ADC-20/ADC-24

High-Resolution Data Loggers

User's Guide

Contents

1   Introduction

1.1   Overview

The ADC-20 and ADC-24 High-Resolution Data Loggers are multichannel, high-accuracy USB data loggers for use with PCs. They require no external power supply and take up no expansion slots. They come complete with PicoLog, a program offering all the features of a stand-alone data logger.

You should have the following items in your ADC-20 or ADC-24 package:

- ADC-20 or ADC-24 High-Resolution Data Logger
- Pico Technology Software and Reference CD
- Installation Guide

The CD includes driver software that enables you to write your own programs to control the data logger, using a variety of popular programming languages.

The hardware and software are compatible with Microsoft Windows XP (SP3) or later.

Optional accessories

- PP310 ADC-20/ADC-24 Terminal Board

2   Notices

2.1   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-20 and ADC-24 are designed to measure voltages in the range ±2.5 volts, but are protected against overvoltages of ±30 volts. Any voltages outside the overvoltage protection range may cause permanent damage to the unit.

Mains (line) voltages. The ADC-20 and ADC-24 data loggers are not designed for use with mains (line) voltages.

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 minimize 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 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.

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.

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

2.2   Legal information

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 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. As 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 license is that it excludes usage in mission-critical applications, such as 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.

2.3   CE notice

The ADC-20 and ADC-24 meet the intent of [EMC](#) directive 89/336/EEC and meets the EN61326-1 (1997) Class B Emissions and Immunity standard.

The ADC-20 and ADC-24 also meet the intent of the Low Voltage Directive and meet the BS EN 61010-1:2001 IEC 61010-1:2001 (safety requirements for electrical equipment, control, and laboratory use) standard.

A Declaration of Conformity is available from Pico Technology Ltd.

2.4   FCC notice

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his or her own expense.

For safety and maintenance information see the [safety warning](#).

A Declaration of Conformity is available from Pico Technology Ltd.

2.5   Trademarks

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

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

Windows and Excel are registered trademarks of Microsoft Corporation in the USA and other countries.

2.6   Updates

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.

2.7 Support and refunds

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. If you are still dissatisfied, please return the product and software to your supplier within 14 days of purchase for a full refund.

3   Getting started

3.1   Installing the software

Before you connect the ADC-20 or ADC-24 to your computer for the first time, you must install the software supplied on the CD. Insert the Software and Reference CD, then follow the Install Software link.

A 32-bit driver is installed automatically along with the PicoLog software. Alternatively, you can download the software development kit (SDK) from our website at www.picotech.com: this is available with 32-bit or 64-bit drivers.

3.2   Connecting the data logger

You must install the software before plugging in the data logger. When you have installed the software, connect the data logger's USB cable to a spare USB port on your computer and wait until Windows displays the message "Device is ready to use".

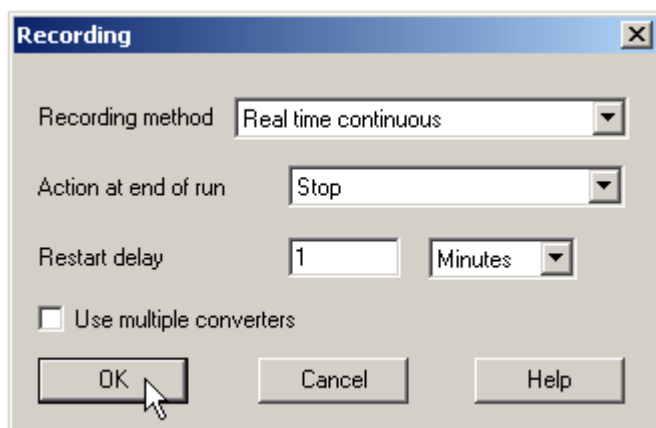## 3.3   Starting PicoLog

To check that the data logger is working, start the PicoLog application as follows:

1. Start PicoLog. (Click the PicoLog icon: ![PicoLog Recorder] in your Windows Start menu).
2. Click File.
3. Point to New settings.

