ADC-10, ADC-12, ADC-40 and ADC-42

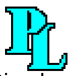
User's Guide

adc1042.en-2

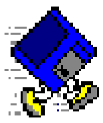
Contents

1 Contents

PicoScope Manual (Osciloscope/ Spectrum analyser software)



PicoLog Manual (Datalogging software)



Software Updates

Writing your own software (overview)

- C/C++
- Visual Basic
- Delphi
- Excel
- Agilent VEE
- LabVIEW
- Linux

Other information

- Contact Pico
- Technical Support
- Signal Conditioners
- Safety
- Legal
- Print this manual

2 Introduction

The Pico ADC-10, ADC-12, ADC-40 and ADC-42 are medium-speed, single-channel, analogue-to-digital converters.

All of these products can be used as virtual instruments (oscilloscope / spectrum analyser) with the PicoScope program, or as a data logger using PicoLog. There are also stand-alone programs to collect large blocks of data under Windows. Alternatively, you can use the driver software to develop your own programs to collect and analyse data from the unit (the same driver works for all units).

There are three differences between these products:

- the ADC-10 and ADC-40 are 8-bit converters, and the ADC-12 and ADC-42 are 12-bit converters
- the ADC-10 and ADC-12 accept 0 to +5 volts, whereas the ADC-40 and ADC-42 accept -5 to +5 volts
- the ADC-40 and ADC-42 have an input impedance of 1 megohm, and so are suitable for use with x10 scope probes. The ADC-10 and ADC-12 have a lower input impedance, and will give misleading results if used with a x10 probe.

This manual describes the physical and electrical properties of the ADC-10/12/40/42, and explains how to use the software drivers. Please refer to the documents listed below for further information about other items.

PicoLog for Windows	Online help file
PicoScope for Windows	Online help file

2.1 Connecting to the PC

The ADC-10, ADC-12, ADC-40 and ADC-42 can be connected to the PC in two ways:

- directly to a printer port on the computer
- to a USB port on the computer, via a Pico USB parallel port adapter.

Printer port operation

When you install the application software from the Pico CD, the computer will ask you which port to use. You should select LPT1, LPT2 or LPT3.

To use the ADC, you should connect it to the printer port on your computer, either directly or using a good quality extension cable.

USB parallel port operation

Please note that USB printer port interfaces are not suitable for use with Pico products. If you wish to connect a Pico product to a USB port, you will need a Pico USB Parallel Port adapter. You will also need Windows XP SP2 or Vista.

When you install the application software from the Pico CD:

- When the computer asks you which port to use, select USB-PP1.
- Once the USB driver software is installed, connect the Pico USB parallel port adapter to your PC: the computer will automatically configure the drivers.

See streaming for more information about the advantages of operating via a Pico USB parallel port.

Checking the installation

To check that the unit is working, start up the PicoScope program and then connect a voltage source to the BNC connector. The ADC has the same connectors as an oscilloscope, so you can use standard oscilloscope probes. Oscilloscope probes with x10 facility require 1 megaohm input impedance and should therefore only be used with the ADC-40 and ADC-42.

PicoScope should now display the voltage that you have connected. If you are using scope probes, when you touch the scope probe tip with your finger, you should see a small 50 Hz mains signal on the screen.

If you have connected the ADC to a parallel port other than the port specified when you installed the software, you will need to go to the Setup panel and then change the port number to the appropriate value (USB port numbers begin with USB-PPx. If you have more than one USB parallel port, they will be numbered according to the order they are plugged into the PC). You will need to exit and re-enter the software to activate the change.

2.2 Legal information

Grant of licence

The material contained in this release is licensed, not sold. Pico Technology Limited grants a license to the person who installs this software, subject to the conditions listed below.

Access

The licensee agrees to allow access to this software only to persons who have been informed of and agree to abide by these conditions.

Usage

The software in this release is for use only with Pico products or with data collected using Pico products.

Copyright

Pico Technology Limited claims the copyright of and retains the rights to all material (software, documents etc) contained in this release.

The user may copy and distribute the entire release in installable form, but you may not copy individual items within the release other than for backup purposes.

Liability

Pico Technology and its agents shall not be liable for any loss, damage or injury, howsoever caused, related to the use of Pico Technology equipment or software, unless excluded by statute.

