DAQ-MTLB

Data Acquisition Toolbox Adapter

User's Manual

Manual Rev. 2.00

Revision Date: December 12, 2005

Part No: 50-11216-1000

Recycled Paper

Advance Technologies; Automate the World.

The information in this document is subject to change without prior notice in order to improve reliability, design, and function and does not represent a commitment on the part of the manufacturer.

In no event will the manufacturer be liable for direct, indirect, special, incidental, or consequential damages arising out of the use or inability to use the product or documentation, even if advised of the possibility of such damages.

Trademarks

NuDAQ, NuIPC, DAQBench are registered trademarks of ADLINK TECHNOLOGY INC.

Product names mentioned herein are used for identification purposes only and may be trademarks and/or registered trademarks of their respective companies.

Getting Service from ADLINK

Customer Satisfaction is top priority for ADLINK Technology Inc. Please contact us should you require any service or assistance.

ADLINK TECHNOLOGY INC.

Web Site: http://www.adlinktech.com

Sales & Service: Service@adlinktech.com

TEL: +886-2-82265877

FAX: +886-2-82265717

Address: 9F, No. 166, Jian Yi Road, Chungho City, Taipei, 235 Taiwan

Please email or FAX this completed service form for prompt and satisfactory service.

Company Information	
Company/Organization	
Contact Person	
E-mail Address	
Address	
Country	
TEL	FAX:
Web Site	
Product Information	
Product Model	
Environment	OS: M/B: CPU: Chipset: BIOS:

Please give a detailed description of the problem(s):

Table of Contents

List of Tables

List of Figures

1 Introduction

The DAQ-MTLB Adapter is a standard MATLAB Data Acquisition Toolbox Adaptor driver supporting the entire line of data acquisition cards (DAQs) from ADLINK. DAQ-MTLB is based on the MATLAB DAQ Toolbox interface as the groundwork for driver development. Right after installation, standard MATLAB DAQ Toolbox operations can easily control ADLINK DAQs, performing all AI, AO, and DIO functions.

For those already familiar with MATLAB DAQ Toolbox, controlling an ADLINK DAQ will be second nature as the exact same standard commands are used. For those who have never used MATLAB DAQ Toolbox, don¡¦t worry. This manual and included examples will quickly get you up to speed on basic MATLAB DAQ Toolbox operations. For a more detailed explanation, please review the Data Acquisition Toolbox section of the MATLAB manual.

1.1 System Requirements

Before using DAQ-MTLB, please ensure that your system has the following:

- An ADLINK DAQ. For a list of DAQ-MTLB supported DAQs, please see Section 2.2
- MATLAB 6.5 or above
- MATLAB Data Acquisition Toolbox 2.2 or above

1.2 MATLAB DAQ Toolbox Command List

This section contains a simple list of MATLAB DAQ Toolbox commands.

1.2.1 General Commands

1.2.1.1 get

Description
Read DAQ object property values

Syntax
value = get(obj, property_name)

Returned Value

Value	DAQ object property values

Arguments

ob	Object number
property_name	Property name

1.2.1.2 set

Description
Set DAQ object property values

Syntax
set(obj, property_name, value)

Arguments

obj	Object number
property_name	Property name
value	Desired set DAQ object property values

1.2.1.3 start

Description
Initiates the object according to its current properties and begins executing code (used on AI and AO objects)

Syntax
start(obj)

Arguments

obj	Object number

1.2.1.4 stop

Description
Stops the program

Syntax
stop(obj)

Arguments

obj	Object number

1.2.1.5 trigger

Description
Triggered object

Syntax
trigger(obj)

Arguments

obj	Object number

Notes
Can only be used when TriggerType is set to Manual

1.2.1.6 wait

Description
Wait for the object to stop running program

Syntax
wait(obj,waittime)

Arguments

obj	Object number
waittime	Wait time, in units of seconds

Notes
If the object does not finish executing the code by the wait time, MATLAB DAQ Toolbox will generate a wait time error.

1.2.2 Analog Input Commands

1.2.2.1 analoginput

Description
Create an analog input object

Syntax
ai = analoginput('mwadlink', id)

Returned Value

ai	Analog input object

Arguments

id	Card ID

Notes
daqhwinfo('mwadlink') can be used to check the card ID of an ADLINK DAQ within the system

1.2.2.2 addchannel

Description
Adds a specific analog input channel to the analog input object

Syntax
chans = addchannel(obj, hw_ch)

Returned Value

chans	Analog input channel object

Arguments

obj	Analog input object
hw_ch	Added channel

1.2.2.3 getdata

Description
Return the data acquired from the analog input object

Syntax
data = getdata(obj)

Returned Value
data Data acquired by the object

Arguments
obj Analog input object

1.2.2.4 getsample

Description
Acquire and return signal (voltage value) on the analog input channel

Syntax
sample = getsample(obj)

Returned Value
sample Signal (voltage value) on the analog input channel

Arguments
obj Analog input object

1.2.3 Analog Output Commands

1.2.3.1 analogoutput

Description
Create an analog output object

Syntax
ao = analogoutput('mwadlink', id)

Returned Value
ao Analog input object

Arguments
id Card ID

Notes
daqhwinfo('mwadlink') can be used to check the card ID of an ADLINK DAQ within the system

1.2.3.2 addchannel

Description
Adds a specific analog input channel to the analog input object

Syntax
chans = addchannel(obj,hw_ch)

Returned Value

chans	Channel object

Arguments

obj	Analog input object
hw_ch	Added channel

1.2.3.3 putdata

Description
Download pre-set data to output to the analog output object

Syntax
putdata(obj,data)

Arguments

obj	Analog input object
data	Data to output

1.2.3.4 putsample

Description
Signal (voltage value) output from updated analog output channel

Syntax
putsample(obj,data)

Arguments

obj	Analog output object
data	Signal (voltage value) to output

1.2.4 Digital Input Output Commands

1.2.4.1 digitalio

Description
Create a digital input/output object

Syntax
dio = digitalio('mwadlink', id)

Returned Value

dio	Digital input/output object

Arguments

id	Card ID

Notes
daqhwinfo('mwadlink') can be used to check the card ID of an ADLINK DAQ within the system

1.2.4.2 addline

Add a specified digital input/output channel to the digital input/output object

Syntax
lines = addline(obj, hw_line, direction)

Returned Value

lines	Channel object

Arguments

obj	Object number
hw_line	Added channel
direction	Input/output direction of the channel ¡§in: represents digital input ¡§out: represents digital output

Notes
Due to hardware restrictions, digital input/output settings must be in units of ports. Setting a single channel to input or output will affect all other channels in that port.

