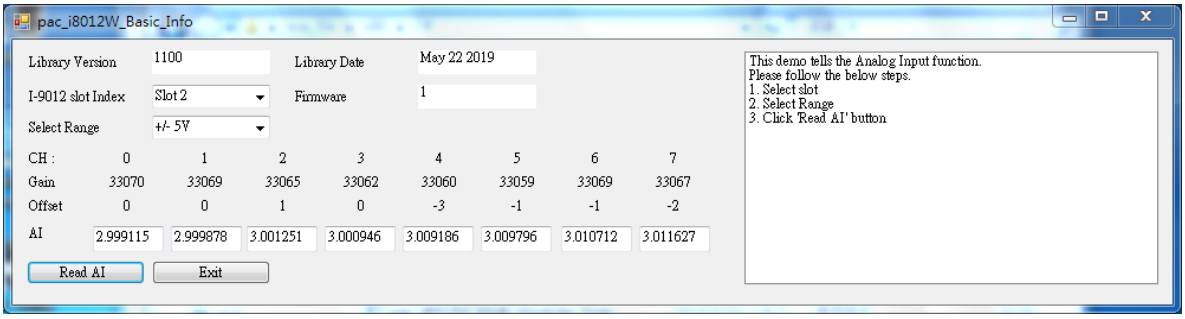
In this case, all channels are connected to a 3 V signal, as shown below:



After the wiring is complete, execute the pac_i8012W_Basic_Info.exe demo.



Select the slot index where the I-9012 module is connected and select the range, then click the “Read AI” button. Note that the slot index starts from 1 on the XP-9000 series host and starts from 0 on the WP-9000 series host.



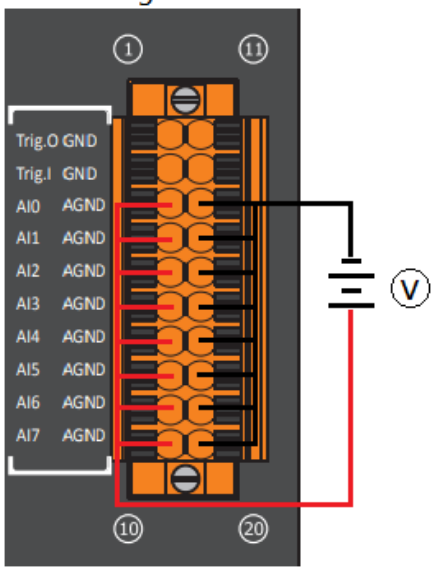
Note also that the APIs provide both a float and a HEX data format for the read function.

2.2. Magic Scan

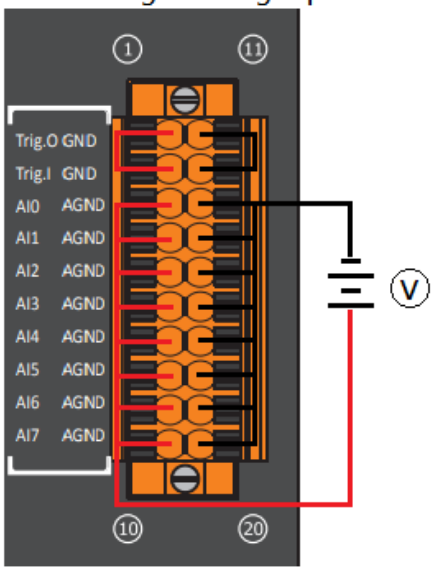
The Magic Scan function is used to sample data at high speed and can be triggered by either an internal clock or an external clock. The external clock can be either the Trig.O pin or another device, such as a waveform generator. The read data can be in block mode, non-block mode or ISR mode.

Depending on the different applications, the wiring method will also be different. The images below illustrate the different wiring methods.

