

- [Introduction](#)
- [Connecting to PC](#)
- [Specifications](#)
- [Technical Information](#)
- [PicoLog Manual \(Data Logging software\)](#)
- [Software updates](#)

## Writing your own software

- [Overview](#)
- [C](#)
- [Delphi](#)
- [Visual Basic](#)
- [Excel](#)
- [Agilent VEE](#)
- [LabView](#)
- [Java](#)
- [PalmPilot](#)
- [Linux](#)

## Other information

- [Technical Support](#)
- [Signal Conditioners](#)
- [Safety](#)

- [Licence conditions](#)
- [Warranty](#)
- [Contacting Pico](#)

---

adc16.en-2 1.10.07

## 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-16 is designed to measure voltages in the range -2.5V to +2.5V. Any voltages in excess of  $\pm 30\text{V}$  may cause permanent damage to the unit.

### **Mains (line) voltages**

No Pico products are designed for use with mains (line) 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/or computer and could cause injury to yourself or others.

You should assume that the product does not have a protective safety earth. Misuse 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.

## Licence conditions



The material contained in this software 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**

Because 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 such as life-support systems.

## Trademarks



**Windows**, **Excel** and **Visual Basic** are registered trademarks or trademarks of Microsoft Corporation in the USA and other countries. **Delphi** is a registered trademark of Borland Software Corporation. **Agilent VEE** is a registered trademark of Agilent Technologies, Inc. **LabView** is a registered trademark of National Instruments Corporation.

**Pico Technology Limited** and **PicoScope** are trademarks of Pico Technology Limited, registered in the United Kingdom and other countries.

**PicoScope** and **Pico Technology** are registered in the U.S. Patent and Trademark Office.

## Warranty



Pico Technology **warrants** upon delivery, and for a period of 24 months unless otherwise stated from the date of delivery, that the Goods will be free from defects in material and workmanship.

Pico Technology shall not be liable for a breach of the warranty if the defect has been caused by fair wear and tear, wilful damage, negligence, abnormal working conditions or failure to follow Pico Technology's spoken or written advice on the storage, installation, commissioning, use or maintenance of the Goods or (if no advice has been given) good trade practice; or if the Customer alters or repairs such Goods without the written consent of Pico Technology.

## Contacting Pico



**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](mailto:support@picotech.com)

Sales: [sales@picotech.com](mailto:sales@picotech.com)

**Web site:** [www.picotech.com](http://www.picotech.com)

- [Analog connector](#) Pin connections
- [Scaling](#) Converting from ADC counts to volts
- [Serial port configurations](#) IRQs etc.
- [Serial connection](#) Pin connections
- [Serial protocol](#) Information to help develop your own driver
- [Conversion times](#) Conversion time versus resolution
- [Modem operation](#) Using with telephone and radio modems

The Pico ADC-16 is a self-contained high-accuracy data logger for use with PCs and compatibles. It is connected to the RS-232 serial port, requiring no external power, and taking up no expansion slots.

You should have the following items supplied with the package:

- ADC-16 high-resolution serial-port logger
- Serial cable
- CD containing the software
- Installation manual

To use the ADC-16 you should connect the 9-pin connector on the ADC-16 to a spare serial port on your computer using the cable provided. If you have a 25-way serial port, use the 9 to 25 way adapter supplied.

To check that the unit is working:

1. Start up PicoLog for Windows
2. Select **File**
3. Select **New Settings**
4. At the **Recording** dialog, click **OK**
5. At the **Sampling** dialog, click **OK**
6. Set the converter type to **ADC-16**
7. Select the COM port to which you have connected the ADC-16
8. Click **OK**
9. At the **ADC-16 channels** dialog, double-click **CH1 Unused**
10. At the **Edit ADC-16 channel** dialog, click **OK**
11. Back at the **ADC-16 channels** dialog, click **OK**
12. The recorder view should now display the voltage on channel 1 (near 0 mV if nothing connected)
13. Next, connect a suitable voltage (e.g. from a 1.5 V battery) to the channel. Pin connections are marked on the ADC-16 and also listed [here](#).