1.2.4.3 getvalue

Description
Read digital input channel signal

Syntax
value = getvalue(obj)

Returned Value

value	Digital input channel signal

Arguments

obj	Digital input object

1.2.4.4 putvalue

Description
The specified value output on the digital output channel

Syntax
putvalue(obj, value)

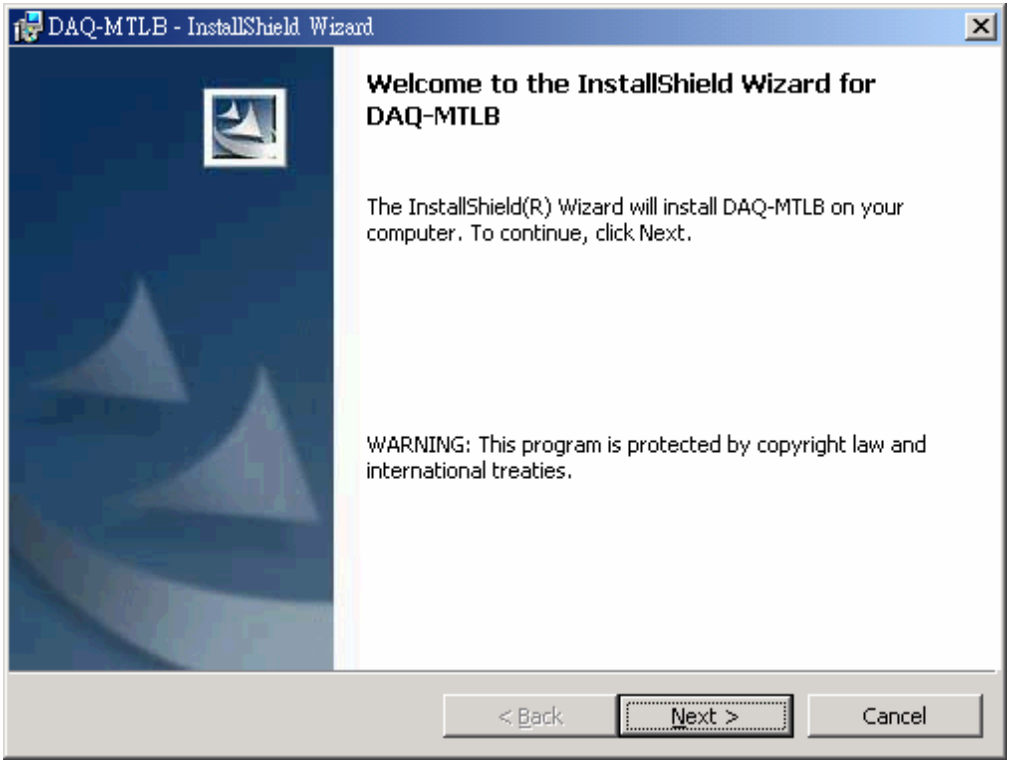
Arguments

obj	Digital input object
value	Signal to output

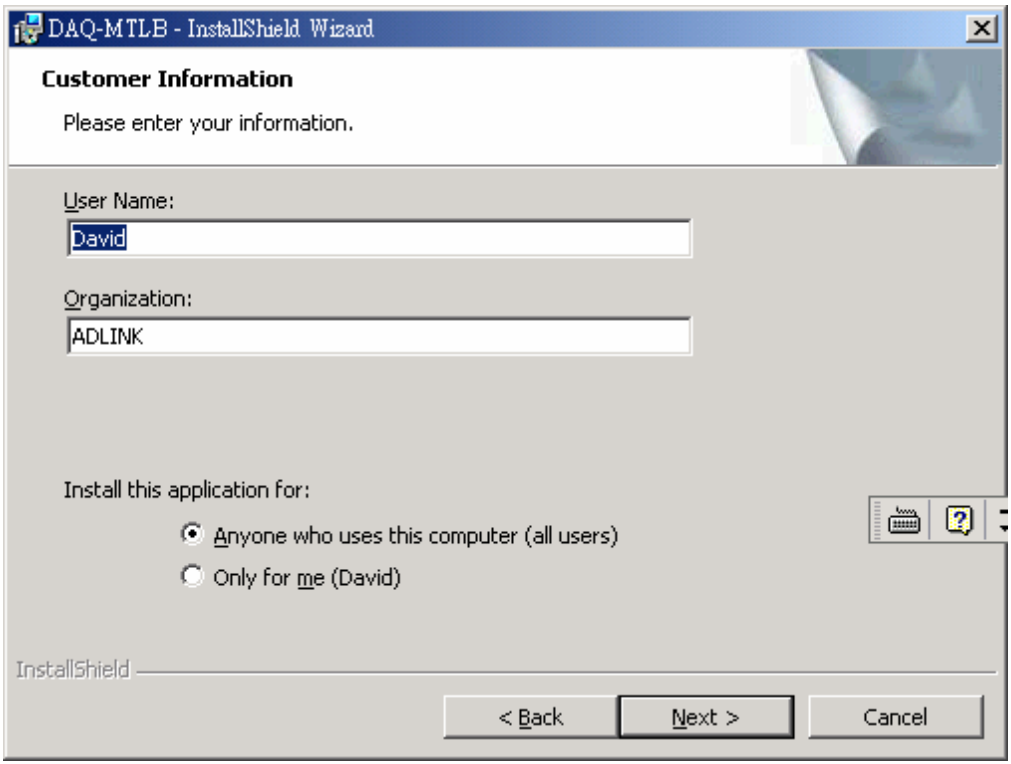
2 Installation

2.1 Installing DAQ-MTLB Adapter

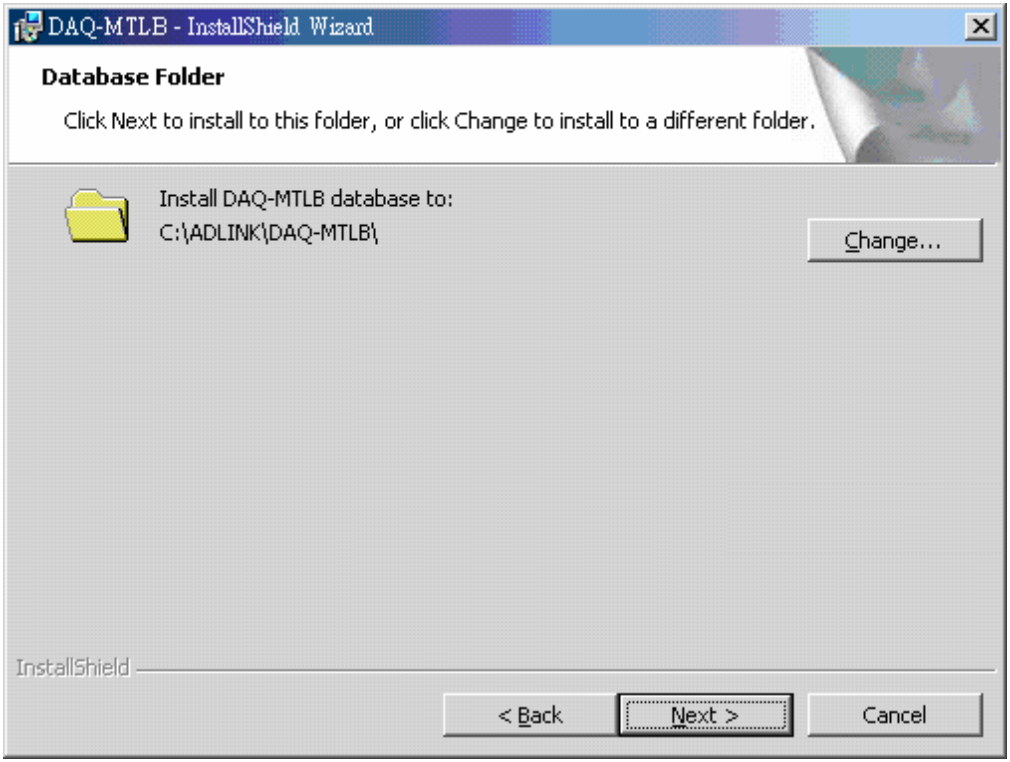
DAQ-MTLB can be found on the ADLINK website or on the All-in-One CD that is included with any product. The following steps explain how to install DAQ-MTLB.