4. At the Recording dialog, click OK.

5. At the Sampling Rate dialog, click OK.
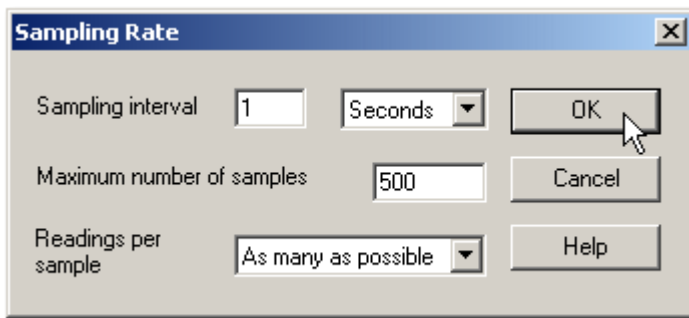
6. In the Converter details dialog, set the Converter type to High Resolution Data Logger. The device type and serial number should appear in the USB Devices list, and the USB enumeration progress indicator should gradually move towards 100%. If the progress indicator does not start moving, disconnect and reconnect the unit and then press Refresh.
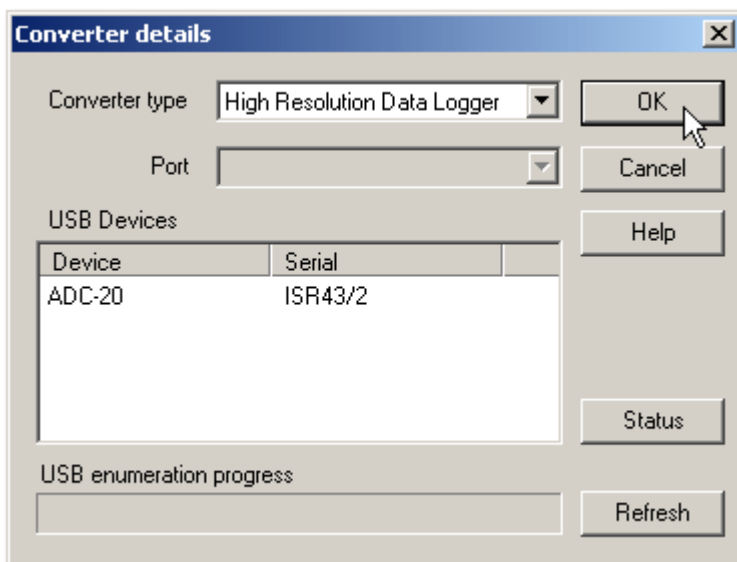
7. Click OK.
8. At the ADC-20 (or ADC-24) channels dialog, double-click on "Channel 1".
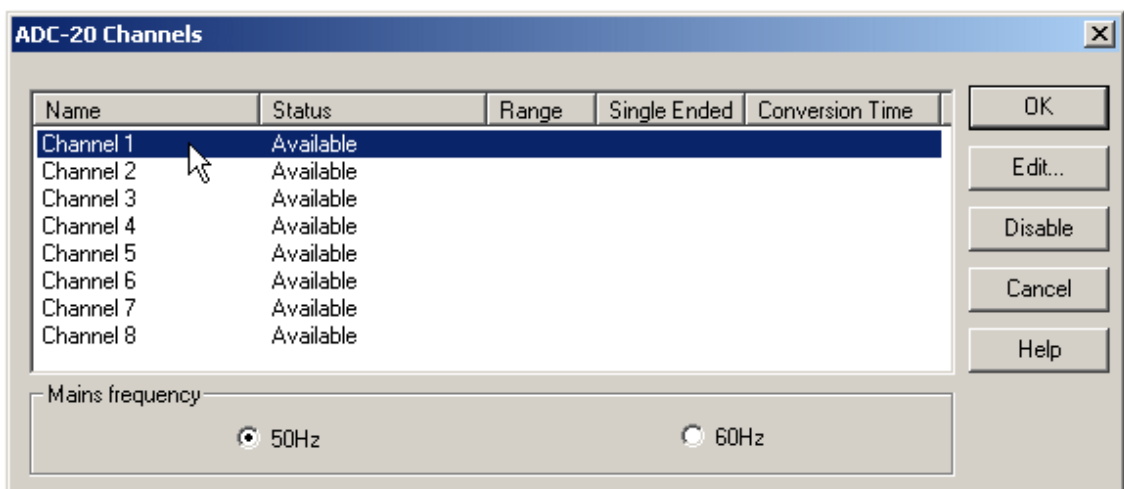
9. At the Edit ADC-20 (or ADC-24) Channel dialog, click OK.
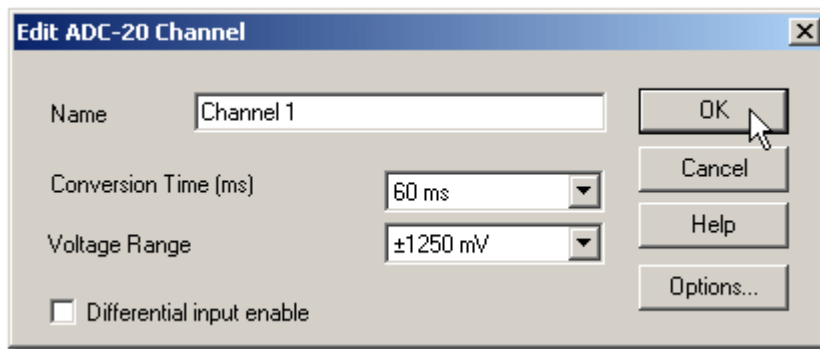
10. Back at the ADC-20 (or ADC-24) Channels dialog, click OK.
11. The PLW Recorder view should now display the voltage on channel 1 (near 0 mV if nothing connected).
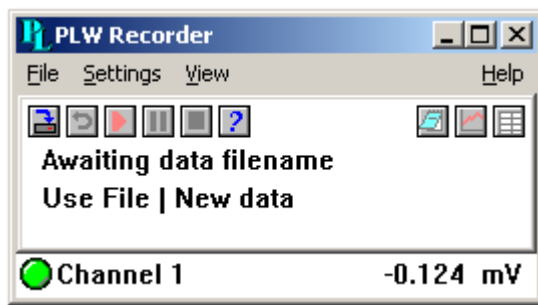
12. Connect a suitable voltage (for example, from a 1.5 V battery) to the channel. Pin connections are marked on the logger and also listed in the Analog connector topic.