## Specification



Resolution	between 8* and 16 bits plus sign (software programmable) * minimum 13 bits in PicoLog for Windows
Sampling rate	See <a href="#">Conversion times</a> (1.5 Hz)
Accuracy	±0.2%
Inputs	8 single-ended or 4 differential
Reference output	±5 V and 2.5 V
Linearity	0.003% (max.)
Input range	±2.5 V
Overvoltage protection	±30 V
Input impedance	1 megaohm
Operating temperature range	0 to 70 C
Temperature coefficient	33 ppm/ C
Input connector	D25 female
Output connector	D9 male to PC serial port

## Analog connector



Analog inputs are connected to the ADC-16 through the female D25 connector. The connections are as follows:

Pin	Function
1	Channel 1 (differential pair with pin 2)
2	Channel 2
3	Channel 3 (differential pair with pin 4)
4	Channel 4
5	Channel 5 (differential pair with pin 6)
6	Channel 6
7	Channel 7 (differential pair with pin 8)
8	Channel 8
9 - 11	Reserved for future expansion
12	Reference Output
14 - 25	Signal Ground (Computer 0 V)

Pin 12 is a stable 5 V output primarily intended for powering transducers such as thermistors. The amount of current that can be drawn out of this pin varies between computers. Almost all computers should be able to supply 1 mA and most several times this. If too much current is drawn from this pin, the voltage will drop below 5 V and the ADC-16's operation cannot be guaranteed.

No attempt should be made to draw more than 10 mA, as permanent damage to the ADC-16 could result. This means that loads of lower than 500 ohms must not be connected. Similarly, this pin must not be driven by external voltages.

If you wish to use the ADC-16 with your own software, it is possible to drive the ADC-16 directly- see the example programs [adc16a.c](#) and [adc16a.pas](#). It is, however, usually easier and more reliable to use the drivers provided.

Drivers are provided for the following operating systems:

[Windows XP \(SP2\)](#)

[Windows Vista](#)

Once you have installed the software, the **DRIVERS** directory contains a demo program which shows exactly how to drive the ADC-16, and an advanced driver which is easier to use and more powerful. It also contains a copy of this manual as a text file. See the [Readme.doc](#) file in the **DRIVERS** directory for the filenames.

The advanced driver routine is supplied as a Windows Dynamic Link Library for 32-bit programs, called [adc1632.dll](#).

The object file uses Pascal linkage conventions and does not require any compiler run-time routines: it can therefore be used with most C compilers.

The Windows DLLs can be used with C, C++, Delphi and Visual Basic programs: it can also be used with programs like Microsoft Excel, where the macro language is a form of Visual Basic, and with the National Instruments LabView application.

More than one application can access the Windows DLL at the same time, as long as the applications do not change the settings for channels that they are not using.

The following table specifies the function of each of the routines in the driver:

<b>Routine</b>	<b>Function</b>
<a href="#">adc16_open_unit</a>	Open the driver to use a specified serial port(s)
<a href="#">adc16_close_unit</a>	Close the port (ALWAYS DO THIS!)
<a href="#">adc16_set_channel</a>	Specify the resolution, single-ended/differential and filtering for a channel

<a href="#">adc16_get_value</a>	Get the most recent ADC reading from a channel
<a href="#">adc16_get_filtered_value</a>	Get the filtered ADC value for this channel
<a href="#">adc16_get_version</a>	Get the ADC-16 version number
<a href="#">adc16_poll</a>	Not normally necessary
<a href="#">adc16_get_cycle</a>	Returns the number conversion cycles since the unit was opened.

## adc16\_open\_unit



```
short int adc16_open_unit (short int port);
```

This routine specifies the serial port number with an ADC-16 unit. If you wish to use more than one ADC-16, you should call the routine once for each ADC-16.

The port must be 1 for COM1, 2 for COM2, etc.