Fitness for purpose

No two applications are the same, so Pico Technology cannot guarantee that its equipment or software is suitable for a given application. It is therefore the user's responsibility to ensure that the product is suitable for the user's application.

Mission-critical applications

Because the software runs on a computer that may be running other software products, and may be subject to interference from these other products, this licence specifically excludes usage in mission-critical applications such as life-support systems.

Viruses

This software was continuously monitored for viruses during production, but the user is responsible for virus checking the software once it is installed.

Support

If you are dissatisfied with the performance of this software, please contact our technical support staff. If we believe that there is a problem, we will try to fix the problem within a reasonable timescale. If you are still dissatisfied, please return the product and software to your supplier within 28 days of purchase for a full refund.

Upgrades

Pico Technology provides upgrades, free of charge, from our web site `www.picotech.com`. We reserve the right to charge for updates or replacements sent out on physical media.

Trade marks

Windows is a registered trademark of Microsoft Corporation.

Pico Technology Limited, DrDAQ and PicoScope are internationally registered trade marks.

2.3 Safety warning

We strongly recommend that you read the general safety information below before using your product for the first time. If the equipment is not used in the manner specified, then the protection provided may be impaired. This could result in damage to your computer and injury to yourself or others.

Maximum input range

The ADC-10 and ADC-12 are designed to measure voltages in the range 0 to +5V. Any voltages in excess of ±30V may cause permanent damage to the unit.

The ADC-40 and ADC-42 are designed to measure voltages in the range -5V to +5V. Any voltages in excess of ±30V may cause permanent damage to the unit.

Mains voltages

No Pico products are designed for use with mains voltages. To measure mains we recommend the use of a differential isolating probe specifically designed for such measurements.

Safety grounding

The ground of every product is connected directly to the ground of your computer via the provided interconnecting cable. This is done in order to minimise interference. Always use the provided cable to attach the product to your computer.

As with most oscilloscopes and data loggers, you should take care to avoid connecting the ground input of the product to anything which may be at some voltage other than ground. If in doubt, use a meter to check that there is no significant AC or DC voltage. Failure to check may cause damage to the product and computer and could cause injury to yourself or others.

You should assume that the product does not have a protective safety earth. Misconfiguration or use on voltages outside the maximum input range can be hazardous.

Repairs

The unit contains no user serviceable parts: repair or calibration of the unit requires specialised test equipment and must be performed by Pico Technology Limited or their authorised distributors.

3 Product Features

3.1 Specification

Resolution	8 bits
Number of input channels	1
Input range	0 to 5 V
Maximum sampling rate	20 kS/s
Repeatability	±4 LSB at 25°C
Absolute accuracy	±1% typical at 25°C
Overvoltage protection	±30 V
Input impedance	200 kiloohm
Output connector	D25 to PC parallel port
Input connectors	BNC
Power requirements	power supply required
Environmental conditions	0 to 70°C 0 to 95% humidity NOT water-resistant

3.2 Asynchronously

The Pico USB parallel port takes samples from the product, under its own control, and stores them until the computer has time to read them. The term **asynchronous** is used to indicate that the time the samples are collected is not the same as the time the PC requests them.

3.3 Streaming

If a device is connected to a Pico USB parallel port, data is collected asynchronously, without any intervention from the PC. This gives considerably more reliable data collection, and sampling does not interfere with the operation of the your computer.

When collecting data from a streaming device using the drivers, three modes are available:

- `BM_SINGLE` - collect a single block of data and exit
- `BM_WINDOW` - collect a series of overlapping blocks of data
- `BM_STREAM` - collect a continuous stream of data.

`BM_SINGLE` is useful when you wish to collect data at high speed for a relatively short period. For example, to collect 1000 readings in 50 ms.

`BM_WINDOW` is useful when collecting several blocks of data at relatively low speeds- for example when collecting 10000 samples over 10 seconds. Collecting a sequence of SINGLE blocks like this would take 10 seconds for each block, so displayed data would not be updated frequently. Using windowing, it is possible to ask for a new block more frequently, for example every second, and to receive a block containing nine seconds of data that have already been seen and one second of new data. The block is effectively a 10-second 'window' that advances one second each time.

`BM_STREAM` is useful when you need to collect data continuously for long periods. In principle, it would be possible to collect data indefinitely. Each time `adc10_get_values` is called, it returns the new readings since the last time it was called. `No_of_values` passed to `adc10_run` must be sufficient to ensure that the buffer does not overflow between successive calls to adc10_get_values. For example, if you call `adc10_get_values` every second, and you are collecting 500 samples per second, `no_of_values` must be at least 500, and preferably 1000 to give some allowance for delays in the operating system.