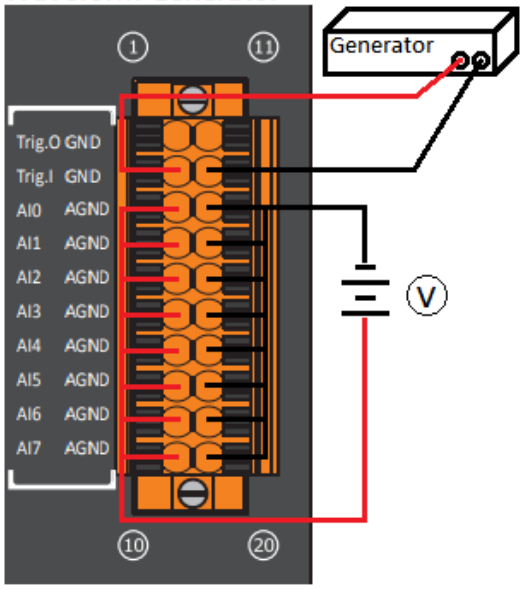
Internal Trig



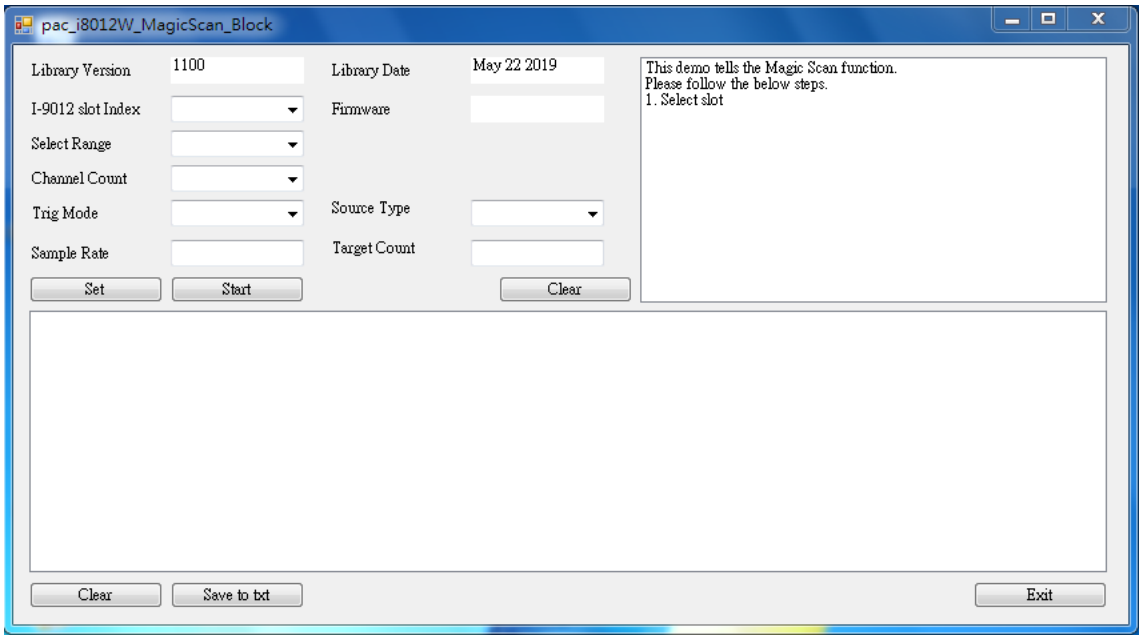
External Trig with Trig.O pin



External Trig with
Waveform Generator

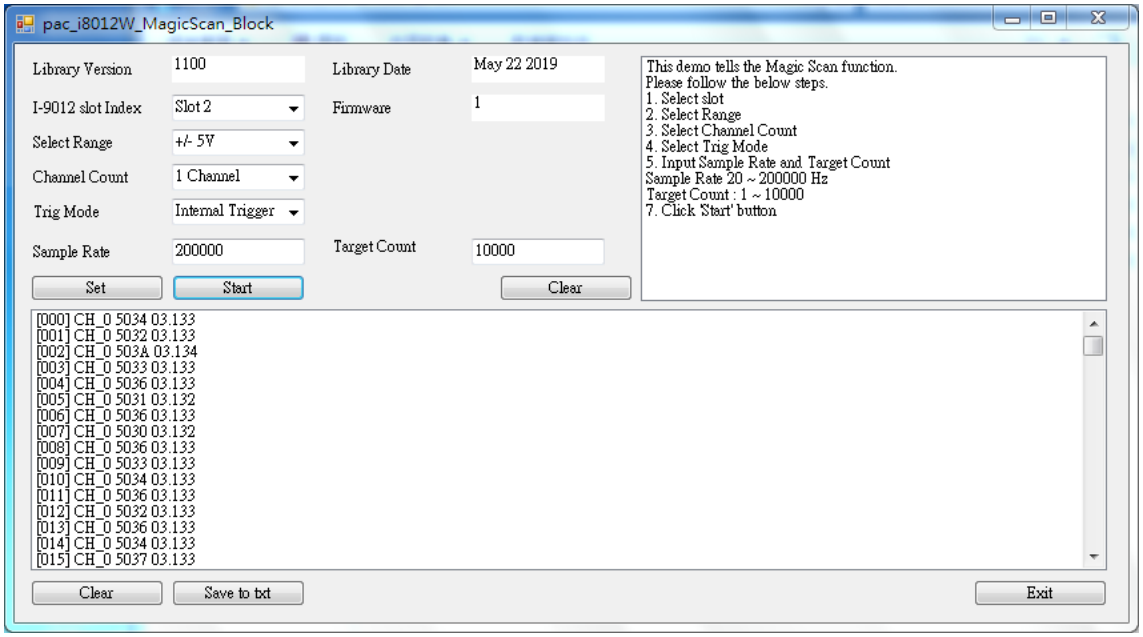


In this case, all channels are connected to a 3 V signal and are triggered using the internal clock. The data is then read in block mode. After the wiring is complete, execute the pac_i8012W_MagicScan_Block.exe demo.

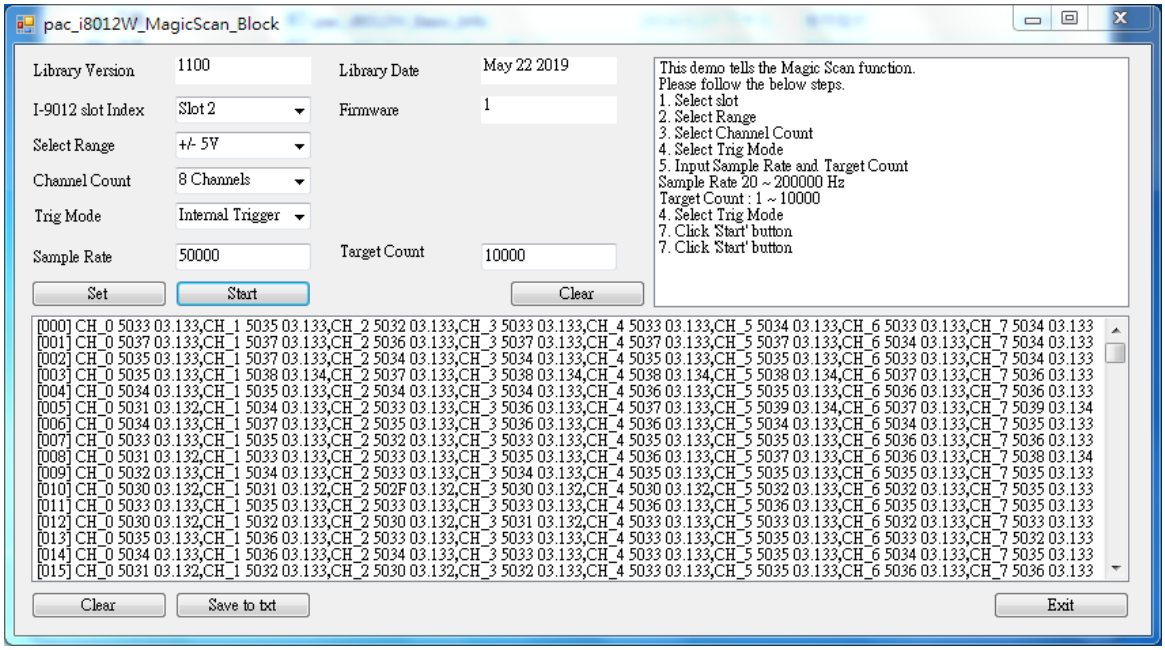


Select the slot index, range, channel count, trig mode and source type, and then enter the input sample rate and the target count. Note that the sample rate is affected by the channel count and the program process. Click the “Set” button and then click the “Start” button.

The image below shows the result of the Magic Scan at 200 kHz for 1 channel.