Under Windows, this information is defined in your [WIN.INI](#) file, so it is not necessary to specify a value.

This routine returns TRUE if the driver successfully opens the ADC-16.

## adc16\_close\_unit



```
void adc16_close_unit (unsigned short int port);
```

This routine disconnects the driver from the specified serial port.

If you successfully open any serial ports, you **MUST** call `adc16_close_unit` for each port before you exit from your program. If you do not, your computer may misbehave until you next reboot it.

```
void adc16_set_channel (  
    unsigned short int port,  
    unsigned short int channel,  
    unsigned short int resolution,  
    unsigned short int single_ended,  
    unsigned short int filter_factor);
```

You should call this routine once for each channel that you would like to take readings from. You can do this any time after calling `adc16_driver_open`. It is permissible to change the settings as and when required.

The fewer channels are selected, the more frequently these channels will be updated: it takes about 1 second per active channel.

#### Arguments:

`port` specifies which serial port you wish to define a channel for.

`channel` specifies which channel you want to set the details for. It should be 1..8 for single-ended or 1,3,5,7 for differential.

The resolution specifies the conversion accuracy: it should be between 8 bits and 16 bits. See appendix A for the conversion times at each resolution.

`single_ended` specifies whether to use the channel as a single-ended input, or to use the channel together with the next channel as a differential input.

`single_ended = FALSE` is valid only for odd-numbered channels.

`filter_factor` controls the time constant of the filter. Each time the driver takes a reading from this channel, it updates the filtered value by adding a proportion of the difference between the measured and filtered values. The `filter_factor` sets the proportion that is added.

The `filter_factor` must be between 1 and 100. A `filter_factor` of 1 means add all of the difference (effectively no filtering) and 100 means add 1/100 of the difference (very slow filtering). A factor of 10 gives a time constant of about a minute when all channels are selected.

The effective resolution is increased by the `filter_factor`, and the noise floor is reduced by approximately the square root of the `filter_factor`.



## adc16\_get\_value



```
short int adc16_get_value (  
    long far      * value,  
    unsigned short int port,  
    short int     channel);
```

Once you open the driver, it constantly takes readings from the ADC-16. When you call this routine, it immediately sets value to the most recent ADC reading for the specified channel.

If a reading is available, it returns TRUE, otherwise it returns FALSE. It will normally return FALSE for a few seconds after you open the driver, until the driver has taken a reading from the specified channel.

### **Arguments:**

`port` specifies which serial port the ADC-16 is attached to.  
`channel` should be 1 for channel 1, 2 for channel 2 and so on.

## adc16\_get\_filtered\_value



```
short int adc16_get_filtered_value (  
    long far * value,  
    unsigned short int port,  
    short int channel);
```

This routine sets value to the filtered value for the specified channel. The filtered value is scaled at `filter_factor` times the ADC value.

## adc16\_get\_version



```
short int adc16_get_version (  
    short int far * version,  
    short int port);
```

This routine returns the version number of the ADC-16 attached to the specified port.

The upper byte of the version is always 16 for an ADC-16: the lower byte is the two hex digits of the version and release.

## adc16\_get\_cycle



```
short int adc16_get_cycle (  
    long far * cycle,  
    short int port);
```

This routine returns the number conversion cycles (conversions on all selected channels) since the unit was opened.

If you wish to get a value for every conversion, rather than taking values at fixed time intervals, you should periodically call this routine, then call [adc16\\_get\\_value](#) each time that cycle changes.

```
void adc16_poll (void)
```

The driver normally controls the ADC-16 using a Windows timer. Some programs (for example, Excel macros) prevent the timer from running, so it is necessary to call `adc16_poll` periodically while you are waiting for data.

It is not normally necessary to call this routine for well-behaved programs which let other applications service timer messages. Try without first, then call it if you do not appear to be getting data from the ADC-16.

## Windows



The Windows 32-bit driver is the file [ADC1632.DLL](#), installed in the [drivers](#) directory. If an application is unable to find the DLL, try moving the DLL to the Windows system directory.