3.4 Technical information

Scaling	(Converting from ADC counts to voltage)

3.4.1 Scaling

The ADC-10 is an 8-bit unipolar analog to digital converter. This means that it produces values in the range 0 to 255 to represent voltages between 0 and 5 volts. To convert from ADC readings to volts, you should multiply by 5 and divide by 255. Thus, an ADC reading of 132 represents 132 x 5 / 255 = 2.588 volts.

The ADC-12 is a 12-bit unipolar analog to digital converter. This means that it produces values in the range 0 to 4095 to represent voltages between 0 and 5 volts. To convert from ADC readings to volts, you should multiply by 5 and divide by 4095. Thus, an ADC reading of 132 represents 132 x 5 / 4095 = 0.161 volts.

The ADC-40 is an 8-bit bipolar analog to digital converter. This means that it produces values in the range 0 to 255 to represent voltages between -5 and 5 volts. To convert from ADC readings to volts, you should subtract 128, multiply by 5 and divide by 128. Thus, an ADC reading of 132 represents (132-128) x 5 / 128 = 0.156 volts.

The ADC-42 is a 12-bit bipolar analog to digital converter. This means that it produces values in the range 0 to 4095 to represent voltages between -5 and 5 volts. To convert from ADC readings to volts, you should subtract 2048, multiply by 5 and divide by 2048. Thus, an ADC reading of 132 represents (132-2048) x 5 / 2048 = -4.677 volts.

4 Driver Formats & Routines

The ADC-10, ADC-12, ADC-40 and ADC-42 are supplied with driver routines that you can build into your own programs. Drivers are provided for the following operating systems:

- Windows XP
- Windows Vista
- Linux

Once you have installed the software, the `Examples` subdirectory contains the drivers and a selection of examples of how to use the drivers. It also contains a copy of this help file in text format.

The driver routine is supplied as Dynamic Link Libraries for Windows.

The Windows DLLs can be used with any programming language or application that can interface with DLLs- for example, C, Delphi, Visual Basic, Excel, Labview, etc. The `Examples` directory contains example programs for several popular programming languages or applications: some of these examples are fairly simple, but the C console mode example, `adc10con.c`, shows how to use all facilities in the driver.

The driver is capable of supporting up to three parallel port units (one each on LPT1, LPT2 and LPT3), and up to four Pico USB parallel port units. The units can be any mixture of ADC-10, ADC-12, ADC-40 and ADC-42.

The following table specifies the function of each of the routines in the Windows drivers:

Routine	**Function**
adc10_get_driver_version	Check that this is the correct driver
adc10_open_unit	Open the driver to use a specified parallel port
adc10_set_unit	Select which ADC-10 unit to use
adc10_close_unit	Close the specified printer port
adc10_get_value	a single reading
adc10_get_value_and_time	a single reading and the time
adc10_set_trigger	Set a trigger event
adc10_set_interval	the time interval for the next call to adc10_get_values, or adc10_get_times_and_values
adc10_is_streaming	Check whether the device supports streaming (USB only)
adc10_run	Start the unit recording
adc10_ready	Check whether the data recording is completed
adc10_stop	Abort data collection
adc10_get_values	a block of readings at fixed intervals
adc10_get_times_and_values	Get a block of readings and their times, at fixed intervals
adc10_get_unit_info	Get information about an ADC10 unit

The driver offers the following facilities:

- specify the printer port or USB parallel port that is connected to the ADC
- take a single reading
- specify a trigger event (only available in block mode)
- collect a block of samples at fixed time intervals

You can specify a sampling interval from 50 µs to a second. If you specify an interval that is shorter than your computer can manage, the driver will tell you how long it will actually take to collect the specified number of samples.

Under Windows, if you connect the product to the computer via a Pico USB parallel port, timing is completely reliable. However, if you connect the product to the computer via the printer port, the sampling may be affected by Windows activities. At the least, there will be gaps in the data every 55 milliseconds due to the Windows timer function. There will be additional gaps if you move the mouse, or have other programs running. We therefore recommend using the `adc10_get_times_and_values` routine, so that you can determine the exact time that each reading was taken.

