ADC-11/ADC-22

User's Guide

adc11.en-2

Contents

1 Introduction

1.1 Overview

The ADC-11 and ADC-22 are medium speed ADCs that come in a number of versions. The table below compares the differences between each of the version.

Product	Resolution	Channels	In connection	Out connection
ADC-11/10	10 bits	11	D25	D25
ADC-11/12	12 bits	11	D25	D25
ADC-11/10 USB	10 bits	11	D25	USB
ADC-11/12 USB	12 bits	11	D25	USB
ADC-22	10 bits	22	D25	D25

These ADCs can be used as oscilloscopes with the PicoScope program, or as data loggers using PicoLog. For information on PicoScope and PicoLog, please consult the associated help files. Alternatively, you can use the driver for the ADC-11 or ADC-22 to develop your own programs to collect and analyse ADC data.

This manual describes the physical and electrical properties of the parallel port and USB versions of the ADC-11/ADC-22, and explains how to use the Windows drivers. The following items are supplied in the package:

- ADC-11 or ADC-22
- CD containing the software
- Installation manual

1.2 Installing the driver

You may choose to install the driver when you install the PicoScope or PicoLog software. Alternately, you can download the driver from our website at www.picotech.com.

1.3 Connecting the ADC

Before you connect your ADC you should install the software supplied on the CD. How you connect your ADC depends on the ADC type and computer specification. The following are the three methods used:

- **ADC-11, USB version**

  To connect the USB ADC-11, plug the cable provided into a USB port on your PC. During installation you should select USB-PP1 when asked for the port.

- **ADC-11 or ADC-22, parallel version, using printer port**

  Connect the unit directly or via a good quality extension cable. During installation of the application software from the Pico CD, you will be asked which port you are using. You should select LPT1, LPT2 or LPT3.

- **ADC-11 or ADC-22, parallel version, using USB-to-parallel-port-converter**

  Connect the unit via a Pico USB parallel port adapter. Please note that USB printer port interfaces are not suitable for use with the ADC-11/22. If you want to connect an ADC-11/22 to a USB port, you will need a Pico USB parallel port adapter. You will also need Windows 98 SE, ME, 2000 or XP. During installation of the application software from the Pico CD you will be asked which port you want to use. You should select USB-PP1. Once the USB driver software is installed, connect the Pico USB parallel port adapter to your PC and the computer will automatically configure the drivers.

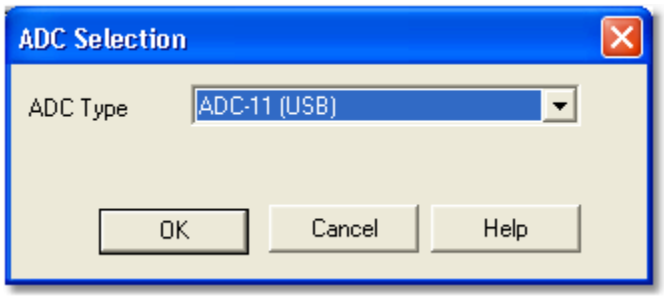
1.4 Software configuration

To check that the unit is working, start up the PicoScope program. You should immediately see a signal trace on the screen, As the inputs are high impedance, until a voltage is applied to the input, the signal will contain spurious noise. If you apply a DC source, such as an AA battery, you should see the signal jump to the corresponding voltage of the DC source (provided it is between 0 and 2.5 V).

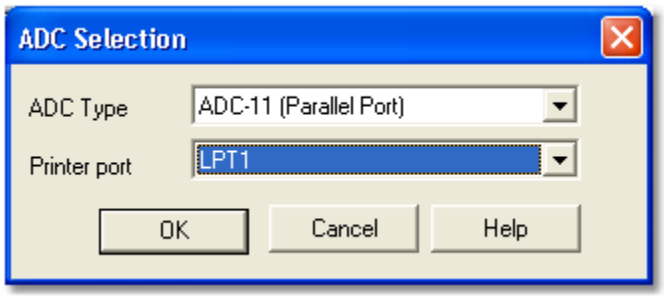
If you have connected the ADC to a port other than the one specified when you installed the software, you will need to do the following:

1. If you are using PicoScope, select Setup from the File menu, select Converter, then go to step 2. If you are using PicoLog, go straight to step 3.