## Windows

The C example program is a generic windows application; that is, it does not use Borland AppExpert or Microsoft AppWizard. To compile the program, create a new project for an Application containing the following files:

[adc16tes.c](#)

[adc16tes.rc](#)

either [adc1632.lib](#) (Borland 32-bit applications)

or [adc16ms.lib](#) (Microsoft Visual C 32-bit applications)

The following files must be in the same directory:

[adc16tes.rch](#)

[adc16.h](#)

[adc1632.dll](#) (All 32-bit applicaitons)

## C++

C++ programs can access all versions of the driver. If [adc16.h](#) is included in a C++ program, the PDEF1 macro expands to extern "C": this disables name-decoration and enables C++ routines to make calls to the driver routines using C headers.

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

If, however, you need to do something that is not possible using PicoLog, you can use an Excel macro to read in a set of data values. The Excel Macro language is similar to Visual Basic.

## **Excel 7**

The example [ADC1632.XLS](#) reads in 20 values from channels 1 and 2, one per second, and assigns them to cells A1..B20.

Copy the following files into your program directory:

adc16.dpr  
adc16.inc  
adc16fm.dfm  
adc16fm.pas

## Version 4 and 5 (32 bits)

The **DRIVERS** subdirectory contains the following files:

ADC1632.VBP

ADC1632.BAS

ADC1632.FRM

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 routine. The [adc16.llb](#) library in the [DRIVERS](#) subdirectory shows how to access this routine.

To use this routine, copy [adc16.llb](#) and [adc1632.dll](#) to your LabView [user.lib](#) directory.

You will then see the `adc16` sub-vi, and an example sub-vi which demonstrates how to use it. 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 COM1), and the channel (1 to 8).. The sub-vi returns the voltage in millivolts.

The example routine [adc16.vee](#) is in the [drivers](#) subdirectory. You will also need a copy of the following file:

[adc16.vh](#)

The example shows how to collect a readings continuously from the ADC-16.

This example was written by Gatespace AB and all documentation and the [adc.jar](#) class is found in the [drivers](#) directory.

The example is supplied compressed as [javaadc.tgz](#).

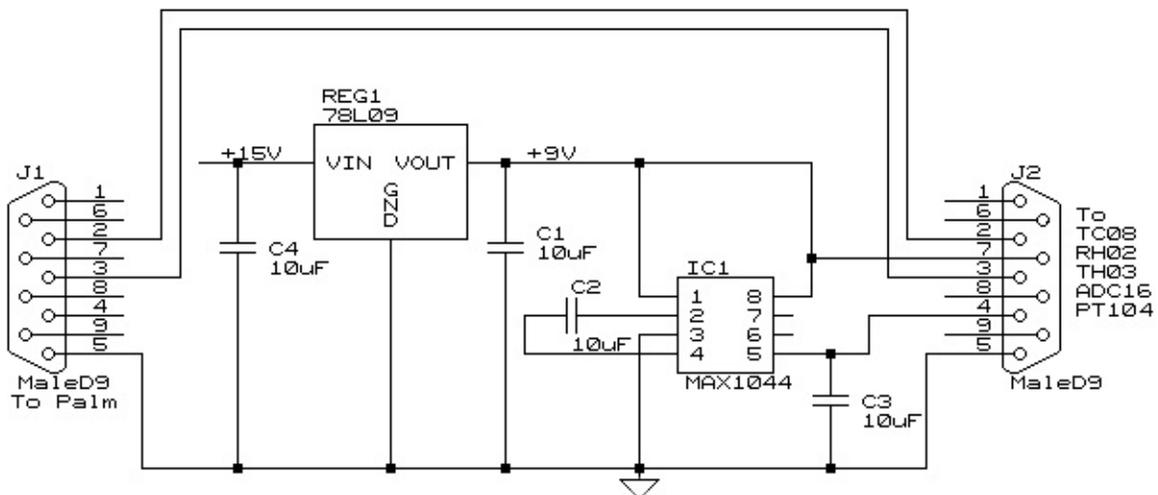
This example is a basic driver that reads voltages from the ADC-16. The driver has been tested with the following Palm Pilots:

- US Robotics Palm Profesional
- Palm 3
- Palm V/Vx
- Palm 3c

The file [adc16.zip](#) is located in the directory. Further information can be found at:

<http://www.picotech.com/applications/palm.html>

It should be noted that due to the power limitations with the Palm Pilot it is necessary to construct a power adapter to power the ADC-16. A suggested design for the power adapter is:



See the [adc16.tar](#) file for more information.