4   About the unit

4.1   Introduction

The ADC-20 and ADC-24 High-Resolution Data Loggers offer the ultimate in precise and accurate readings. Features such as true differential inputs, galvanic isolation and software-selectable sample rates all contribute to a superior noise-free resolution.

The ADC-20 is equipped with a 20-bit A/D converter, and can maintain a gain error of 0.2%. The four true differential inputs may be configured as eight single-ended inputs or any combination in between, such as two differential and four single-ended. The ADC-24 is equipped with a 24-bit A/D converter, and can maintain a gain error of 0.1%.

The eight true differential inputs may be configured as 16 single-ended inputs or any combination in between. Power and connection to a PC or laptop is through a USB 1.1, USB 2.0 or 3.0 port. Using the supplied PicoLog software, you can record, monitor and analyze collected data, even exporting to third-party applications such as Microsoft Excel.

4.2   Specifications

	ADC-20	ADC-24

Input channels
    Differential
    Single-ended
	
	Configurable up to 4
	Configurable up to 8
		
		Configurable up to 8
		Configurable up to 16

Conversion time per channel	60 ms, 100 ms, 180 ms, 340 ms, 660 ms	
Input voltage ranges	2 ranges ±1250 mV and ±2500 mV	7 ranges ±39 mV to ±2500 mV
Resolution	20 bits	24 bits
Noise-free resolution	See table below	
Overvoltage protection	±30 V between any input and common	

Common-mode range
    Channel to common
    Channel to common
    Common to earth ground
	±1.9 V (±39 mV to ±1250 mV ranges)
	±3.0 V (±2500 mV range)
	±30 V	

Common-mode rejection ratio
    Channel to common
    Common to earth ground
	
	95 dB (DC to 60 Hz)
	>125 dB (DC to 60 Hz)	