The normal calling sequence to collect a block of data is as follows:

1. Check that the driver version is correct
2. Open the driver
3. Set trigger mode (if required)
4. Set sampling mode (channels and time per sample)
5. While you want to take measurements,
6. Run
7. While not ready
8. Wait
9. End while
10. Get a block of data
11. End while
12. Close the driver

4.1 Driver Formats

4.1.1 Windows XP/Vista

The Windows NT parallel port driver, `PICO.SYS`, is installed in Windows. The operating system must be told that the driver is available: this is normally done automatically by the setup program, but can also be done manually using the the regdrive.exe program which is copied into the `PICO` directory.

Type in:

```
regdrive pico
```

The Windows USB port driver, `PICOPP.SYS`, is installed in `Examples`. The file `picopp.inf`, must be placed in so that Windows knows which driver to load when the USB parallel port is plugged in.

The Windows 32-bit driver is accessed using the file `ADC1032.DLL`, and is installed in `Examples`. The DLL uses STDCALL linkage conventions, and undecorated names. The same `ADC1032.dll` file can be used in all 32-bit versions of Windows, for both parallel port and USB port connected products.

4.1.2 Linux

The ADC-10/12/40/42 are supported under Linux using the `picopar` parallel port driver kit. The tar file `picopar.tar`, available from the Pico web site, contains source code for the driver and example programs, together with full instructions to compile, install and run the software.

The Linux parallel port driver kit supports only units connected direct to the parallel port: it does not support USB-connected devices.

4.2 Procedures

4.2.1 adc10_get_driver_version

```
short adc10_get_driver_version (void)
```

This routine returns the version number of the ADC-10 driver. You can use it to check that your application is used only with the driver version that it was designed for use with.

Generally speaking, new driver versions will be fully backward compatible with earlier versions, though the converse is not always true, so it should be safe to check that the driver version is greater than or equal to the version that it was designed for use with.

The version is a two-byte value, of which the upper byte is the major version and the lower byte is the minor version.

4.2.2 adc10_open_unit

```
short adc10_open_unit
(
    short port,
    short product
)
```

This routine opens the ADC-10 driver.

If possible, it checks the BIOS printer address table and gets the address of the specified printer port. If not, it assumes that the printer ports 1..3 are at 0x378, 0x278 and 0x3BC.

It then calibrates the timing functions for the computer. It returns TRUE if successful. If it is not successful, you can call adc10_get_unit_info to find out why it failed.

Arguments

`port`	The number of the parallel port or USB port that the ADC-10 is connected to: 1 - LPT1 2 - LPT2 etc 101 - USB-PP1 102 - USB-PP2 etc
`product`	This identifies the type of product that you wish to use: 10 - ADC-10 12 - ADC-12 40 - ADC-40 42 - ADC-42

4.2.3 adc10_close_unit

```
short adc10_close_unit
(
   short port
)
```

This routine closes the ADC-10 driver.

Arguments

`port` The number of the parallel or USB port

4.2.4 adc10_set_unit

```
short adc10_set_unit
(
   short port
)
```

This routine is used to select the unit to use for subsequent operations. It is only necessary to use this function if you wish to have more than one unit open at the same time.

4.2.5 adc10_get_value

```
short adc10_get_value (void)
```

This routine reads the current value of one channel. Depending on your computer, it will take approx 100 µs to take one reading.

See also adc10_get_value_and_time, which reports the exact time at which the reading was taken.

4.2.6 adc10_get_value_and_time

```
void adc10_get_value_and_time
(
    unsigned long * sample_time,
    short         * value
)
```

This routine reads the current value from the currently selected ADC-10, and the time in microticks at which the reading was taken. Depending on your computer, it will take approx 100 µs to take one reading.

`sample_time` is the time in microticks for the reading. There are 2^{32} microticks per hour or 1,193,046 per second. The sample time therefore wraps around once an hour.

Value is scaled in ADC counts.

4.2.7 adc10_set_trigger

```
void adc10_set_trigger
(
    unsigned short enabled,
    unsigned short auto_trigger,
    unsigned short auto_ms,
    unsigned short dir,
    unsigned short threshold,
    short          delay
)
```

This routine defines a trigger event for the next block operation, and specifies the delay between the trigger event and the start of collecting the data block. Note that the delay can be negative for pre-trigger.