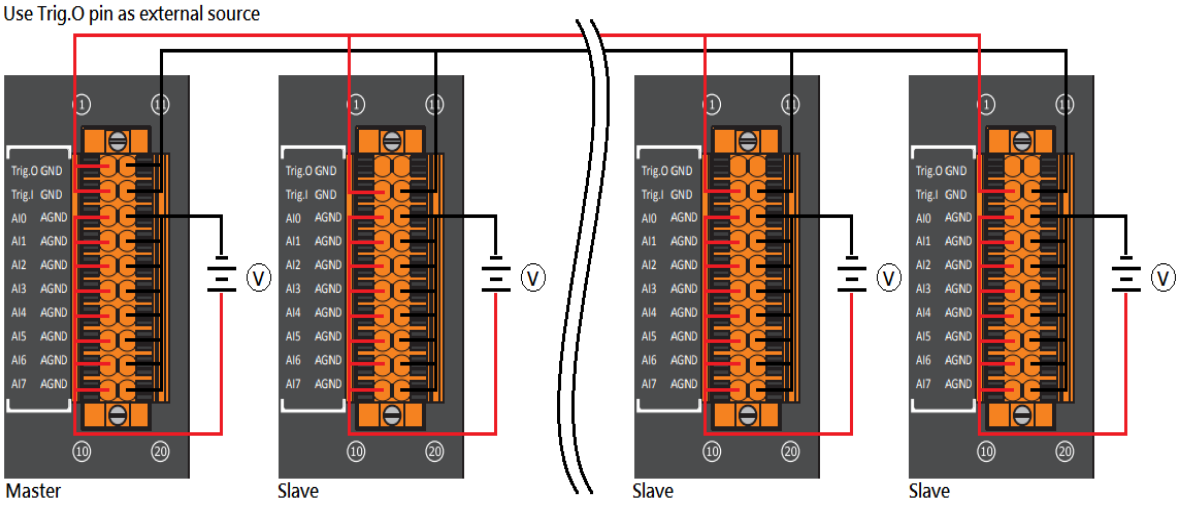
The image below shows the result of the Magic Scan at 50 kHz for 8 channels.



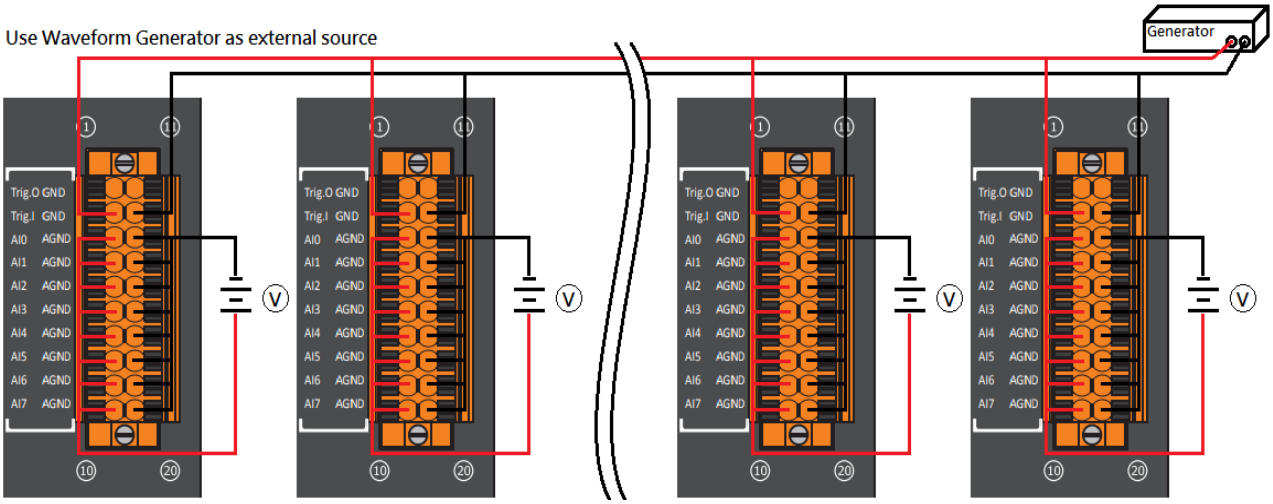
2.3. Multiple Module Scan

This function is used to instruct multiple I-9012 modules to simultaneously sample data and can only be triggered by an external clock, which can be the Trig.O pin or another device, such as a waveform generator, and data is read in non-block mode.

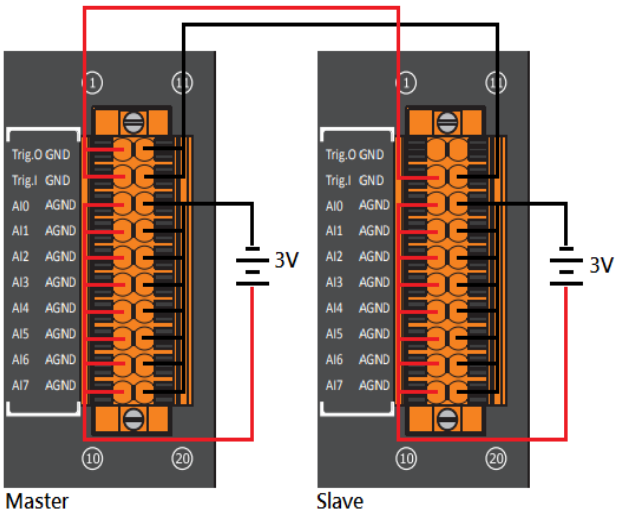
Depending on the different application, the wiring method will also different. The images below illustrate the different wiring methods.