Noise rejection	120 dB typical at 50 to 60 Hz	
Input impedance	2 M (differential), 1 M (single-ended)	
Gain error from source mpedance	0.5 ppm per ohm source impedance due to input impedance	
Gain error	0.2%	0.1% (0.2% at ±2500 mV)
Offset error	6 µV (±39 mV range) 7 µV (±78 mV range) 9 µV (±156 mV range) 13 µV (±313 mV range) 20 µV (±625 mV range) 36 µV (±1250 mV range) 400 µV (±2500 mV range)	

	ADC-20	ADC-24
Digital I/O		4
Output level, high		> 2.40 V
Output level, low	None	< 0.40 V
Input level, high		> 2.20 V
Input level, low		< 0.88 V
Isolation (input to input)	None	
Isolation (input to ground)	Galvanic, up to ±30 V AGND and DGND isolated	
Reference output	+2.5 V ±2.5 mV @ 2 mA	
Power outputs	+5 V ±1.0 V @ 2 mA -5 V ±1.5 V @ 2 mA	

Environmental conditions
   Operating temperature
   Quoted input accuracy
   Storage temperature
   Operating humidity
   Storage humidity
	0 °C to 45 °C
	20 °C to 30 °C
	-20 °C to 60 °C
	5% to 80% RH, non-condensing
	5% to 95% RH, non-condensing	

Recommended calibration interval	1 year	
PC connection	USB 1.1 (USB 2.0 compatible) Cable length approx. 4.4 m (13.8 ft)	
Input connector	DB25 female	
Power supply	100 mA (max.) from USB port	
Dimensions	13.5 cm x 18.4 cm x 3.6 cm (5.31 in x 7.24 in x 1.41 in)	
Weight	Approx. 505 g (17.8 oz)	

4.3   Analog connector

Analog inputs are connected to the ADC-20 and ADC-24 through the female DB25 connector. The connections are as follows:
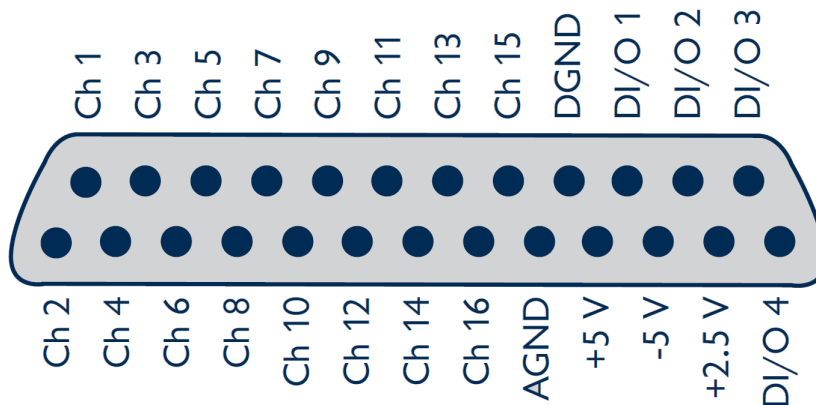
Pin	Function	Pin	Function
1	Channel 2 (Channel 1-)	14	Channel 1
2	Channel 4 (Channel 3-)	15	Channel 3
3	Channel 6 (Channel 5-)	16	Channel 5
4	Channel 8 (Channel 7-)	17	Channel 7
5	Channel 10 (Channel 9-)	18	Channel 9
6	Channel 12 (Channel 11-)	19	Channel 11
7	Channel 14 (Channel 13-)	20	Channel 13
8	Channel 16 (Channel 15-)	21	Channel 15
9	Analog Ground	22	Digital Ground
10	+5 volts	23	Digital I/O 1
11	-5 volts	24	Digital I/O 2
12	+2.5 volts	25	Digital I/O 3
13	Digital I/O 4		

Note 1:   Channel numbers in brackets apply when the input is set to differential mode.

Note 2:   Channels 9 to 16 exist only on the ADC-24.

Note 3:   Digital I/Os 1 to 4 exist only on the ADC-24.

Note 4:   Pins 10 and 11 are low-current outputs for powering small sensors. Do not exceed the current limits given in the Specifications table.

Note 5:   The analog and digital grounds are not connected together inside the unit. You should not connect them together externally, as this would degrade the accuracy of the unit.

Note 6:   For easy connection to the DB25 connector, we recommend that you use the ADC-20/ADC-24 terminal board.