Arguments

`enabled immediately`	this is TRUE if the ADC-10 is to wait for a trigger event, and FALSE if the ADC-10 is to start collecting data
`auto_trigger`	this is TRUE if the ADC-10 is to trigger after a specified time (even if no trigger event occurs). This prevents the computer from locking up, if no trigger event occurs.
`auto_ms`	specifies the time in ms after which auto_trigger will occur.
`dir`	the direction can be rising or falling: rising = 0, falling = 1
`threshold`	this is the threshold at which a trigger event on channel A or B takes place. It is scaled in ADC counts.
`delay`	This specifies the delay, as a percentage of the block size, between the trigger event and the start of the block. Thus, 0% means the first data value in the block, and -50% means that the trigger event is in the middle of the block.

4.2.8 adc10_set_interval

```
unsigned long adc10_set_interval
(
    unsigned long us_for_block,
    unsigned long ideal_no_of_samples
)
```

This routine specifies the time interval per sample and the channels to be used for calls to adc10_get_values or adc10_get_times_and_values.

Arguments

`us_for_block`	target total time in which to collect `ideal_no_of samples`, in micro seconds.
`ideal_no_of_samples`	specifies the number of samples that you intend to collect. This number is only used for timing calculations: you can actually collect a different number of samples when you call adc10_get_values.

An example of a call to this routine to set for 100 readings in 10000 μs is:

```
actual = adc10_set_interval (10000, 100);
```

The routine returns the actual time to collect this number of samples. This actual time may be greater than the target time if you specified a sampling interval that is faster than your computer can manage. If the specified sampling rate was too fast, you have the following choices:

- If the total time is important, collect fewer than the ideal number of samples so that the total block time is correct.
- If the number of samples is important, collect the same number of samples then allow for the fact that they took longer to collect.

4.2.9 adc10_is_streaming

```
short adc10_is_streaming (void)
```

This routine can be used to determine whether the device is capable of supporting streaming. If so, it will return TRUE (1). A streaming device collects data asynchronously. USB devices generally support streaming, whereas parallel port devices do not.

4.2.10 adc10_run

```
void adc10_run
(
    unsigned long  no_of_values,
    unsigned short method
)
```

This routine starts a streaming unit collecting data It collects readings at intervals and from channels specified in the most recent `adc10_set_interval` call. For non-streaming devices, this function has no effect.

Arguments

`no_of_values`	the number of samples to collect
`method`	the data collection method: `BM_SINGLE` (0) - collect a single block and stop `BM_WINDOW` (1) - collect a sequence of overlapping blocks `BM_STREAM` (2) - collect a continuous stream of data

4.2.11 adc10_ready

```
short adc10_ready (void)
```

This routine indicates whether a streaming device has completed its data collection. It returns TRUE if the device is ready to transfer data. For non-streaming devices, it always return TRUE.

4.2.12 adc10_stop

```
void adc10_stop (void)
```

This function cancels any pending request for data from a streaming device. It has no effect for non-streaming devices.

4.2.13 adc10_get_values

```
unsigned long adc10_get_values
(
    unsigned short HUGE * values,
    unsigned long         no_of_values
)
```

This routine reads in a block of values. It collects readings at intervals and from channels specified in the most recent `adc10_set_interval` call.

For streaming devices, you should not call this routine until `adc10_ready` returns TRUE.

For non-streaming devices, if F9 or F10 is pressed while collecting, the routine will return immediately.

In BM_SINGLE and BM_WINDOW modes, the return value will be zero if a key was pressed, and the total time in microseconds if a block was successfully collected.

In BM_STREAM mode, the return value will be the number of new readings placed in the values buffer.

4.2.14 adc10_get_times_and_values

```
unsigned long adc10_get_times_and_values
(
    long HUGE            * times,
    unsigned short HUGE * values,
    unsigned long         no_of_values
)
```

This routine reads a block of values from the unit in the most recent adc10_open_unit or adc10_set_unit call. It takes readings at nominal intervals specified in the most recent adc10_set_interval call, and returns the actual times for each reading.

For streaming devices, you should not call this routine until adc10_ready returns TRUE.

For non-streaming devices, if the F9 or F10 key is pressed while collecting, the routine will return immediately.

In BM_SINGLE and BM_WINDOW modes, the return value will be zero if a key was pressed, and the total time in microseconds if a block was successfully collected.

In BM_STREAM mode, the return value will be the number of new readings placed in the values buffer.