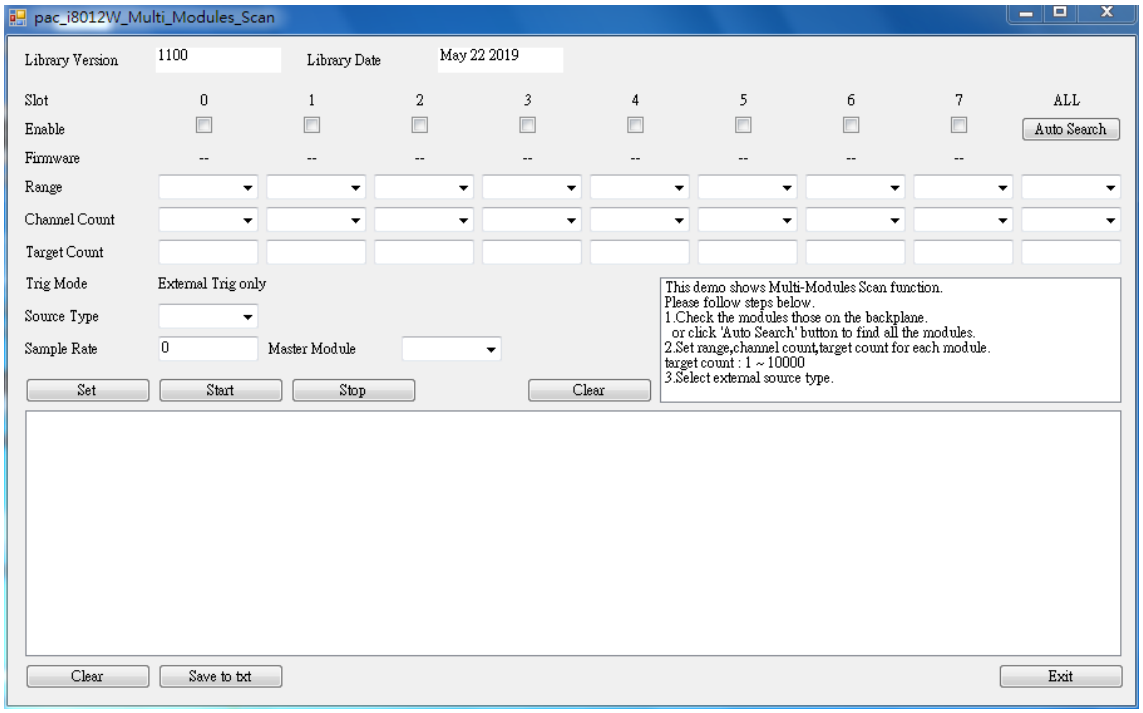
Note that if you are using the Trig.O pin as an external source, you will need to set one of the I-9012 modules as the master device.



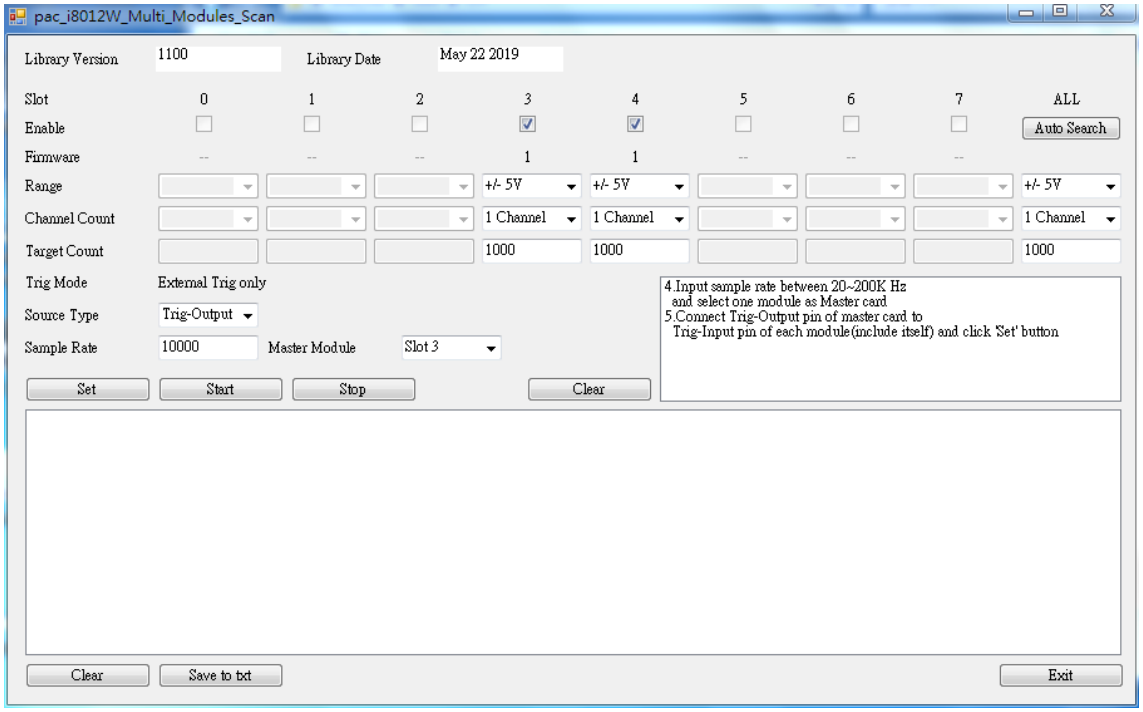
In this case, there are two I-9012 modules, where all the channels on both modules are connected to a 3 V signal and are triggered using the Trig.O pin. The data is then read in non-block mode, as shown below:



After the wiring is complete, execute the pac_i8012W_Multi_Modules_Scan.exe demo.