4.4   Noise-free resolution and conversion time

The table below shows the number of noise-free bits of resolution for the full range of conversion times.

Conversion time per channel	Voltage range & Noise-free resolution (bits)						
	±39 mV	±78 mV	±156 mV	±313 mV	±625 mV	±1250 mV	±2500 mV
	ADC-24 only					ADC-20 and ADC-24	
660 ms	17	18	19	20	20	20	20
340 ms	17	18	19	19	19	20	20
180 ms	16	17	18	19	19	19	19
100 ms	16	17	18	18	18	19	19
60 ms	15	16	17	18	18	18	18

4.5   ADC-20/ADC-24 terminal board

For easy connection to the DB25 connector, we recommend that you use the ADC-20/ADC-24 terminal board, part number PP310. This has screw terminals to allow you to connect wires to all of the data logger's inputs and outputs without soldering. It also has space for voltage-divider resistors, a temperature sensor and a quad op-amp.

See the *ADC-20/ADC-24 Terminal Board User's Guide* for instructions on connecting signals to the terminal board. If you are not using a terminal board, you can still refer to the Terminal Board User's Guide for information on connecting various types of signal to the data logger. http://www.elso.sk/product.php?id_product=2136

4.6   LED

The ADC-20 and ADC-24 have an LED, next to the entry point of the USB cable, that you can use to verify that the unit is working. The LED flashes whenever the unit is taking readings. It also flashes briefly during "enumeration", the process that PicoLog uses to detect all Pico USB devices plugged in to the computer.

5   Programmer's reference

5.1   Recording methods

The ADC-20/ADC-24 driver provides three methods of recording data. All these methods support USB 1.1.

- Streaming
  The driver constantly polls the device, and samples are placed in a buffer until retrieved by your application. Precise sample timing is controlled by the unit.

- Single Value (blocking)
  You make a single request for a sample, blocking the calling thread, and when the sample has been received the driver returns the value to your application.

- Single Value (non-blocking)
  You make a single request for a sample without blocking the calling thread, and when the sample has been received the driver returns the value to your application.

5.2   Windows driver

The `picohrdl.dll` dynamic link library (DLL) in the `lib` subdirectory of your SDK installation is a driver that allows you to program your ADC-20 or ADC-24 data logger. It is supplied in 32-bit and 64-bit versions. The driver exports the function definitions in standard C format, but this does not limit you to programming in C. You can use the API with any programming language that supports standard C calls. It can also be used with programs like Microsoft Excel. The driver supports all Windows operating systems from Windows 7 onwards.

5.3   Scaling

To convert from ADC values to volts, first obtain the minimum and maximum ADC values for the selected channel by calling the HRDLGetMinMaxAdcCounts function in the driver. Next, scale the ADC value to the voltage range you specified when you called HRDLSetAnalogInChannel. You can calculate the voltage range programmatically by using

   Vmax = 2500 mV / (2^r)

where r is the range constant you supplied to HRDLSetAnalogInChannel (0 for ±2500 mV, 1 for ±1250 mV and so on).

You can then use Vmax to calculate the scaled voltage, V, with the following formula

   V = (raw_ADC_value / max_ADC_Value) * Vmax

where raw_ADC_value is the reading from the device and max_ADC_value is the max ADC value for the device obtained from HRDLGetMinMaxAdcCounts.

5.4   Driver functions

The following sections describe the functions available to an application using the ADC-20 and ADC-24. All functions are C functions using the standard call naming convention (`__stdcall`) and are exported with both decorated and undecorated names.