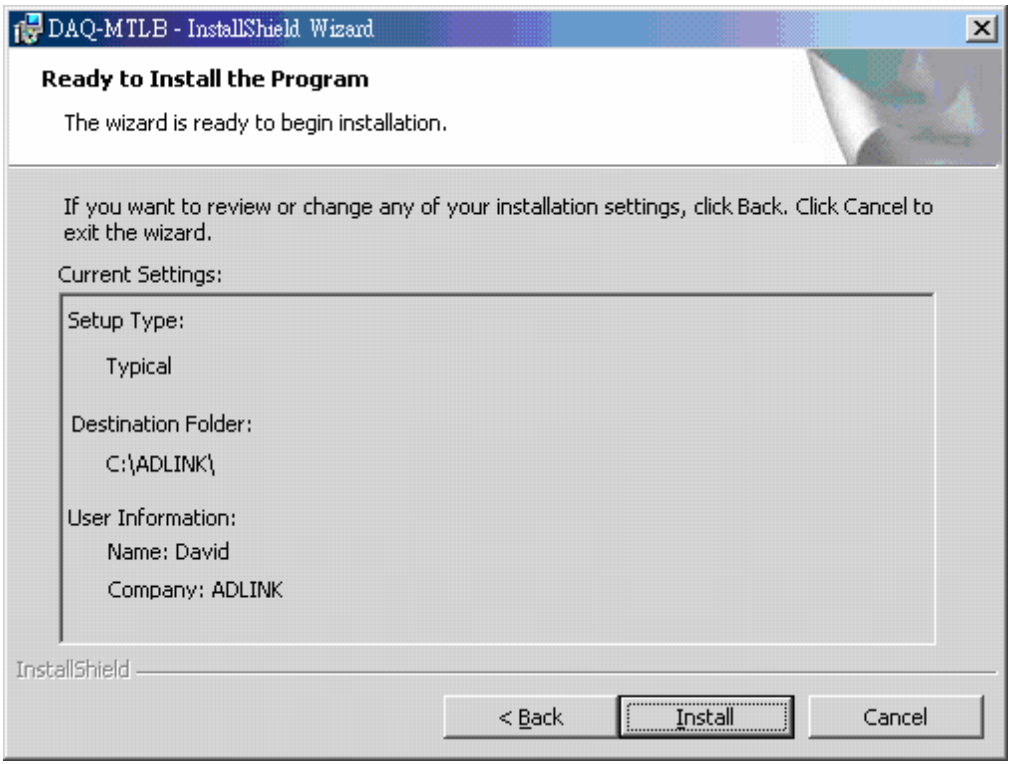
1. Execute DAQ-MTLB Setup.exe. The InstallShield Wizard will appear and guide you through the installation process.



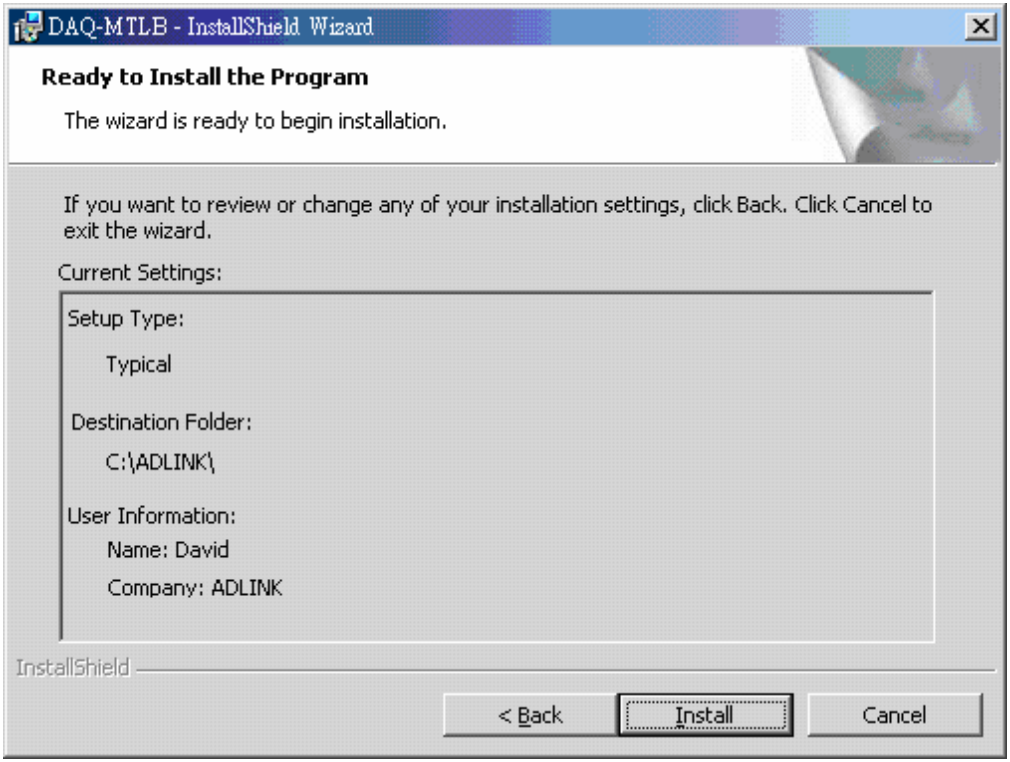
2. Click “Next” to enter the “Customer Information” window. Enter user information and click "Next".