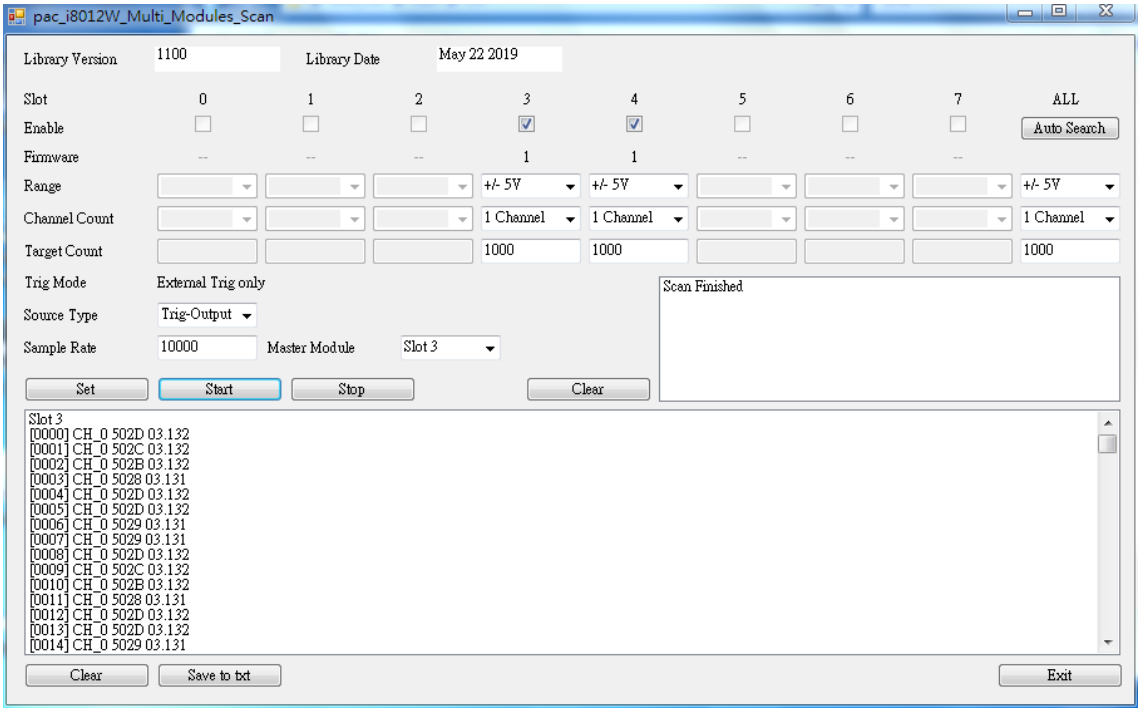


Check the checkbox for the index of the slot where the I-9012 module is inserted, select the range and the channel count, enter the target count, select the source type and the input sample rate, and then choose the master device. Note that the sample rate may be affected by the channel count and the program process.



Click the “Set” button and then click the “Start” button.

The image below shows the result of the Multi Module Scan using two modules at 10 kHz in both for 1 channel.



3. Demo Programs

ICP DAS provides a range of demo programs for different platforms that can be used to verify the functions of the I-9012 module. The source code contained in these programs can also be reused in your own custom programs if needed. The following is a list of the locations where both the demo programs and the associated libraries can be found on either the ICP DAS website or the enclosed CD.

Platform	Location
On the WP-9000	
Library	CD:\WinPAC_AM335x\wp-9000\SDK\IO_Modules ftp://ftp.icpdas.com/pub/cd/winpac_am335x/wp-9000/sdk/io_modules/
Demo	**VC2008 Demo:** CD:\WinPAC_AM335x\wp-9000\demo\PAC\Vc2008\IO\Local ftp://ftp.icpdas.com/pub/cd/winpac_am335x/wp-9000/demo/pac/vc2008/io/local/ **C# Demo:** CD:\WinPAC_AM335x\wp-9000\demo\PAC\C#\IO\Local ftp://ftp.icpdas.com/pub/cd/winpac_am335x/wp-9000/demo/pac/c%23/io/local/
On the XP-9000-WES7	
Library	CD:\ippc-wes7\sdk\IO ftp://ftp.icpdas.com/pub/cd/ippc-wes7/sdk/io/
Demo	**VC Demo:** CD:\ ippc-wes7\demo\pacsdk\vc\io\local ftp://ftp.icpdas.com/pub/cd/ippc-wes7/demo/pacsdk/vc/io/local/ **C# Demo:** CD:\ ippc-wes7\demo\pacsdk\csharp.net\io\local ftp://ftp.icpdas.com/pub/cd/ippc-wes7/demo/pacsdk/csharp.net/io/local/

4. API References

ICPDAS supplies a range of C/C++ API functions for the I-9012 module. When developing a custom program, refer to either the pac_i8012W.h header file or the API functions described in the following sections for more detailed information.

ICPDAS also supplies a range of C# functions that can be used to develop custom .NET programs. These functions are ported from the relevant C/C++ functions. For more information related to.NET functions, refer to either the pac_i8012W.h file or the following sections.

API Naming Table

The following table describes the platforms and which the product series it is included together with the different part of the function name.

Platform	Product included	API prefix characters
CE7	WP-9000-CE7 series	"pac_i8012W_ "+ function name
WES7	XP-9000-WES7 series	"pac_i8012W_ "+ function name

The following is an overview of the functions provided in the pac_i8012W.lib for use with Windows (CE and WES) platforms.

Function for Windows	Description
pac_i8012W_Init	This function is used to check the hardware ID for the I-9012 module and initialize the module.
pac_i8012W_GetLibVersion	This function is used to read the version information for the libraries.
pac_i8012W_GetLibDate	This function is used to read the build date for the libraries.
pac_i8012W_GetFirmwareVersion	This function is used to read the FPGA version from the I-9012 module.
pac_i8012W_Read_AI	This function is used to read the Analog Input information from a specific channel in float format.
pac_i8012W_Read_AIHex	This function is used to read the Analog Input information from a specific channel in HEX format.
pac_i8012W_ConfigMagicScan	This function is used to configure the Magic Scan function on an I-9012 module.
pac_i8012W_StartMagicScan	This function is used to start the Magic Scan function.
pac_i8012W_StopMagicScan	This function is used to stop the Magic Scan function.
pac_i8012W_Read_FIFO_Block	This function is used to read the FIFO in block mode.
pac_i8012W_Read_FIFO_NonBlock	This function is used to read the FIFO in non-block mode.
pac_i8012W_InstallMagicScanISR	This function is used to install the ISR function for the Magic Scan function.
pac_i8012W_UnInstallMagicScanISR	This function is used to uninstall the ISR function for the Magic Scan function.
pac_i8012W_ClearINT	This function is used to clear the interrupt signal.
pac_i8012W_ReadFIFO_ISR	This function is used to read the FIFO in the ISR function.
pac_i8012W_ConfigTrigOut	This function is used to set the frequency of the Trig.O output.

Function for Windows	Description
pac_i8012W_Enable_TrigOut	This function is used to start the output on the Trig.O pin.
pac_i8012W_Disable_TrigOut	This function is used to stop the output on the Trig.O pin
pac_i8012W_ReadGainOffset	This function is used to read the gain and the offset for a specific channel.
pac_i8012W_CalibrationHEX	This function is used to calibrate the data in HEX format.
pac_i8012W_CalibrationFloat	This function is used to calibrate the data in float format.

4.1. pac_i8012W_Init

This function is used to check the hardware ID for the I-9012 module and initialize the module.

Syntax

```
short pac_i8012W_Init (int slot);
```

Parameters

slot: The specified slot number (0 - 7).

Return Values

0: The module in the specific slot is an I-9012 module.