Function	Description
HRDLCloseUnit	Shuts down unit
HRDLCollectSingleValueAsync	Starts the unit sampling one value without blocking the calling application's flow
HRDLGetMinMaxAdcCounts	Returns the maximum and minimum ADC count available for the device
HRDLGetNumberOfEnabledChannels	Returns the number of analog channels enabled
HRDLGetSingleValue	Takes one sample for the specified channel at the selected voltage range and conversion time
HRDLGetSingleValueAsync	Retrieves the reading when the HRDLCollectSingleValueAsync has been called
HRDLGetTimesAndValues	Returns the requested number of time-stamped samples for each enabled channel
HRDLGetUnitInfo	Writes unit information to a character string
HRDLGetValues	Returns the requested number of samples for each enabled channel
HRDLOpenUnit	Opens data logger unit
HRDLOpenUnitAsync	Opens unit without blocking the calling thread
HRDLOpenUnitProgress	Checks the progress of an asynchronous open operation
HRDLReady	Indicates when readings are ready to be collected
HRDLRun	Starts the device sampling and storing the samples into the driver's buffer
HRDLSetAnalogInChannel	Enables or disables the analog channel
HRDLSetDigitalIOChannel	Sets a digital output or input (ADC-24 only)
HRDLSetInterval	Sets the sampling time interval
HRDLSetMains	Sets the mains noise rejection frequency
HRDLStop	Stops the device when streaming

5.4.1   HRDLCloseUnit

```
int16_t HRDLCloseUnit
(
    int16_t handle
)
```

Shuts down a [data logger](#) unit.

Arguments

`handle`   The handle, returned by [HRDLOpenUnit](#), of the unit being closed

Returns

1   if a valid handle is passed
0   if not

5.4.2   HRDLCollectSingleValueAsync

```
int16_t HRDLCollectSingleValueAsync
(
    int16_t handle,
    int16_t channel,
    int16_t range,
    int16_t conversionTime,
    int16_t singleEnded
)
```

This function starts the unit sampling one value without blocking the calling application's flow. Used in conjunction with HRDLGetSingleValueAsync and HRDLReady.

Arguments

handle   Handle returned by HRDLOpenUnit

channel   Channel number to convert. If the channel is not valid then the function will fail.

range   The voltage range to be used. If the range is not valid, the function HRDLGetSingleValueAsync will return 0.

conversionTime   The time interval in which the sample should be converted. If the conversion time is invalid,the function HRDLGetSingleValueAsync will fail and return 0.

singleEnded   The type of voltage to be measured:

                  0: differential
                  nonzero: single-ended

Returns

1   if a valid handle is passed and the settings are correct
0   if not

5.4.3   HRDLGetMinMaxAdcCounts

```
int16_t HRDLGetMinMaxAdcCounts
(
    int16_t handle,
    int32_t * minAdc,
    int32_t * maxAdc,
    int16_t channel
)
```

This function returns the maximum and minimum ADC count available for the device referenced by handle.

Arguments

handle   Handle returned by [HRDLOpenUnit](#)

minAdc   Pointer to an `int32_t`, used to return the minimum ADC count available for the unit referred to by handle

maxAdc   Pointer to an `int32_t`, used to return the maximum ADC count available for the unit referred to by handle

channel   Channel number for which maximum and minimum ADC count are required

Returns

1   if a valid handle is passed
0   if not

5.4.4   HRDLGetNumberOfEnabledChannels

```
int16_t HRDLGetNumberOfEnabledChannels
(
    int16_t handle,
    int16_t * nEnabledChannels
)
```

This function returns the number of analog channels enabled.

Arguments

handle   Handle returned by HRDLOpenUnit

nEnabledChannels   Pointer to an `int16_t`, where the number of channels enabled
                        will be written

Returns

1   if a valid handle is passed
0   if not

5.4.5   HRDLGetSingleValue

```
int16_t HRDLGetSingleValue
(
    int16_t handle,
    int16_t channel,
    int16_t range,
    int16_t conversionTime,
    int16_t singleEnded,
    int16_t * overflow
    int32_t * value
)
```

This function takes one sample for the specified channel at the selected voltage range and conversion time.

Arguments

`handle`   Handle returned by HRDLOpenUnit.

`channel`   The channel number to convert.

                 ADC-20: 1 to 8
                 ADC-24: 1 to 16

                 If the channel is not valid then the function will fail and return 0.

`range`   The voltage range to be used. See HRDLSetAnalogInChannel for possible values. If the range is not valid, the function will return 0.

`conversionTime` The time interval in which the sample should be converted. See HRDLSetInterval for possible values. If the conversion time is invalid, the function will fail and return 0.

`singleEnded`   The type of voltage to be measured.

                 0: differential
                 nonzero: single-ended

`overflow`   Pointer to a bit field that indicates when the voltage on a channel has exceeded the upper or lower limits.