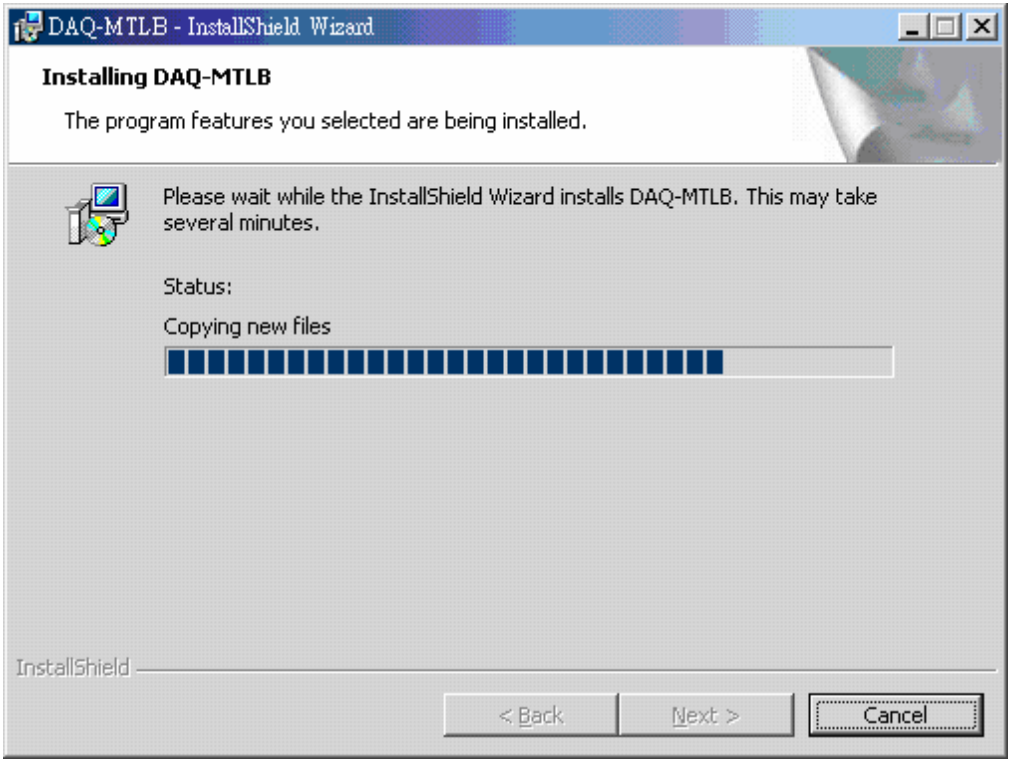
3. Select the path to install DAQ-MTLB. The default path is C:\ADLINK\DAQ-MTLB\



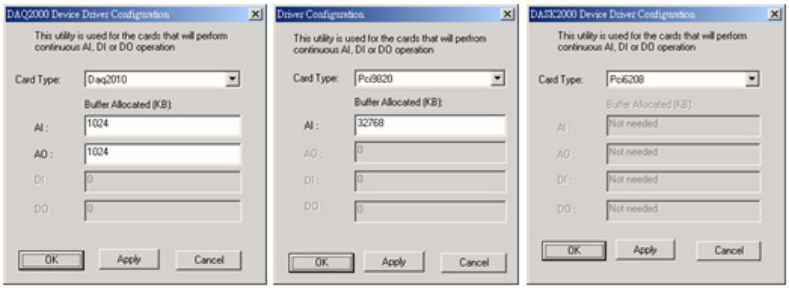
4. Confirm installation settings and click “install”

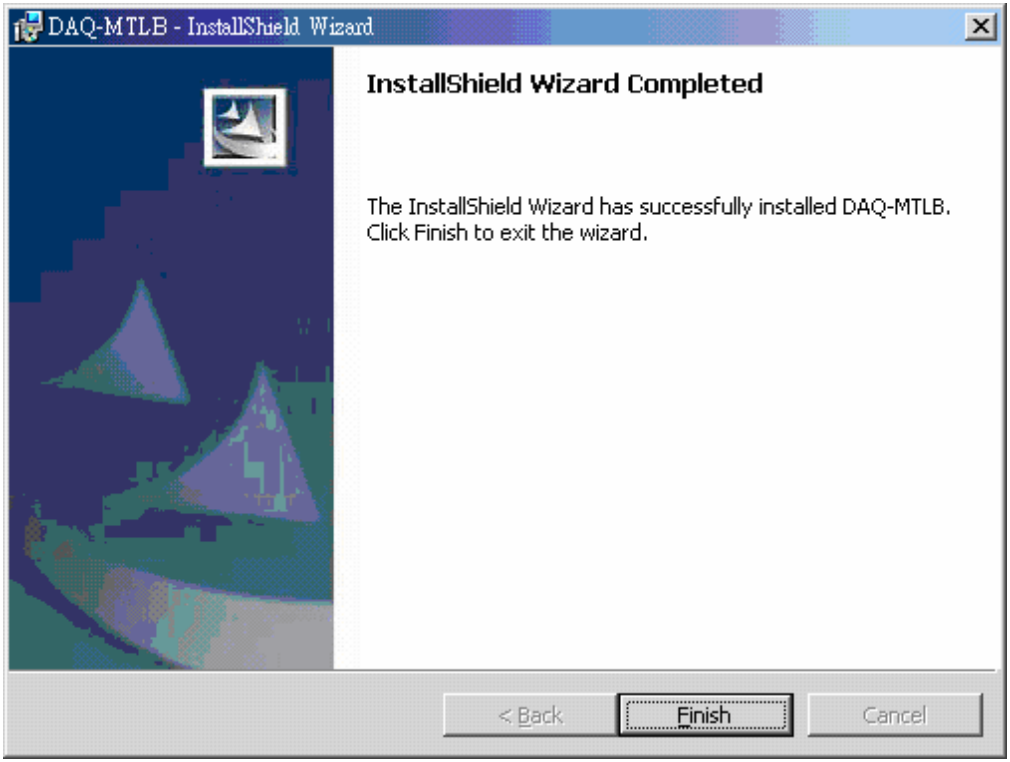


5. Confirm user settings and click “Next” to proceed to the “Installing DAQ-MTLB” window.



6. After all files are copied to your system, a "Driver Configuration" dialog box will appear to allow you to set DAQ driver memory settings. If you do not have special memory configuration requirements, just click “OK” to use the default settings.





7. Restart your computer after installation completes.

2.2 Supported DAQ List

2.2.1 Supported PCI DAQ List

DAQ	AI	AO	DIO
DAQ-2005	Y	Y	Y
DAQ-2006	Y	Y	Y
DAQ-2010	Y	Y	Y
DAQ-2204	Y	Y	Y
DAQ-2205	Y	Y	Y
DAQ-2206	Y	Y	Y
DAQ-2208	Y	-	Y
DAQ-2213	Y	-	Y
DAQ-2214	Y	Y	Y
DAQ-2501	Y	Y	Y
DAQ-2502	Y	Y	Y
PCI-6208V/A	-	Y	Y
PCI-6216V	-	Y	Y
PCI-6308V/A	-	Y	Y
PCI-7200	-	-	Y
PCI-7224	-	-	Y
PCI-7230	-	-	Y
PCI-7233	-	-	Y
PCI-7234	-	-	Y
PCI-7248	-	-	Y
PCI-7250	-	-	Y
PCI-7256	-	-	Y
PCI-7258	-	-	Y
PCI-7296	-	-	Y
PCI-7300	-	-	Y
PCI-7348	-	-	Y
PCI-7396	-	-	Y
PCI-7432	-	-	Y