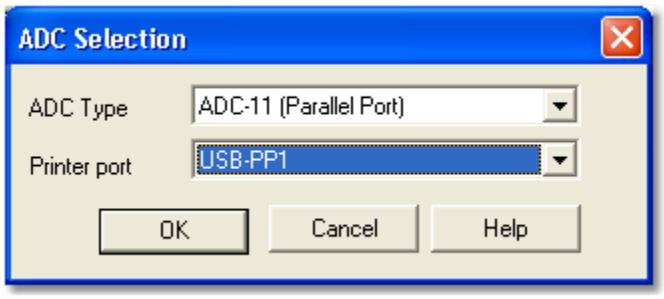
   The **ADC selection** dialog box appears. This one has the parallel ADC-11 on USB-PP1 selected



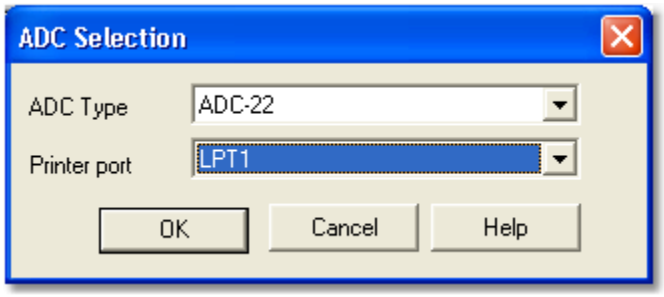
   Here is the same dialog box with the parallel ADC-11 on LPT1 selected



   This is the dialog box showing the USB ADC-11 selected on USB-PP1



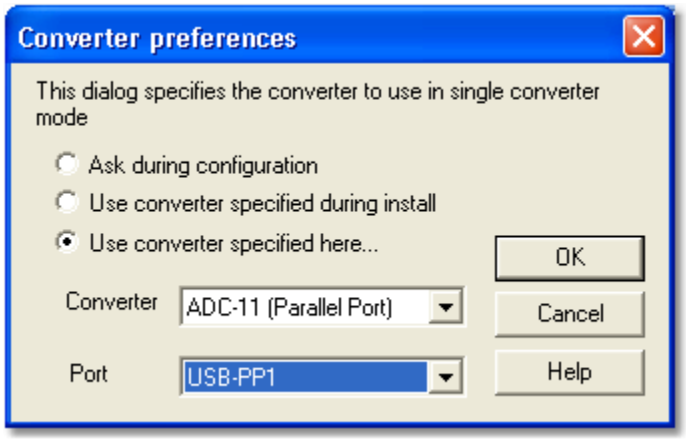
This is what the dialog box will look like with the ADC-22 on LPT1 selected



2. Click the Printer port drop-down arrow and select the appropriate port number. USB port numbers begin with USB-PPx. If you are using more than one Pico USB device, they will be numbered according to the order they are plugged into the PC. Note: You will need to exit PicoScope and restart the software before any changes are activated

3. If you are using PicoLog, select **Preferences** from the **File** menu, then select **Converter.**

   The **Converter preferences** dialog box appears



4. Select **Use converter specified here...**

   The **Converter** drop-down list changes from greyed out to active.

5. From the **Converter** drop-down list, select the ADC you are using.

6. From the **Port** drop-down list, select the appropriate port number. USB port numbers begin with USB-PPx. If you are using more than one Pico USB device, they will be numbered according to the order they are plugged into the PC. Note: You will need to exit PicoLog and restart the software before any changes are activated

1.5 Accessories

A terminal board is available separately for the ADC-11 and USB ADC-11. The terminal board plugs into the 25 way D-type input connector of the unit to provide screw terminal input connections for each of the 11 input channels and ground.

1.6 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/or injury to yourself or others.

Maximum input range

The ADC-11, USB ADC11 and ADC-22 are designed to measure voltages in the range of 0 - 2.5 V. If a voltage outside this range is connected to any of the input channels of the USB ADC11 the software will display an overvoltage indication (a warning triangle in PicoScope, and readings replaced by asterisks in PicoLog). For applications directly using the driver, overvoltage is reported to the user through the API. Any voltages in excess of ±30V may cause permanent damage to a unit.

Mains Voltages

Pico products are not 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 interconnecting cable provided. This is done to minimise interference. If the PC (especially laptops) is not grounded, reading stability cannot be guaranteed and it may be necessary to manually ground the equipment.