4.2.15 adc10_get_unit_info

```
short adc10_get_unit_info
(
    char  * str,
    short   str_lth,
    short   line,
    short   port
)
```

If the specified unit failed to open, this routine returns a text string which explains why the unit was not opened.

If the specified unit is open, the routine returns version information about the ADC-10 DLL, the Windows driver and the sampling rate.

Arguments

str	character string buffer for result

`str_lth`	length of buffer
`line`	0 to 3: selects which line to return
`port`	the printer port number (1..3) to return information for

4.2.16 adc10_labview_single

```
unsigned short adc10_labview_single
(
  float          * result,
  unsigned short   port,
  unsigned short   type
)
```

This function is provided for LabVIEW programmers. It returns a single conversion result.

Arguments

`result`	returns the result
`port`	the port number that the ADC-10, -12, -40 or -42 is connected to
`type`	the variant of the converter: 10: ADC-10 12: ADC-12 40: ADC-40 42: ADC-42

5 Programming Support

5.1 C

Windows

There are two C example programs: one is a very simple GUI application, and the other is a more comprehensive console mode program that demonstrates all of the facilities of the driver.

The GUI example program is a generic windows application- ie it does not use Borland AppExpert or Microsoft AppWizard. To compile the program, create a new project for a Windows Application containing the following files:

	`adc10tes.c`
	`adc10tes.rc`
either	`adc1032.lib` (Borland 32-bit applications)
or	`adc10ms.lib` (Microsoft Visual C 32-bit applications)

The following files must be in the same directory:

`adc10tes.rch`
`adc10w.h`
`adc1032.dll` (All 32-bit applications)

The console example program is a generic windows application- ie it does not use Borland AppExpert or Microsoft AppWizard. To compile the program, create a new project for a Console Application containing the following files:

	`adc10con.c`
either	`adc1032.lib` (Borland 32-bit applications)
or	`adc10ms.lib` (Microsoft Visual C 32-bit applications)

The following files must be in the same directory:

`adc10w.h`
`adc1032.dll` (All 32-bit applications)

C++

C++ programs can access all versions of the driver. If adc10w.h is included in a C++ program, the PREF1 macro expands to extern "C": this disables name-mangling (or decoration, as Microsoft call it), and enables C++ routines to make calls to the driver routines using C headers.

5.2 Delphi

The Delphi programming example requires the following files:

```
adc10.dpr
adc10.inc
adc10fm.dfm
adc10fm.pas
```

5.3 Excel

The easiest way to get data into Excel is to use the PicoLog for Windows program.

However, you can also write an Excel macro which calls `adc10xx.dll` to read in a set of data values. The Excel Macro language is similar to Visual Basic.

The example `ADC10xx.XLS` reads in 20 values from channels 1 and 2, one per second, and assigns them to cells A1..B20.

Note: it is usually necessary to copy the .DLL file to your directory.

5.4 Visual Basic

Version 4 and 5 (32 bits)

The `Examples` subdirectory contains the following files:

```
ADC1032.VBP
ADC1032.BAS
ADC1032.FRM
```

5.5 LabVIEW

The routines described here were tested using LabVIEW for Windows 95 version 4.0.

While it is possible to access all of the driver routines described earlier, it is easier to use the special Labview access routines if only single readings are required. The `adc10.llb` library in the `Examples` sub-directory shows how to access these routines.

To use these routines, copy `adc10.llb` and `adc1032.dll` to your LabVIEW `user.lib` directory. You will then find four sub-vis to access the ADC-10, ADC-12 ADC-40 and ADC-42, and some example sub-vis which demonstrate how to use them.

You can use one of these sub-vis for converter that you wish to use. The sub-vi accepts the port (1 for LPT1) and returns a voltage.

5.6 Agilent-VEE

The example routine `adc10.vee` is in the `Examples` subdirectory. This example uses the driver definitions in `adc10.vh`. It was tested using Agilent-VEE version 5 under Windows 95.

The example shows how to collect a block of data from the ADC-10. It would be necessary to alter the scaling for use with the ADC-12, ADC-40 or ADC-42.

Index

Pico Technology Ltd

The Mill House
Cambridge Street
St Neots PE19 1QB
United Kingdom
Tel: +44 (0) 1480 396 395
Fax: +44 (0) 1480 396 296
Web: www.picotech.com

adc1042.en-2 18.6.07