Table 2-1: Supported PCI DAQ

DAQ	AI	AO	DIO
PCI-7433	-	-	Y
PCI-7434	-	-	Y
PCI-8554	-	-	Y
PCI-9111	Y	Y	Y
PCI-9112	Y	Y	Y
PCI-9113	Y	-	-
PCI-9114	Y	-	Y
PCI-9118	Y	Y	Y
PCI-9810/12	Y	-	Y
PCI-9820	-	-	-

Table 2-1: Supported PCI DAQ

2.2.2 Supported PXI DAQ List

DAQ	AI	AO	DIO
PXI-2005	Y	Y	Y
PXI -2006	Y	Y	Y
PXI -2010	Y	Y	Y
PXI -2204	Y	Y	Y
PXI -2205	Y	Y	Y
PXI -2206	Y	Y	Y
PXI -2208	Y	-	Y
PXI -2213	Y	-	Y
PXI -2214	Y	Y	Y
PXI -2501	Y	Y	Y
PXI -2502	Y	Y	Y

Table 2-2: Supported PXI DAQ

2.2.3 Supported CompactPCI DAQ List

DAQ	AI	AO	DIO
cPCI-6208V/A	-	Y	Y
cPCI-6216V	-	Y	Y
cPCI-6308V/A	-	Y	Y
cPCI-7200	-	-	Y
cPCI-7230	-	-	Y
cPCI-7248	-	-	Y
cPCI-7249	-	-	Y
cPCI-7252	-	-	Y
cPCI-7300	-	-	Y
cPCI-7432	-	-	Y
cPCI-7433	-	-	Y
cPCI-7434	-	-	Y
cPCI-8554	-	-	Y
cPCI-9112	Y	Y	Y
cPCI-9116	Y	-	Y

Table 2-3: Supported CompactPCI DAQ

3 DAQ-MTLB Examples

The examples were performed in MATLAB 7.0.4.

3.1 Search for DAQ Adaptors and Devices

This example show how to search for installed DAQ adaptors and supported DAQ-MTLB devices in the system.

```
>> hwinfo = daqhwinfo; adqpters =
     hwinfo.InstalledAdaptors %search for
     installed DAQ adaptors
adpters =
     'mwadlink'
     'parallel'
     'winsound'
>> ADLINK_INFO = daqhwinfo('mwadlink')%search
     for supported DAQ-MTLB hardware in the
     system

ADLINK_INFO =

     AdaptorDllName:
     'D:\Source\mwADLINK\ReleaseMinSize\mwADLINK
     .dll'
     AdaptorDllVersion: '1, 1, 0, 1'
          AdaptorName: 'mwadlink'
          BoardNames: {'DAQ-2010'  'PCI-9113'
     'PCI-9812'  'PCI-9111'}
     InstalledBoardIds: {'0'  '1'  '2'  '3'}
     ObjectConstructorName: {4x3 cell}
```

3.2 Analog Input

3.2.1 Capture a Single Analog Input Signal

This example shows how to use DAQ-2010 analog input channel #0 to capture a single channel analog input signal.

```
>> ai_device = analoginput('mwadlink', 0)%Opens
     the analog input functionality of device #0
     (DAQ-2010)
```

Display Summary of Analog Input (AI) Object Using 'DAQ_2010'.

```
Acquisition Parameters:
     100000 samples per second on each channel.
     100000 samples per trigger on each channel.
     1 sec. of data to be logged upon START.
     Log data to 'Memory' on trigger.

Trigger Parameters:
     1 'Immediate' trigger(s) on START.

Engine status:
     Waiting for START.
     0 samples acquired since starting.
     0 samples available for GETDATA.

AI object contains no channels.

>> ai0  =  addchannel(ai_device, 0)%Add channel
     #0 to ai_device

     Index:  ChannelName:  HwChannel:
     InputRange:  SensorRange:  UnitsRange:
     Units:
     1       ''           0          [-10 10]
     [-10 10]      [-10 10]     'Volts'

>> getsample(ai_device) %Read the voltage value
     on channel #0

ans =

     0.0835
```

3.2.2 Capture a Continuous Analog Waveform

This example shows how to use analog input channel #0 from DAQ-2010 to continuously capture a waveform.

```
>> ai_device = analoginput('mwadlink', 0)%Opens
     the analog input functionality of device #0
     (DAQ-2010)
```

Display Summary of Analog Input (AI) Object Using 'DAQ-2010'.

```
Acquisition Parameters:
     100000 samples per second on each channel.
     100000 samples per trigger on each channel.
     1 sec. of data to be logged upon START.
     Log data to 'Memory' on trigger.

Trigger Parameters:
     1 'Immediate' trigger(s) on START.

Engine status:
     Waiting for START.
     0 samples acquired since starting.
     0 samples available for GETDATA.

AI object contains no channels.

>> ai0  =  addchannel(ai_device, 0) %Add channel
     #0 to ai_device

     Index:  ChannelName:  HwChannel:
     InputRange:  SensorRange:  UnitsRange:
     Units:
     1       ''           0          [-10 10]
     [-10 10]       [-10 10]      'Volts'

>> set(ai_device, 'SampleRate', 1000)%Set
     SampleRate to 1000
>> set(ai_device, 'SamplePerTriger', 1000)%Set
     SamplePerTriger to 1000
>> start(ai_device)  %Start data acquisition
```

```
>> wait(ai_device, 10)%Wait for data acquisition
     to complete (wait timeout is 10 seconds)
>> ai_data = getdata(ai_device);%Get the waveform
     captured by ai_device object
>> plot (ai_data)%Plot the captured waveform
```

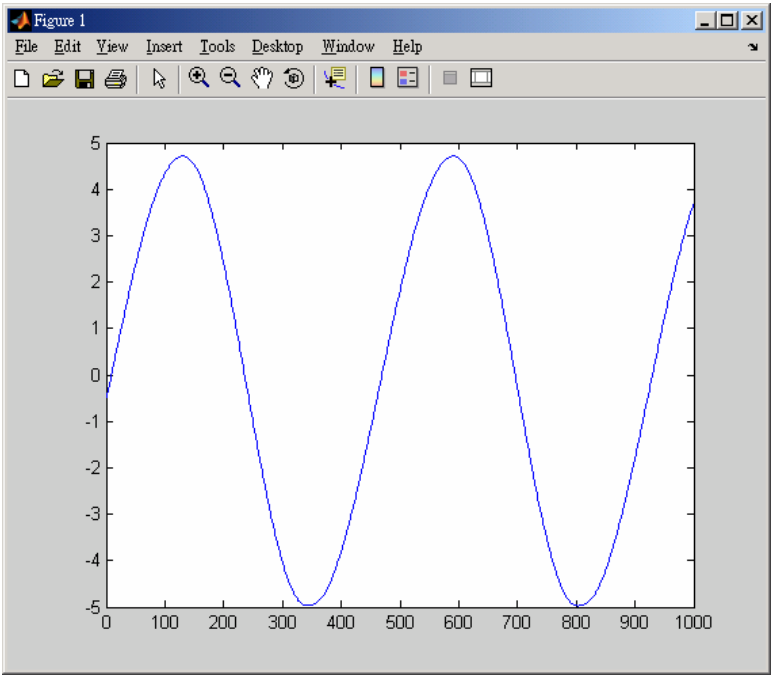


Figure 3-1: Continuous Analog Waveform

3.2.3 Capturing a Continuous Analog Waveform and Using Harding Triggering

This example shows how to use analog input channel #0 of the DAQ-2010 combined with hardware triggering to capture an analog waveform.

```
>> ai_device = analoginput('mwadlink', 0) %Opens
     the analog input functionality of device #0
     (DAQ-2010)
```