As PicoLog software displays readings in volts, this section is only required if you are writing your own software.

The ADC-16 generates readings which are in ADC counts. The scaling of these values depends on the resolution. In  $n$ -bit mode, the maximum input voltage (2.5 V) is represented by  $2^n-1$ , and the minimum input voltage (-2.5 V) is represented by  $-(2^n-1)$ . To convert an ADC value to a voltage, multiply by 2.5 and divide by the maximum ADC value at the resolution that you are using.

<b>Resolution</b>	<b>Maximum ADC value</b>
8	255
9	511
10	1,023
11	2,047
12	4,095
13	8,191
14	16,383
15	32,767
16	65,535

## Serial port configurations



Serial drivers need to know two things about a serial port in order to operate the port correctly - its base address and its interrupt number (IRQ).

Most PCs have two serial ports fitted, COM1 and COM2. All software works correctly when just these ports are fitted. There is also *de facto* standard for two more serial ports - COM3 and COM4. The addresses of these ports are unique, but the IRQ numbers are the same as for COM1 and COM2.

Port	Base address	Interrupt	Standard?
COM1	3F8	4	Yes
COM2	2F8	3	Yes
COM3	3E8	4	De facto
COM4	2E8	3	De facto
COM5...			No

Some serial port cards are capable of sharing an interrupt, and some software is capable of driving a serial port that is sharing an interrupt, but it is likely that either hardware or software cannot handle interrupt sharing. If you install additional COM ports as COM3 or COM4, using the default IRQs, these may interfere with devices connected to COM1 and COM2. For example, an ADC-16 connected to COM3 may cause a mouse connected to COM1 not to function correctly.

The better quality serial port cards make it possible to use other interrupts, such as IRQ5 or IRQ10. There are no hard and fast rules about what IRQs are used by specific devices, so it is necessary to study the technical information supplied with your computer and the proceed by trial and error. The following table of typical IRQ usages may be helpful.

IRQ0	55ms timer
IRQ1	Keyboard
IRQ2	Cascaded IRQ8..F
IRQ3	COM2
IRQ4	COM1

IRQ5	LPT2
IRQ6	Diskette
IRQ7	LPT1
IRQ8	Real time clock
IRQ9	Unavailable
IRQ10	general
IRQ11	general
IRQ12	general
IRQ13	Co-processor
IRQ14	Hard disk
IRQ15	general

If you wish to use non-standard IRQ values with Windows, you should find the following lines in [C:plw.ini](#), and modify the details to the values that you have selected:

```
COM4Base=02E8  
COM4Irq=3  
COM3Base=03E8  
COM3Irq=4
```

It is possible to buy multi-channel serial port cards (normally four or eight channels) where the serial ports use the same interrupt. There is no standard for these cards. These cards are normally supplied with Windows drivers, and so the ADC-16 DLL should work correctly once the Windows drivers for the serial card are installed.

## Serial connection



The information presented here is necessary only if you wish to write your own driver, or to connect the ADC-16 to the PC in some unusual way (for example, via a radio modem).

The ADC-16 uses the following RS-232 data lines (pin connections as on ADC-16)

Pin	Name	Usage
3	TX	Data from the PC to the ADC-16
2	RX	Data from the ADC-16 to the PC
7	RTS	Held at a positive voltage ( $> 7$ V) to power the ADC-16
5	GND	0 V line
4	DTR	Held at a negative voltage ( $< -7$ V) to power the ADC-16

The driver powers up the ADC-16 by enabling RTS and disabling DTR to provide the correct polarity power supply. If these are set incorrectly no damage will occur to either PC or ADC-16.