-1: There is no I-9012 module in the slot.

Note

This function must be called once before using any APIs for the I-9012 module. If there are two or more modules installed in the host, this API needs to be called for each I-9012 module using the corresponding slot number.

Examples

[C/C++]

```
int _tmain(int argc, _TCHAR* argv[]){
    int slot=-1,i=0;
    for(i=0;i<8;i++){       // Search every slot
        if(pac_i8012W_Init(i)==0){     // Find the first I-9012 module
            slot=i;
            break;
        }
    }
    return 0;
}
```

[C#]

```
static void Main(string[] args){
    int slot=-1,i = 0;
    for (i = 0; i < 8; i++){        // Search every slot
        if (pac8012WNet.pac8012W.Init(i) == 0){      // Find the first I-9012 module
            slot = i;
            break;
        }
    }
}
```

4.2. pac_i8012W_GetLibVersion

This function is used to read the version number for the pac_i8012W.lib and pac_i8012W.dll libraries.

Syntax

```
short pac_i8012W_GetLibVersion(void);
```

Parameters

slot: The specified slot number (0 - 7).

Return Values

The version number for the libraries. E.g., 0x1100 indicates version 1.1.0.0

Examples

[C/C++]

```
int _tmain(int argc, _TCHAR* argv[]){
    short ver=0;
    ver=pac_i8012W_GetLibVersion();
    return 0;
}
```

[C#]

```
static void Main(string[] args){
    short LibVer = 0;
    LibVer = pac8012WNet.pac8012W.LibVersion();
}
```

4.3. pac_i8012W_GetLibDate

This function is used to read the build date for the pac_i8012W.lib and pac_i8012W.dll libraries.

Syntax

```
void pac_i8012W_GetLibDate(char libDate[]);
```

Parameters

libDate[]: The build date for the libraries.

Return Values

None

Examples

[C/C++]

```
int _tmain(int argc, _TCHAR* argv[]){
    char libDate[20];
    pac_i8012W_GetLibDate(libDate);
    return 0;
}
```

[C#]

```
static void Main(string[] args){
        string LibDate = "";
        LibDate = pac8012WNet.pac8012W.LibDate();
}
```

4.4. pac_i8012W_GetFirmwareVersion

This function is used to read the FPGA version number from the I-9012 module.

Syntax

```
short pac_i8012W_GetFirmwareVersion(int slot);
```

Parameters

slot: *The specified slot number (0 - 7).*

Return Values

The version number for the FPGA.

Examples

[C++]

```
int _tmain(int argc, _TCHAR* argv[]){
    int slot=1;
    short ver=0;
    pac_i8012W_Init(slot);   // Initialize
    ver=pac_i8012W_GetFirmwareVersion(slot);
    return 0;
}
```

[C#]

```
static void Main(string[] args){
        int slot = 1;
        short FarmVer = 0;
        pac8012WNet.pac8012W.Init(slot);   // Initialize
        FarmVer = pac8012WNet.pac8012W.FirmwareVersion(slot);
}
```

4.5. pac_i8012W_Read_AI

This function is used to read the Analog Input from a specific channel in float format.

Syntax

```
int pac_i8012W_Read_AI
(
    int slot,
    int ch,
    int gain,
    float* fval
);
```

Parameters

slot: The specified slot number (0 - 7).

ch: The specified channel (0 - 7).

gain: The specified input range (0 - 1)

0: +/-5 V

1: +/-10 V

**fval*: [Output] The input data in float format.

Return Values

Refer to the Error Codes listed in Appendix A.

Examples

[C/C++]

```
int _tmain(int argc, _TCHAR* argv[]){
    int slot=1,ch=0,gain=0;
    float fval=0;
    pac_i8012W_Init(slot);   // Initialize
    pac_i8012W_Read_AI(slot,ch,gain,&fval);
    return 0;
}
```

[C#]

```
static void Main(string[] args){
        int slot = 1, ch = 0, gain = 0;
        float fval = 0;
        pac8012WNet.pac8012W.Init(slot);   // Initialize
        pac8012WNet.pac8012W.ReadAI(slot, ch, gain, ref fval);
}
```

4.6. pac_i8012W_Read_AIHex

This function is used to read the Analog Input from a specific channel in Hexadecimal format.

Syntax

```
int pac_i8012W_Read_AIHex
(
    int slot,
    int ch,
    int gain,
    short* hval
);
```

Parameters

slot: The specified slot number (0 - 7).

ch: The specified channel (0 - 7).

gain: The specified input range (0 - 1)

0: +/-5 V

1: +/-10 V

**hval:* [Output] The data in Hexadecimal format.

Return Values

Refer to the Error Codes listed in Appendix A.

Examples

[C/C++]

```
int _tmain(int argc, _TCHAR* argv[]){
    int slot=1,ch=0,gain=0;
    short hval=0;
    pac_i8012W_Init(slot);   // Initialize
    pac_i8012W_Read_AIHex(slot,ch,gain,&hval);
    return 0;
}
```

[C#]

```
static void Main(string[] args){
        int slot = 1, ch = 0, gain = 0;
        short hval = 0;
        pac8012WNet.pac8012W.Init(slot);   // Initialize
        pac8012WNet.pac8012W.ReadAIHex(slot, ch, gain, ref hval);
}
```

4.7. pac_i8012W_ConfigMagicScan

This function is used to configure the Magic Scan function on the I-9012 module.

Syntax

```
int pac_i8012W_ConfigMagicScan
(
    int slot,
    int gain,
    int chcnt,
    int type,
    unsigned long samplerate,
    unsigned long* realrate
);
```

Parameters

slot: The specified slot number (0 - 7).

gain: The specified input range (0 - 1)

0: +/-5 V

1: +/-10 V

chcnt: The number of channels used for the scan. (1 - 8)

E.g.: when chcnt is set to 1, Magic Scan will scan channel 0.

When chcnt is set to 2, Magic Scan will scan both channel 0 and channel 1.

type: The trigger mode for the Magic Scan (0 - 1)

0: Internal trigger.

1: External trigger.

samplerate: The sampling rate for Magic Scan for an internal trigger (20 - 200000)

* realrate: [Output]The real sampling rate.

The sampling rate is calculated by dividing the basic clock at a 16-bit resolution.

Return Values

Refer to Error Codes listed in Appendix A.