As with most oscilloscopes and data loggers, you should take care to avoid connecting the inputs of the product to any equipment which may be at an unsuitable voltage. If in doubt, use a meter to check that there is no hazardous AC or DC voltage. Failure to check may cause damage to the product and/or computer and could cause injury to yourself or others.

Take great care when measuring temperatures near mains equipment. If a sensor is accidentally connected to mains voltages, you risk damage to the converter or your computer and your computer chassis may become live.

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

Cleaning

The product may be cleaned externally using a damp cloth with water.

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.

1.7 Legal information

The material contained in this release is licensed, not sold. Pico Technology Limited grants a licence 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 these conditions and agree to abide by them.

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. You may copy and distribute the entire release in its original state, but must 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: Pico Technology cannot guarantee that its equipment or software is suitable for a given application. It is your responsibility, therefore, to ensure that the product is suitable for your application.

Mission-critical applications

This software is intended for use on a computer that may be running other software products. For this reason, one of the conditions of the licence is that it excludes usage in mission-critical applications, for example life support systems.

Viruses

This software was continuously monitored for viruses during production, but you are 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, who will try to fix the problem within a reasonable time scale. If you are still dissatisfied, please return the product and software to your supplier within 28 days of purchase for a full refund.

Upgrades

We provide upgrades, free of charge, from our web site. We reserve the right to charge for updates or replacements sent out on physical media.

Trademarks

Pico Technology Limited, PicoScope, PicoLog, DrDAQ and EnviroMon are trademarks of Pico Technology Limited, registered in the United Kingdom and other countries. Pico Technology acknowledges the following product names as trademarks of their respective owners: Windows, Excel, Visual Basic, LabVIEW, Agilent VEE, HP VEE, Delphi.

1.8 Company details

Address: Pico Technology Limited
The Mill House
Cambridge Street
St Neots
Cambridgeshire
PE19 1QB
United Kingdom

Phone: +44 (0) 1480 396 395
Fax: +44 (0) 1480 396 296

Email:
Technical Support support@picotech.com
Sales sales@picotech.com
Web site:
www.picotech.com

2 Product information

2.1 Specifications

	ADC-11/10	ADC-11/12	ADC-22	USB ADC-11/10	USB ADC-11/12
Analog bandwidth	DC to 5 kHz				
Maximum sampling rate	20 kS/s				
Analog Inputs	11	11	22	11	11
Digital outputs	1 or 2	1	1	2	2
Voltage range	0 to 2.5 V				
Linearity (at 25 °C)	1 LSB				
Resolution (bits)	10	12	10	10	12
Accuracy	1%	0.5%	1%	1%	0.5%
Overload protection	±30V (inputs and to ground)				
Input impedance	>= 1 MΩ				
Environmental operating conditions	20-30°C for quoted accuracy, 0 to 70°C overall. 20-90% RH				
PC connection	Parallel / USB 1.1 using Pico USB adapter (USB 2.0 compatible)			USB 1.1 (USB 2.0 compatible)	

2.2 Scaling

The ADC-22 and Version 1 ADC-11/10 are 10-bit ADCs. This means they produce values in the range 0 to 1023 to represent voltages between 0 and 2.5 volts. To convert from ADC readings to volts, you should multiply by 2.5 and divide by 1023. Thus, an ADC reading of 132 represents 132 x 2.5 / 1023 = 0.323 volts.

Note: Although the Version 2 ADC-11/10 is a 10-bit device, it returns 12-bit values for compatibility. Use `adc11_max_value` to distinguish one device from another.

The ADC-11 produces values in the range 0 to 4095. To convert ADC readings into volts, you should multiply by 2.5 and divide by 4095. Thus, an ADC reading of 132 represents 132 x 2.5 / 4095 = 0.08 volts.

2.3 Streaming

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

When you are collecting data from a streaming device using the drivers, three operational 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 10,000 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 9 seconds of repeat data and 1 second of new data. The block is effectively a 10-second window that advances one second per cycle.

BM_STREAM is useful when you need to collect data continuously for long periods. In principle, it would be possible to collect data indefinitely. Every time adc11_get_values is called, it returns the new readings since the last time it was called. No_of_values passed to adc11_run must be sufficient to ensure that the buffer does not overflow between successive calls to adc11_get_values. For example, if you call adc11_get_values every second, and you are collecting 500 samples per second, no_of_values must be at least 500, or preferably 1000, to allow for delays in the operating system.