About a second after powering on the ADC-16, the driver can communicate with the ADC-16 as a normal RS232 device. The ADC-16 operates at 9600 baud with 1 stop bit and no parity.

The driver controls the ADC-16 using the following sequence

1. Switch RTS on and DTR off to provide power.
2. Wait for more than 1 second for the ADC-16 to settle
3. Send an single control byte to the ADC-16
4. Wait for the 3 byte response from the ADC-16

Steps 3 and 4 are repeated for each measurement.

The ADC-16 signals the end of conversion by sending three bytes. No data should be sent to the ADC-16 during the conversion, as it may be lost or corrupted.

## Control byte description

The control byte tells the ADC-16 the information it needs to carry out a conversion. It has the following format, where bit 7 is the MSB and bit 0 is the LSB:

Bits	Function	Value
7 - 5	Set the channel	000 = channel 1 001 = channel 2 ..... 110 = channel 7 111 = channel 8
4 - 1	Select Resolution	0111 = 8 bit 1000 = 9 bit ..... 1110 = 15 bit 1111 = 16 bit
0	Mode	1 = single ended

	0 = differential
--	------------------

If the mode bit is 1, each channel voltage will be measured with respect to ground (single ended operation). If the mode bit is 0 then adjacent channels act as differential pairs. Differential operation can be useful if problems due to earth loops are encountered.

If you select channel 1 and differential operation, the ADC-16 will measure the voltage between channels 1 and 2. Similarly selecting channel 3 will cause the voltage to be measured between channels 3 and 4. Whilst in differential mode, selecting even channel numbers may give incorrect results.

The control byte 00000001 is a request for the version number: see below for details of the version response. The following examples show complete control bytes and their effects:

Binary	Hex	Effect
000 1111 1	1F	channel 1, 16 bits, single ended
110 0111 0	CE	channels 7 and 8, 8 bits, differential mode.
000 0000 1	01	request version number

### Response format

On receipt of a control byte containing a valid data request, the ADC-16 will start a conversion cycle. At the end of this conversion, the ADC-16 will respond by sending three bytes. Note that three bytes are sent even for 8-bit readings.

Byte	Contents	Value
1	Sign	value $\geq$ 0: ASCII + (hex 2B) value $\leq$ : ASCII - (hex 2D)
2	Result MSB	Binary
3	Result LSB	Binary

For example, the value +41349 (which is hex 85A1) would be sent as hex 2B, 85, A1.

On receipt of a control byte containing a valid version request, the ADC-16 will respond immediately with the version details in the following format:

Byte	Contents	Value
1	ADC type	16 (hex 10)

2	version	decimal version number
---	---------	------------------------

## Conversion times



The ADC-16's conversion time is affected by both the resolution and the applied voltage. The figures listed below are worst case for each resolution. If a severe overload is applied to the input of the ADC-16 it will shut itself down: no further communication with the ADC-16 will be possible until approximately 1 second after the overload is removed.

<b>Resolution (bits)</b>	<b>Conversion Time(ms)</b>
8	6.6
9	8.9
10	14
11	23
12	41
13	78
14	151
15	298
16	657

## Modem operation



The ADC-16 is normally connected directly to the computer, but it is also possible to access the ADC-16 via a modem using the Windows driver.

It is necessary to provide power to the ADC-16, either by instructing the modem to provide power or by connecting a power supply directly to the ADC-16. See [serial connections](#) for information.

For some radio modems, there is a delay between sending text to the modem and its arrival at the other end, and a similar delay for the response from the TC08. If, for example, the maximum possible delay is 150 ms each way, 300 ms total, the following text should be added to `WIN.INI` so that the driver waits longer for each response.

```
[ADC16]
Turnround=300
```

**Warning:** In order to comply with current legislation, use only radio modems which comply with the RTTE directive.