Examples

Refer to the demos for Magic Scan.

4.8. pac_i8012W_StartMagicScan

This function is used to start the Magic Scan function.

Syntax

```
int pac_i8012W_StartMagicScan(int slot);
```

Parameters

slot: The Specified slot number (0 - 7).

Return Values

Refer to the Error Codes listed in Appendix A.

Examples

Refer to the demos for Magic Scan.

4.9. pac_i8012W_StopMagicScan

This function is used to stop the Magic Scan function.

Syntax

```
int pac_i8012W_StopMagicScan(int slot);
```

Parameters

slot: The Specified slot number (0 - 7).

Return Values

Refer to the Error Codes listed in Appendix A.

Examples

Refer to the demos for Magic Scan.

4.10. pac_i8012W_Read_FIFO_Block

This function is used to read data from the FIFO in block mode.

Syntax

```
int pac_i8012W_Read_FIFO_Block
(
    int slot,
    short FIFOData[],
    unsigned long cnt,
    unsigned long* readFIFOcnt
);
```

Parameters

slot:	The specified slot number (0 - 7).
FIFOData[]:	The specified short format array used to store the data that is read from the FIFO.
cnt:	The amount of data required.
**readFIFOcnt:*	[Output] The total count for the data read in this process.

Return Values

Refer to the Error Codes listed in Appendix A.

Examples

Refer to the demo for Magic Scan Block mode.

4.11. pac_i8012W_Read_FIFO_NonBlock

This function is used to read data from the FIFO in non-block mode.

Syntax

For Windows (CE and WES)

```
int pac_i8012W_Read_FIFO_NonBlock
(
    int slot,
    short FIFOData[],
    unsigned long cnt,
    short* readFIFOcnt
);
```

Parameters

slot: The specified slot number (0 - 7).

FIFOData[]: The specified short format array used to store the data that is read from the FIFO.

cnt: The amount of data required.

**readFIFOcnt*: [Output] The total count for the data read in this process.

Return Values

Refer to the Error Codes listed in Appendix A.

Examples

Refer to the demo for Magic Scan Non-Block mode.

4.12. pac_i8012W_InstallMagicScanISR

This function is used to install the ISR function for the Magic Scan function.

Syntax

For Windows (WES)

```
short pac_i8012W_InstallMagicScanISR
(
   int slot,
   void (__stdcall *isr)(int slot),
   short leveltype
);
```

For Windows (CE)

```
short pac_i8012W_InstallMagicScanISR
(
   int slot,
   void (*isr)(int),
   short leveltype
);
```

Parameters

slot: The specifiedc slot number (0 - 7).

*void (__stdcall *isr)(int slot)/void (*isr)(int)*:

The function pointer passed for the ISR.

leveltype:

The interrupt trigger condition (0 - 1) based on the amount of data in the FIFO buffer. When this amount is greater than the value defined by the leveltype parameter, the interrupt will be triggered and the ISR will be executed to handle the interrupt event.

0: 10

1: 2048

Return Values

Refer to the Error Codes listed in Appendix A.

Examples

Refer to the demo for Magic Scan ISR mode.

4.13. pac_i8012W_UnInstallMagicScanISR

This function is used to uninstall the ISR function from the Magic Scan function.

Syntax

```
short pac_i8012W_UnInstallMagicScanISR(int slot);
```

Parameters

slot: The specified slot number (0 - 7).

Return Values

Refer to the Error Codes listed in Appendix A.

Examples

Refer to the demo for Magic Scan ISR mode.

4.14. pac_i8012W_ClearINT

This function is used to clear the interrupt signal.

Syntax

```
short pac_i8012W_ClearINT(int slot);
```

Parameters

slot: *The Specified slot number (0 - 7).*

Return Values

Refer to the Error Codes listed in Appendix A.

Examples

Refer to the demo for Magic Scan ISR mode.

4.15. pac_i8012W_ReadFIFO_ISR

This function is used to read data from the FIFO in ISR mode.

Syntax

```
short pac_i8012W_ReadFIFO_ISR
(
    int slot,
    short FIFOData[],
    short* readFIFOcnt
);
```

Parameters

slot: The specified slot number (0 - 7).

FIFOData[]: The specified short format array used to store the data that is read from the FIFO.

**readFIFOcnt*: [Output] The total count for the data read in this process

Return Values

Refer to the Error Codes listed in Appendix A.

Examples

Refer to the demo for Magic Scan ISR mode.

4.16. pac_i8012W_ConfigTrigOut

This function is used to set the frequency for the Trig.O output.

Syntax

```
int pac_i8012W_ConfigTrigOut
(
    int slot,
    unsigned long freq,
    unsigned long* realfreq
);
```

Parameters

slot:	The specified slot number (0 - 7).
freq:	The frequency for the Trig.O output (20 - 200000)
** realfreq*:	[Output] The real frequency. The frequency is calculated by dividing the basic clock by the 16-bit resolution.

Return Values

Refer to the Error Codes listed in Appendix A.

Examples

[C/C++]

```
int _tmain(int argc, _TCHAR* argv[]){
    int slot=1;
    unsigned long freq=20000,realfreq=0;
    pac_i8012W_Init(slot);
    pac_i8012W_ConfigTrigOut(slot,freq,&realfreq);
    return 0;
}
```

[C#]

```
static void Main(string[] args){
        int slot = 1;
        uint freq = 20000, realfreq=0;
        pac8012WNet.pac8012W.Init(slot);
        pac8012WNet.pac8012W.ConfigTrigOut(slot, freq, ref realfreq);
}
```

4.17. pac_i8012W_Enable_TrigOut

This function is used to start the output for the Trig.O pin.

Syntax

```
short pac_i8012W_Enable_TrigOut(int slot);
```

Parameters

slot: The specified slot number (0 - 7).

Return Values

Refer to the Error Codes listed in Appendix A.

Examples

[C/C++]

```
int _tmain(int argc, _TCHAR* argv[]){
    int slot=1;
    unsigned long freq=20000,realfreq=0;
    pac_i8012W_Init(slot);
    pac_i8012W_ConfigTrigOut(slot,freq,&realfreq);
    pac_i8012W_Enable_TrigOut(slot);
    return 0;
}
```

[C#]

```
static void Main(string[] args){
        int slot = 1;
        uint freq = 20000, realfreq=0;
        pac8012WNet.pac8012W.Init(slot);
        pac8012WNet.pac8012W.ConfigTrigOut(slot, freq, ref realfreq);
        pac8012WNet.pac8012W.EnableTrigOut(slot);
}
```

4.18. pac_i8012W_Disable_TrigOut

This function is used to stop the output of the Trig.O pin.