3 Technical reference

3.1 Introduction

The ADC-11 and ADC-22 units are supplied with driver routines that you can build into your own programs. Drivers for the parallel and USB versions of the ADC-11 and ADC-22, and the ADC-11/ADC-22 attached to a USB port via a Pico parallel port converter, are supported by the following operating systems:

- Windows XP SP2
- Windows Vista

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

The driver routines are supplied as Windows DLLs.

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 and applications. Some of these examples are fairly simple, but the C console mode example, `adc11con.c`, illustrates how to use all the facilities available 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 USB units.

3.2 Windows XP SP2/Vista

The Windows parallel port driver, `PICO.SYS`, is installed in `windows\system32\drivers`. The operating system must be notified that the driver is available. This is normally done automatically by the setup program, but can also be done manually using the `regdrive.exe` program, copied into the Pico directory. Type in `regdrive pico`

The Windows USB port driver, `PICOPP.SYS`, is installed in `\windows\system32\drivers`. The file `picopp.inf`, must be placed in `\windows\inf` so that Windows knows which driver to load when the USB device is plugged in.

The 32-bit Windows drivers are accessed using the file `ADC1132.DLL` and is installed in `drivers\win32`. The DLL uses `STDCALL` linkage conventions, and undecorated names. The same `ADC1132.dll` file can be used in all 32-bit versions of Windows, with both parallel port and USB port products.

3.3 Driver routines

3.3.1 Introduction

The following table explains each of the driver routines supplied with the ADC-11/ADC-22:

Routine	Description
adc11_get_driver_version	Returns the driver version.
adc11_open_unit	Opens the driver, instructing it to use a specific port.
adc11_set_unit	Specifies the type of ADC-11/ADC-22 unit to use.
adc11_close_unit	Closes the specified port.
adc11_set_do	Sets the digital output.
adc11_set_do2	Sets the second digital output.
adc11_is_streaming	Checks whether the device supports streaming (USB only.)
adc11_run	Starts the unit recording.
adc11_ready	Checks whether data collection is completed.
adc11_stop	Aborts data collection.
adc11_set_trigger	Sets a trigger event from a specified channel.
adc11_set_interval	Sets the channels and time interval for the next call to adc11_get_values, or adc11_get_times_and_values
adc11_get_value	Retrieves a single reading from a specified channel.
adc11_get_values	Retrieves a block of readings at fixed intervals (see scaling.)
adc11_get_value_and_time	Retrieves a single reading from a specified channel and shows the time at which the reading was taken.
adc11_get_times_and_values	Retrieves a block of readings and the times at which they were taken, at fixed intervals (see scaling.)
adc11_get_unit_info	Retrieves information about the ADC-11/ADC-22 unit.
adc11_max_value	Returns the maximum ADC value.
adc11_open_usb_unit_async	Opens a USB ADC-11 or an ADC-11/22 on a Pico USB parallel port, without the calling thread being blocked.
adc11_open_usb_unit_progress	Checks the progress of the most recent call to adc11_open_unit_async.

The driver allows you to perform each of the following actions:

- Assign the port to which you are connecting the ADC-11/ADC-22
- Take a single reading from a particular channel
- Collect a block of samples at fixed time intervals from one or more channels
- Set up a trigger event for a particular channel (only available in block mode)

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.

If you connect the product to the computer via a USB 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 very 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 recommend, therefore, using the adc11_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)

Here is some pseudocode describing the process involved in taking measurements:

```
Run
While not ready
  Wait
End while
  Retrieve a block of data
End While
Close the driver
```

3.3.2 USB and parallel mode

3.3.2.1 adc11_get_driver_version

```
PREF1 short PREF2 adc11_get_driver_version (void)
```

This routine returns the version number of the ADC-11/ADC-22 driver. This is useful if you want to check that the version that you are writing your application for matches the version installed on your computer. In general, new releases of drivers are backwards-compatible.

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

Arguments:	none	
Returns:		Driver version. Upper byte is the major version; lower byte is the minor version. For example, 0x0301

3.3.2.2 adc11_open_unit

```
PREF1 short PREF2 adc11_open_unit
(
  short port,
  short product
)
```

This routine opens the ADC-11/ADC-22 driver.