Display Summary of Analog Input (AI) Object Using 'DAQ_2010'.

```
Acquisition Parameters:
     100000 samples per second on each channel.
     100000 samples per trigger on each channel.
     1 sec. of data to be logged upon START.
     Log data to 'Memory' on trigger.

Trigger Parameters:
     1 'Immediate' trigger(s) on START.

Engine status:
     Waiting for START.
     0 samples acquired since starting.
     0 samples available for GETDATA.

AI object contains no channels.

>> ai0  =  addchannel(ai_device, 0) %Add channel
     #0 to ai_device

     Index:  ChannelName:  HwChannel:
     InputRange:  SensorRange:  UnitsRange:
     Units:
     1       ''           0          [-10 10]
     [-10 10]       [-10 10]      'Volts'

>> set(ai_device, 'SampleRate', 1000)%Set
     SampleRate to 1000
>> set(ai_device, 'SamplePerTriger', 1000)%Set
     SamplePerTriger to 1000
```

```
>> set(ai_device, 'TriggerType',
     'HWAnalogChannel') %Set TriggerType as
     HWAnalogChannel
>> set(ai_device, 'TriggerCondition', 'BelowLow')
     %Set Triggercondition as BelowLow
>> set(ai_device, 'TriggerConditionValue', [-1
     1]) %Set TriggerconditionValue as [-1 1]
>> start(ai_device)%Start acquiring data
>> wait(ai_device, 30)%Wait for data acquisition
     to complete (wait timeout is 30 seconds)
>> ai_data = getdata(ai_device);%Get the waveform
     captured by ai_device object
>> plot (ai_data)%Plot the captured waveform
```

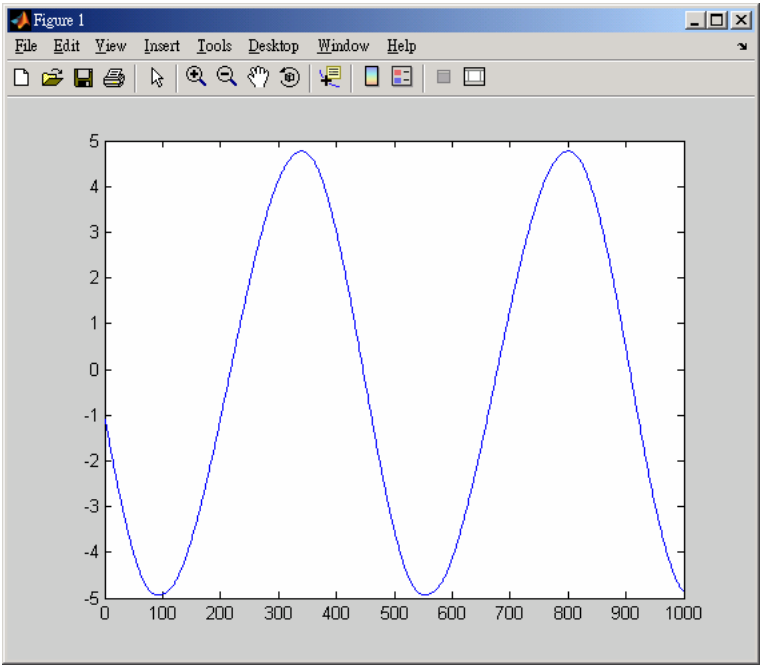


Figure 3-2: Continuous Analog Waveform and Using Harding Triggering

3.3 Analog Output

3.3.1 Output a Single Analog Output Signal

This example shows how to use DAQ-2010 analog output channel #0 to output a single analog signal

```
>> ao_device = AnalogOutput( 'mwadlink', 0)
     %Opens the analog output functionality of
     device #0 (DAQ-2010)
```

Display Summary of Analog Output (AO) Object Using 'DAQ-2010'.

```
Output Parameters:
     1000 samples per second on each channel.

Trigger Parameters:
     1 'Immediate' trigger on START.

Engine status:
     Waiting for START.
     0 total sec. of data currently queued for
     START.
     0 samples currently queued by PUTDATA.
     0 samples sent to output device since START.

AO object contains no channels.

>> ao0 = addchannel(ao_device, 0) %Add channel #0
     to ai_device

     Index:  ChannelName:  HwChannel:
     OutputRange:  UnitsRange:  Units:
     1       ''           0          [-10 10]
     [-10 10]     'Volts'

>> putsample(ao_device, 3) %Output 3V on channel
     #0
```

3.3.2 Output a Continuous Analog Output Signal

This example shows how to use DAQ-2010 analog output channel #0 to output a continuous analog signal

```
>> ao_device = AnalogOutput( 'mwadlink', 0)
     %Opens the analog output functionality of
     device #0 (DAQ-2010)
```

Display Summary of Analog Output (AO) Object Using 'DAQ-2010'.

```
Output Parameters:
     1000 samples per second on each channel.

Trigger Parameters:
     1 'Immediate' trigger on START.

Engine status:
     Waiting for START.
     0 total sec. of data currently queued for
     START.
     0 samples currently queued by PUTDATA.
     0 samples sent to output device since START.

AO object contains no channels.

>> ao0 = addchannel(ao_device, 0)%Add channel #0
     to ai_device

     Index:  ChannelName:  HwChannel:
     OutputRange:  UnitsRange:  Units:
     1       ''           0          [-10 10]
     [-10 10]     'Volts'

>> ao_data = 10 * sin(linspace(0, 2 * pi,
     1000)'); %Edit a sine wave signal and put in
     ao_data
>> putdata(ao_device, ao_data)%Download ao_data
     to analog output object ao_device
```

```
>> start(ao_device)  %Start continuous analog
      output. The sine wave will be viewable on
      channel #0
```

3.4 Digital Input/Output

3.4.1 Digital Input

This example shows how to acquire digital input signals using DIO Port A of the DAQ-2010

```
>> dio_device = digitalio( 'mwadlink', 0)
     %Opens DIO functionality of device #0 (DAQ-
     2010)
```