Syntax

```
short pac_i8012W_Disable_TrigOut(int slot);
```

Parameters

slot: The specified slot number (0 - 7).

Return Values

Refer to the Error Codes listed in Appendix A.

Examples

[C/C++]

```
int _tmain(int argc, _TCHAR* argv[]){
    int slot=1;
    unsigned long freq=20000,realfreq=0;
    pac_i8012W_Init(slot);
    pac_i8012W_ConfigTrigOut(slot,freq,&realfreq);
    pac_i8012W_Disable_TrigOut (slot);
    return 0;
}
```

[C#]

```
static void Main(string[] args){
        int slot = 1;
        uint freq = 20000, realfreq=0;
        pac8012WNet.pac8012W.Init(slot);
        pac8012WNet.pac8012W.ConfigTrigOut(slot, freq, ref realfreq);
        pac8012WNet.pac8012W.DisableTrigOut(slot);
}
```

4.19. pac_i8012W_ReadGainOffset

This function is used to read the calibrated gain value and the offset value.

Syntax

```
short pac_i8012W_ReadGainOffset
(
    int slot,
    int ch,
    int gain,
    unsigned short* GainValue,
    short* offsetValue
);
```

Parameters

slot: The specified slot number (0 - 7).

ch: The specified channel (0 - 7).

gain: The specified input range (0 - 1)

0: +/-5 V

1: +/-10 V

**GainValue*: [Output] The gain value for the input range of the specified channel.

**offsetValue*: [Output] The offset value for the input range of the specified channel.

Return Values

Refer to the Error Codes listed in Appendix A.

Examples

[C/C++]

```
int _tmain(int argc, _TCHAR* argv[]){
    int slot=1,gain=0,ch=0;
    unsigned short GainValue=0;
    short offsetValue=0;
    pac_i8012W_Init(slot);
    for(gain=0;gain<2;gain++)
          for(ch=0;ch<8;ch++)
                pac_i8012W_ReadGainOffset(slot,ch,gain,&GainValue,&offsetValue);
    return 0;
}
```

[C#]

```
static void Main(string[] args){
        int slot = 1, gain = 0, ch = 0;
        ushort gainval = 0;
        short offset = 0;
        pac8012WNet.pac8012W.Init(slot);
        for (gain = 0; gain < 2; gain++)
              for (ch = 0; ch < 8;ch++ )
             pac8012WNet.pac8012W.ReadGainOffset(slot, ch, gain, ref gainval, ref offsetval);
}
```

4.20. pac_i8012W_CalibrationHEX

This function is used to calibrate the data that is read from the FIFO.

Syntax

```
int pac_i8012W_CalibrationHEX
(
    int slot,
    int ch,
    int gain,
    short raw,
    short* hval
);
```

Parameters

slot: The specified slot number (0 - 7).

ch: The specified channel (0 - 7).

gain: The specified input range (0 - 1)

0: +/-5 V

1: +/-10 V

raw: The uncalibrated data that is read from the FIFO.

*hval: [Output] The calibrated data in hexadecimal format.

Return Values

Refer to the Error Codes listed in Appendix A.

Examples

[C/C++]

```
int _tmain(int argc, _TCHAR* argv[]){
    int slot=1,ch=0,gain=0,i=0;
    short raw[10],hval[10];
    pac_i8012W_Init(slot);
    ...
    //raw[10] stores the data that is read from the FIFO.
    ...
    for(i=0;i<10;i++)
          pac_i8012W_CalibrationHEX(slot,ch,gain,raw[i],&hval[i]);
    return 0;
}
```

[C#]

```
static void Main(string[] args){
        int slot = 1, ch = 0, gain = 0,i = 0;
        short[] raw = new short[10];
        short[] hval = new short[10];
        pac8012WNet.pac8012W.Init(slot);
        ...
        //raw[10] stores the data that is read from the FIFO.
        ...
        for (i=0;i<10;i++)
              pac8012WNet.pac8012W.CalibrationHEX(slot,ch,gain,raw[i],ref hval[i]);
}
```

4.21. pac_i8012W_CalibrationFloat

This function is used to calibrate the data that is read from the FIFO.

Syntax

```
int pac_i8012W_CalibrationFloat(int slot,int ch,int gain,short raw,float* fval);
```

Parameters

slot: The specified slot number (0 - 7).

ch: The specified channel (0 - 7).

gain: The specified input range (0 - 1)

0: +/- 5V

1: +/- 10V

raw: The uncalibrated data that is read from the FIFO.

**fval*: [Output] The calibrated data in float format.

Return Values

Refer to the Error Codes listed in Appendix A.

Examples

[C/C++]

```
int _tmain(int argc, _TCHAR* argv[]){
    int slot=1,ch=0,gain=0,i=0;
    short raw[10]
    float fval[10];
    pac_i8012W_Init(slot);
    ...
    //raw[10] stores the data that is read from the FIFO.
    ...
    for(i=0;i<10;i++)
          pac_i8012W_CalibrationFloat(slot,ch,gain,raw[i],&fval[i]);
    return 0;
}
```

[C#]

```
static void Main(string[] args){
       int slot = 1, ch = 0, gain = 0,i = 0;
       short[] raw = new short[10];
       float[] fval = new float[10];
       pac8012WNet.pac8012W.Init(slot);
       ...
       //raw[10] stores the data that is read from the FIFO.
       ...
       for (i=0;i<10;i++)
             pac8012WNet.pac8012W.CalibrationFloat (slot,ch,gain,raw[i],ref fval[i]);
}
```

Appendix A. Error Codes

Error Code	Definition	Description
0	NoError	This indicates that there have been no errors
-1	ID_ERROR	There was a problem with the module ID
-2	FIFO_FULL	The FIFO is full, or the data is wrong.
-3	FIFO_EMPTY	The FIFO is empty, or there is no data in the FIFO.

Appendix B. Revision History

This chapter provides revision history information to this document.

The table below shows the revision history.

Revision	Date	Description
1.0.0	May 2019	Initial issue