The routine then calibrates the timing functions of the computer, returning TRUE if successful. If it is not successful, you can call adc11_get_unit_info to find out why it failed.

Arguments:	port	Details of the port to which the ADC-11/ADC-22 is connected: 1 - LPT1 2 - LPT2 3 - LPT3 101 - USB-PP1 102 - USB-PP2 103 - USB-PP3 104 - USB-PP4
	product	Product details: 11 - ADC-11 22 - ADC-22 111 - USB ADC-11
Returns:		TRUE if successful

3.3.2.3 adc11_set_unit

```
PREF1 short PREF2 adc11_set_unit
(
  short port
)
```

This routine selects the unit to be used for subsequent operations. It is only necessary to use this function if you want to manage more than one unit at the same time.

Arguments:	port	The parallel port number, as outlined in adc11_open_unit
Returns:		TRUE if successful; FALSE if a unit is not open on that port

3.3.2.4 adc11_close_unit

```
void PREF2 adc11_close_unit
(
  short port
)
```

This routine closes the ADC-11/ADC-22 driver.

Arguments:	port	The parallel port number, as outlined in adc11_open_unit
Returns:		None

3.3.2.5 adc11_set_do

```
PREF 1 void PREF2 adc11_set_do
(
  short do_value
)
```

This routine sets the state of digital output number one on the currently selected ADC. Any non-zero value will turn the digital output on; zero will turn it off.

Arguments:	do_value	1 - turns the digital output on 0 - turns the digital output off
Returns:		None

3.3.2.6 adc11_set_do2

```
PREF 1 void PREF2 adc11_set_do2
(
  short do_value
)
```

If the ADC you are using has two digital outputs, this routine sets the state of the second of the digital outputs. Any non-zero value will turn the digital output on; zero will turn it off.

The second of the digital outputs on an ADC-11 is allocated to pin 14. This pin is fitted as standard on the USB ADC-11, but on the parallel port ADC-11, if you want to make use of it, you will need to specifically request connection at purchase.

Arguments:	do_value	1 - turns the digital output on 0 - turns the digital output off
Returns:		None

3.3.2.7 adc11_is_streaming

```
short adc10_is_streaming (void)
```

This routine is for finding out whether the device is capable of supporting streaming. Devices supporting streaming collect data in an asynchronous manner. In general, USB devices support streaming, and parallel port devices do not.

Arguments:	none	
Returns:		TRUE, if device is capable of streaming

3.3.2.8 adc11_run

```
void adc11_run
(
  unsigned long  no_of_values,
  unsigned short method
)
```

This routine instructs a unit supporting streaming to begin to collect data, then collect readings at intervals from channels specified in the most recent adc11_set_interval call. When it is used with devices that do not support streaming, the 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

Returns:		None

3.3.2.9 adc11_ready

```
short adc11_ready (void)
```

This routine indicates whether a device supporting streaming has completed its data collection. It returns TRUE if the device is ready to transfer data. If it is used with devices that do not support streaming, it always return TRUE.

Arguments:	none	
Returns:		TRUE, if ready

3.3.2.10 adc11_stop

```
void adc11_stop (void)
```

This function cancels any pending request for data from devices supporting streaming. If it is used with devices that do not support streaming, the function has no effect.

Arguments:	none	
Returns:		None

3.3.2.11 adc11_set_trigger

```
PREF1 void PREF2 adc11_set_trigger
(
  unsigned short enabled,
  unsigned short auto_trigger,
  unsigned short auto_ms,
  unsigned short channel,
  unsigned short dir,
  unsigned short threshold,
  unsigned 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 collection of the data block. Note that the delay can be a negative value if you want to pre-trigger.

Arguments:	`enabled`	TRUE if the ADC is to wait for a trigger event, and FALSE if the ADC is to start collecting data immediately.
	`auto_trigger`	TRUE if the ADC is to trigger after a specified time if no trigger event occurs. This prevents the computer from locking up if no trigger event occurs on a parallel port unit.
	`auto_ms`	Specifies the time in ms after which auto_trigger will occur.
	`channel`	Specifies which channel is to be used as the trigger input. The channel number is 1 for Channel 1, through to 11 for Channel 11 (ADC-11) or 22 for Channel 22 (ADC-22).
	`dir`	The direction can be rising (FALSE) or falling (TRUE).
	`threshold`	The threshold at which a trigger event on the trigger channel takes place. It is scaled in ADC counts.
	`delay`	The delay between the trigger event and the start of the block as a percentage of the block size. Thus, 0% means that the trigger event is the first data value in the block, and -50% means that the trigger event is in the middle of the block.

Returns:		None

3.3.2.12 adc11_set_interval