Display Summary of DigitalIO (DIO) Object Using 'DAQ-2010'.

```
Port Parameters:
     Port 0 is port configurable for reading and
     writing.
     Port 1 is port configurable for reading and
     writing.
     Port 2 is port configurable for reading and
     writing.
     Port 3 is port configurable for reading and
     writing.

Engine status:
     Engine not required.

DIO object contains no lines.

>> di_lines = addline(dio_device, 0:7, 'in')
     %Adds channels 0-7 to dio_device and sets
     them as input
   Index:  LineName:  HwLine:  Port:  Direction:
   1       ''         0        0      'In'
   2       ''         1        0      'In'
   3       ''         2        0      'In'
   4       ''         3        0      'In'
   5       ''         4        0      'In'
   6       ''         5        0      'In'
   7       ''         6        0      'In'
   8       ''         7        0      'In'
```

```
>> di_value = getvalue(di_lines) %Read the
     digital input values on channels 0-7 of Port
     A

di_value =

     1     1     1     1     1     1     1     1
```

3.4.2 Digital Output

This example shows how to output digital output signals using DIO Port A of the DAQ-2010

```
>> dio_device = digitalio( 'mwadlink', 0)
     %Opens DIO functionality of device #0 (DAQ-
     2010)
```

Display Summary of DigitalIO (DIO) Object Using 'DAQ-2010'.

```
Port Parameters:
     Port 0 is port configurable for reading and
     writing.
     Port 1 is port configurable for reading and
     writing.
     Port 2 is port configurable for reading and
     writing.
     Port 3 is port configurable for reading and
     writing.

Engine status:
     Engine not required.

DIO object contains no lines.

>> do_lines = addline(dio_device, 0:7, 'out')
     %Adds channels 0-7 to dio_device and sets
     them as output

   Index:  LineName:  HwLine:  Port:  Direction:
   1       ''         0        0      'Out'
   2       ''         1        0      'Out'
   3       ''         2        0      'Out'
   4       ''         3        0      'Out'
   5       ''         4        0      'Out'
   6       ''         5        0      'Out'
   7       ''         6        0      'Out'
   8       ''         7        0      'Out'

>> putvalue(do_lines, [1 0 1 0 1 0 1 0]) %Output
     [1 0 1 0 1 0 1 0] through channels 0-7
```

4 Explanation of Common DAQ-MTLB Properties

4.1 Common Analog Input Properties

4.1.1 AI object properties

4.1.1.1 InputType

Description
Type of specified analog input signals

Values
'Differential' Differential input
'SingleEnded' Single-ended input
'NonReferencedSingleEnded' Non-referenced single-ended input

Syntax
```
Set(obj, 'InputType', 'SingleEnded')
```

Notes
Refer to the hardware user's guide for connection methods of each analog input signal type.

4.1.1.2 SampleRate

Description
Sample rate of designated data

Values
10 ~ highest sample rate for the DAQ

Syntax
```
Set(obj, 'SampleRate', 1000)
```

Notes
Due to hardware restrictions, some data sample rates set by users may not be able to be used. If this occurs, DAQ-MTLB will automatically adjust DAQ sample rate to the nearest tolerable value.

4.1.1.3 SamplesPerTrigger

Description
Designates the number of samples captured with each trigger.

Values
1 ~ The largest amount of system memory permitted

Syntax
```
Set(obj, 'SamplesPerTrigger', 1000)
```

Notes
When setting the number of samples for each capture, ensure that the system's file or memory space is sufficient.

4.1.1.4 TriggerType

Description
Specify the desired trigger type.

Values

Values	Description	Supported Products
'Immediate'	Immediately start capturing and recording data when an input/output object issues a start() command	9111, 9112, 9113, 9114, 9116, 9118, 9810, 9812, 9820, 2005, 2006, 2010, 2204, 2205, 2206, 2208, 2213, 2204, 2501, 2502
'Manual'	Begin sampling data when a start() command is issued from the input/output object; wait until a trigger() command is issued before recording data.	9111, 9112, 9113, 9114, 9116, 9118, 9810, 9812, 9820, 2005, 2006, 2010, 2204, 2205, 2206, 2208, 2213, 2204, 2501, 2502
'Software'	Begin sampling data when a start() command is issued from the input/output object; wait until set trigger conditions are met before recording data.	9111, 9112, 9113, 9114, 9116, 9118, 9810, 9812, 9820, 2005, 2006, 2010, 2204, 2205, 2206, 2208, 2213, 2204, 2501, 2502
'HwDigital'	Immediately start capturing data when an input/output object issues a start() command, but wait until after an external digital trigger fires before recording. This selection requires hardware support.	9111, 9114, 9116, 9118, 9810, 9812, 9820, 2005, 2006, 2010, 2204, 2205, 2206, 2208, 2213, 2204, 2501, 2502
'HwAnalogChannel'	Start capturing data when an input/output object issues a start() command, but wait until analog input channel #0 signal trigger conditions are met before recording. This selection requires hardware support.	9810, 9812, 9820, 2005, 2006, 2010, 2204, 2205, 2206, 2208, 2213, 2204, 2501, 2502
'HwAnalogPin'	Start capturing data when an input/output object issues a start() command, but wait until external analog trigger channels signal conditions are met before recording. This selection requires hardware support.	2005, 2006, 2010, 2204, 2205, 2206, 2208, 2213, 2204, 2501, 2502

Syntax

```
Set(obj, 'TriggerType', 'Immediate')
```

Notes

Refer to the hardware user's guide for connection methods and signal definitions of each analog trigger.

4.1.1.5 TriggerCondition

Description

Specified trigger conditions (in conjunction with TriggerConditionValue and a specific TriggerType

Values

'None'	None
'Rising'	Rising trigger signal (for analog triggers)
'Falling'	Falling trigger signal (for analog triggers)
'TriggerPositive'	Positive edge trigger (for digital triggers)
'TriggerNegative'	Negative edge trigger (for digital triggers)
'AboveHigh'	Above the high trigger value (for analog triggers)
'BelowLow'	Below the low trigger value (for analog triggers)
'InsideRegion'	Inside the trigger region (for analog triggers)
'HighHysteresis'	High hysteresis trigger (for analog triggers)
'LowHysteresis'	Low hysteresis trigger (for analog triggers)

Reference the following table for each type of TriggerCondition and TriggerType combinations:

TriggerType	Permitted Values	Supported Products
'Immediate'	'None'	9111, 9112, 9113, 9114, 9116, 9118, 9810, 9812, 9820, 2005, 2006, 2010, 2204, 2205, 2206, 2208, 2213, 2204, 2501, 2502
'Manual'	'None'	9111, 9112, 9113, 9114, 9116, 9118, 9810, 9812, 9820, 2005, 2006, 2010, 2204, 2205, 2206, 2208, 2213, 2204, 2501, 2502
'Software'	'Rising' 'Falling'	9111, 9112, 9113, 9114, 9116, 9118, 9810, 9812, 9820, 2005, 2006, 2010, 2204, 2205, 2206, 2208, 2213, 2204, 2501, 2502
'HWDdigital'	'TriggerPositive' 'TriggerNegative'	9111, 9114, 9116, 9118, 9810, 9812, 9820, 2005, 2006, 2010, 2204, 2205, 2206, 2208, 2213, 2204, 2501, 2502

TriggerType	Permitted Values	Supported Products
'HWAnalogChannel'	'Rising' 'Falling'	9810, 9812, 9820
	'AboveHigh' 'BelowLow' 'InsideRegion' 'HighHysteresis' 'LowHysteresis',	2005, 2006, 2010, 2204, 2205, 2206, 2208, 2213, 2204, 2501, 2502
'HWAnalogPin'	'AboveHigh' 'BelowLow' 'InsideRegion' 'HighHysteresis' 'LowHysteresis'	2005, 2006, 2010, 2204, 2205, 2206, 2208, 2213, 2204, 2501, 2502

Syntax

```
Set(obj, 'TriggerType', 'Immediate')
```

Notes

Refer to the hardware's manual for more information of each type of triggering condition.

4.1.1.6 TriggerConditionValue

Description

Set one or two values used with TriggerCondition to determine triggering conditions.

Values

TriggerCondition	Permitted Values
'Rising' 'Falling'	A single number.
'AboveHigh' 'BelowLow' 'InsideRegion' 'HighHysteresis' 'LowHysteresis'	Two numbers (from small to large)

Syntax

```
Set(obj, 'TriggerConditionValue', 0)%A single
     number.
```

```
Set(obj, 'TriggerConditionValue', [-1 1]) %Two
     numbers.
```

4.1.1.7 TriggerDelay

Description

Sets a delay in number of samples or time before a trigger is generated and data is captured.

Values

TriggerType	Permitted Values
'Immediate'	0
'Manual'	0
'Software'	-2000000~2000000
'HWDdigital'	0~32767
'HWAnalogChannel'	0~32767
'HWAnalogPin'	0~32767

Syntax

```
Set(obj, 'TriggerDelay', 0)
```

Notes

TriggerDelay settings can be used achieve trigger modes such as Post-trigger, Delay-Trigger, Pre-Trigger, Middle-Trigger. Refer to the hardware's manual for more information on these types of triggering modes.

4.1.1.8 TriggerDelayUnits

Description

Specify TriggerDelay units.

Values

'Seconds'	In units of seconds
'Samples'	In units of number of samples

4.1.1.9 TriggerChannel

Description

Sets the trigger signal source channel.

Values
Channel object

Syntax
```
Set(obj, 'TriggerChannel', chan_obj)
```

Notes
TriggerChannel can only be used when TriggerType is set as Software.
When TriggerType is set as 'HWAnalogChannel', only input channel #0 can be the trigger signal source.

4.1.1.10 SubType

Description
Sets the DAQ SubType. ADLINK's 9111, 9114, 9118, 9812 10) require SubType to be set.

Values

Values	Products
'DG'	9111DG, 9114DG, 9118DG
'HG'	9114HG, 9118HG
'HR'	9111HR, 9118HR
9810	9810
9812	9812

Syntax
```
Set(obj, 'SubType', 'DG')
```

4.1.2 AI channel object properties

4.1.2.1 InputRange

Description
Sets an input range

Values
[min max] min is the smallest input range value, max is the largest input range value.

Syntax
```
Set(obj, 'InputRange', [-5 5])
```

Notes
Different DAQs have different input range settings. Refer to the hardware's manual for more information on input ranges.

4.2 Common Analog Output Properties

4.2.1 AO object properties

4.2.1.1 SampleRate

Description
Output rate of designated data

Values
10 ~ highest output rate for the DAQ

Syntax
```
Set(obj, 'SampleRate', 1000)
```

4.2.1.2 RepeatOutput

Description
Sets the number of repeated outputs.

Values
-1 Continuously output a waveform until the user sends a stop() command.
0~65535 Number times of repeated outpus.

Syntax
```
Set(obj, 'RepeatOutput', 0)
```

Notes
If RepeatOutput is set to 0, repeated output will not occur and the waveform will output one time.
If RepeatOutput is set to 1, repeated output will occur once, and the waveform will output two times.
And so on...

4.2.2 AO channel object properties

4.2.2.1 OutputRange

Description
Sets the analog output range (minimum and maximum values)

Values
[min max] min is the smallest output range value, max is the largest output range value.

Syntax

```
Set(obj, 'OutputRange', [-5 5])
```

Notes
Different DAQs have different output range settings. Refer to the hardware's manual for more information on output ranges.

4.3 Common Digital Input/Output Properties

4.3.1 DIO Line Object

4.3.1.1 Direction

Description
Sets a DIO Line object as digital input or output.

Values
- ’in’ Sets a DIO Line object as digital input.
- ’out’ Sets a DIO Line object as digital output.

Syntax
```
Set(obj, 'Direction', 'In')
```

Notes
Due to hardware restrictions, digital input or output settings of the DIO object must be in units of ports. Modifying a single DIO channel will affect all the DIO channels of that port.

Warranty Policy

Thank you for choosing ADLINK. To understand your rights and enjoy all the after-sales services we offer, please read the following carefully.

1. Before using ADLINK's products please read the user manual and follow the instructions exactly. When sending in damaged products for repair, please attach an RMA application form which can be downloaded from: http://rma.adlinktech.com/policy/.
2. All ADLINK products come with a two-year guarantee:
   - The warranty period starts from the product's shipment date from ADLINK's factory.
   - Peripherals and third-party products not manufactured by ADLINK will be covered by the original manufacturers' warranty.
   - For products containing storage devices (hard drives, flash cards, etc.), please back up your data before sending them for repair. ADLINK is not responsible for loss of data.
   - Please ensure the use of properly licensed software with our systems. ADLINK does not condone the use of pirated software and will not service systems using such software. ADLINK will not be held legally responsible for products shipped with unlicensed software installed by the user.
   - For general repairs, please do not include peripheral accessories. If peripherals need to be included, be certain to specify which items you sent on the RMA Request & Confirmation Form. ADLINK is not responsible for items not listed on the RMA Request & Confirmation Form.

3. Our repair service is not covered by ADLINK's two-year guarantee in the following situations:
   - Damage caused by not following instructions in the user's manual.
   - Damage caused by carelessness on the user's part during product transportation.
   - Damage caused by fire, earthquakes, floods, lightening, pollution, other acts of God, and/or incorrect usage of voltage transformers.
   - Damage caused by unsuitable storage environments (i.e. high temperatures, high humidity, or volatile chemicals).
   - Damage caused by leakage of battery fluid during or after change of batteries by customer/user.
   - Damage from improper repair by unauthorized technicians.
   - Products with altered and/or damaged serial numbers are not entitled to our service.
   - Other categories not protected under our warranty.
4. Customers are responsible for shipping costs to transport damaged products to our company or sales office.
5. To ensure the speed and quality of product repair, please download an RMA application form from our company website: http://rma.adlinktech.com/policy. Damaged products with attached RMA forms receive priority.

If you have any further questions, please email our FAE staff: service@adlinktech.com.