```
PREF1 unsigned long PREF2 adc11_set_interval
(
  unsigned long   us_for_block,
  unsigned long   ideal_no_of_samples,
  short         * channels,
  short           no_of_channels
)
```

This routine specifies the time interval per sample and the channels to be used for calls to adc11_get_values or adc11_get_times_and_values.

Arguments:	us_for_block	Target total time in which to collect ideal_no_of samples, in μs
	ideal_no_of_samples	Specifies the number of samples that you intend to collect. This number is used only for timing calculations: you can actually collect a different number of samples when you call adc11_get_values
	channels	The address of an array, listing the channels to be used.
	no_of_channels	Specifies the number of channels to be used.
Returns:		Sample interval in μs.

The following is an example of a call to this routine using Channels 2, 3 and 5:

```
short channels [3];
channels [0] = 2;
channels [1] = 3;
channels [2] = 5;

adc11_set_interval (10000, 100, channels, 3);
```

The routine returns the actual time taken to collect the samples. The actual time may be greater than the target time if you specified a sampling interval that is too fast for your computer. If it turns out that the sampling rate was too fast, you have the following choices:

- If the total time is important, to make sure the total block time is correct, collect fewer samples than is ideal
- If the number of samples is important, collect the same number of samples then allow for the delay imposed by your computer

3.3.2.13 adc11_get_value

```
short PREF2 adc11_get_value
(
  short channel
)
```

This routine reads the current value of a single channel from the currently selected ADC-11/ADC-22. Depending on your computer, it will take approx 200µs to take one reading. The channel number is 1 for Channel 1, through to 11 for Channel 11 (up to a maximum of 22 if you are operating the ADC-22.)

Arguments:	`channel`	1 - returns a reading from Channel 1 2 - returns a reading from Channel 2 ... 22- returns a reading from Channel 22
Returns:		- the channel reading in ADC counts. See scaling

3.3.2.14 adc11_get_values

```
unsigned long PREF2 adc11_get_values
(
  unsigned short * 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 `adc11_set_interval` call.

If a block was successfully collected, the return value reflects the total time in μs it took to collect a block of data.

Arguments:	`values`	A pointer to an array where the sample values will be stored. The sample values are in ADC counts - see Scaling.
	`no_of_values`	The number of values that will be written to the `values` array.

Returns:		Number of values returned, which may not be as much as no_of_values if streaming.

3.3.2.15 adc11_get_value_and_time

```
PREF1 void PREF2 adc11_get_value_and_time
(
  short channel,
  unsigned long  * sample_time,
  unsigned short * sample
)
```

This routine reads a single value from the unit in the adc11_open_unit call and returns the actual time for the reading.

If a reading was successfully collected, the return value reflects the number of values returned.

Arguments:		
	sample_time	A pointer to a variable that will store the sample time, in ms.
	sample	A pointer to a variable where the sample value will be stored. Sample in ADC counts.

Returns:		
		None

3.3.2.16 adc11_get_times_and_values

```
PREF1 unsigned long PREF2 adc11_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 adc11_open_unit call. It takes readings at nominal intervals specified in the most recent adc11_set_interval call, and returns the actual times for each reading.

If a block was successfully collected, the return value reflects the total time in microseconds it took to collect a block of data.

Arguments:	times	A pointer to an array that will store the sample times in µs.
	values	A pointer to an array that will store the sample values in ADC counts.
	no_of_values	The number of values that will be written to the times array and values array.

Returns:		The number of values returned, which may not be as much as no_of_values if streaming.

3.3.2.17 adc11_get_unit_info

```
PREF1 short PREF2 adc11_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 explaining why the unit was not opened. If the specified unit is open, the routine returns version information about the DLL and the Windows driver.

Arguments:	`line`	The line number to return: 0 - driver version 1 - ADC-11/ADC-22 status 2 - kernel driver version 3 - sample rate
	`str`	A pointer to a string that will contain the unit information.
	`str_lth`	The string length.
	`port`	The port to which the ADC-11/ADC-22 is connected.
Returns:		The number of bytes written to `str`.

3.3.2.18 adc11_max_value

```
PREF1 short PREF2 adc11_max_value (void)
```

This routine returns the maximum ADC value for the unit. This will be either 1023 or 4095.

Arguments:	none	
Returns:		Maximum ADC value

3.3.3 USB mode only

3.3.3.1 adc11_open_usb_unit_async

```
short adc11_open_usb_unit_async
(
  short port,
  short unitType
)
```

Opens a USB ADC-11 or an ADC-11/22 on a Pico USB parallel port, without the calling thread being blocked.

Arguments:	`port`	The port to which the device is connected, for example, 101. For a full list of the port numbers, see adc11_open_unit.
	`unitType`	If the device is an ADC-11 on a Pico parallel port, use 11. If the device is an ADC-22 on a Pico parallel port, use 22. If the device is a USB ADC-11, use 111.
Returns:		-1 if the unit fails to open, 0 if no USB unit is found or 1 if the device opens successfully

3.3.3.2 adc11_open_usb_unit_progress

```
short adc11_open_usb_unit_progress
(
  short * progressPercent
)
```

Checks the progress of the most recent call to adc11_open_usb_unit_async.

Arguments:		
	progressPercent	- a pointer to the short that is to receive the percentage progress

Returns:		
		-1 if no open unit operation is in progress, 0 if the unit is still opening 1 if the operation completed successfully

3.4 Programming

3.4.1 Introduction

We supply examples for the following programming languages:

- C and C++
- Delphi
- Excel
- LabVIEW
- Visual Basic
- Agilent VEE

3.4.2 C and C++

C

ADC-11

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:

```
      adc11tes.c
      adc11tes.rc
      adc1132.lib (Borland 32-bit applications)
or    adc11ms.lib (Microsoft Visual C 32-bit applications)
```

The following files must be in the same directory:

```
      adc11tes.rch
      adc11w.h
      adc1132.dll (All 32-bit applications)
```

The example console 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:

```
      adc11con.c
      adc1132.lib (Borland 32-bit applications)
or    adc11ms.lib (Microsoft Visual C 32-bit applications)
```

The following files must be in the same directory:

```
      adc11w.h
      adc1132.dll (All 32-bit applications)
```

USB ADC-11

Use the following files:

```
usbadc11.lib
usbadc11api.h
usbadc11bc.lib
usbadc11con.c
```


C++

C++ programs can access all versions of the driver. If `adc11w.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.

3.4.3 Delphi

`adc11.dpr` is a complete program, which opens the driver and reads values from Channel 1.

The file `adc11.inc` contains a set of procedure prototypes that you can include into your own programs.

You will also require the following files:

```
adc11.vh
adc11fm.dfm
adc11fm.pas
```

3.4.4 Excel

ADC-11

The easiest way to transfer data to Excel is to use PicoLog. However, you can also write an Excel macro which calls `adc1132.dll` to read in a set of data values. The Excel macro language is similar to Visual Basic.

The example adc`1132.xls` reads in 20 values from Channels 1 and 2, one per second, and assigns them to cells A1..B20.

Use Excel Version 7 and above.

Note that it is usually necessary to copy the DLL file to your `\windows\system` directory.

USB ADC-11

Use the following file:

```
usbadc11.xls
```

3.4.5 LabVIEW

The routines described here were tested using LabVIEW 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 `adc11.llb` library in the `DRIVERS\WIN32` sub-directory shows how to access these routines.

To use these routines, copy `adc11.llb` and `adc1132.dll` to your LabVIEW `user.lib` directory. You will then find two sub-vis to access the ADC-11 and ADC-22, and some example sub-vis which demonstrate how to use them.

You can use one of these sub-vis for each of the channels that you wish to measure. The sub-vi accepts the port (`1` for LPT1) and channel (`1` to `11` or `1` to `22`, depending on converter type) and returns a voltage.

3.4.6 Visual Basic

Version 4 and 5 (32-bit)

The DRIVERS\WIN32 sub-directory contains the following files:

ADC1132.VBP
ADC1132.BAS
ADC1132.FRM

3.4.7 Agilent-VEE

The example routine adc11.vee is in the Examples subdirectory. It was tested using Agilent VEE version 5. The example shows how to collect a block of data from the ADC.

Index

S

V

W

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

adc11.en-2 15.6.07