                 Bit 0: Channel 1
                 …
                 Bit 15: Channel 16

`value`   Pointer to an `int32_t` where the ADC value will be written.

Returns

1   if a valid handle is passed and settings are correct
0   if not

If the function fails, call HRDLGetUnitInfo with `info = HRDL_ERROR (7)` to obtain the error code. If the error code is `HRDL_SETTINGS (5)`, then call HRDLGetUnitInfo again with `info = HRDL_SETTINGS_ERROR (8)` to determine the settings error.

5.4.6   HRDLGetSingleValueAsync

```
int16_t HRDLGetSingleValueAsync
(
    int16_t handle,
    int32_t * value,
    int16_t * overflow
)
```

This function retrieves the reading when the HRDLCollectSingleValueAsync has been called.

Arguments

handle   Handle returned by HRDLOpenUnit

value   Pointer to an `int32_t` where the ADC value will be written

overflow   Pointer to a value that indicates when the voltage on a channel has exceeded the upper or lower limits.

> Bit 0: Channel 1
> …
> Bit 15: Channel 16

Returns

1   if a valid handle is passed and the function succeeds
0   if not

Sample code

Code extract to get a single value reading without blocking the calling thread:

```
void main()
{
  BOOL  bConversionFinished = FALSE;
  int16_t channelNo;
  int32_t value;
  int16_t handle;

  // Open and initialize the unit
     .
     .
     .


  // Set the channel parameters
  channelNo = HRDL_ANALOG_IN_CHANNEL_1;
  range = HRDL_2500_MV;
  singleEnded = TRUE;
  bConversionFinished = FALSE;

  while (true)
  {
    PollSingleValue(handle,
                    &bConversionFinished,
                    &value,
                    channelNo,
                    range,
                    singleEnded);
```

```
        if (bConversionFinished == TRUE)
        {
          // Do something with the value
          channelNo++;
          // This would be HRDL_ANALOG_IN_CHANNEL_8 for the ADC-20
          if (channelNo > HRDL_ANALOG_IN_CHANNEL_16)
          {
            channelNo = HRDL_ANALOG_IN_CHANNEL_1;
          }
        }
        else
        {
          // Do something else while waiting for the reading from the
          // unit
        }
      }
    }


    void PollSingleValue(int16_t handle,
                         BOOL *bConversionFinished,
                         int32_t *lValue,
                         int16_t channel,
                         int16_t range,
                         int16_t singleEnded)
    {
      static BOOL bStartConversion = FALSE;
      int16_t      overflow;

      // Test to see if the conversion has finished
      if (bStartedConversion)
      {
        if (HRDLReady(handle))
        {
          HRDLGetSingleValueAsync(handle, lValue, &overflow);
          bConversionFinished = TRUE;
          bConversionStarted = FALSE;
        }
      }

      // Test to see if no conversion is in progress
      if (!bStartedConversion)
      {
        // Start the conversion going
        bStartedConversion = HRDLCollectSingleValueAsync(handle,
                                                         channel,
                                                         range,
                                                         conversionTime,
                                                         singleEnded);

        bConversionFinished = TRUE;
      }
    }
```

5.4.7   HRDLGetTimesAndValues

```
int32_t HRDLGetTimesAndValues
(
    int16_t handle,
    int32_t * times,
    int32_t * values,
    int16_t * overflow,
    int32_t noOfValues
)
```

This function returns the requested number of samples for each enabled channel and the times when the samples were taken, so the `values` array needs to be (number of values) x (number of enabled channels). When one or more of the digital IOs are enabled as inputs, they count as one additional channel. The function informs the user if the voltages for any of the enabled channels have overflowed.

Arguments

`handle`   Handle returned by HRDLOpenUnit.

`times`   Pointer to an `int32_t` where times will be written.
`values`   Pointer to an `int32_t` where sample values will be written. If more than one channel is active, the samples are interleaved. If digital channels are enabled then they are always the first values. See table below for the order in which data are returned.

`overflow`   Pointer to an `int16_t` indicating any inputs that have exceeded their maximum voltage range. Channels with overvoltages are indicated by a high bit, with the LSB indicating channel 1 and the MSB channel 16.

`noOfValues`   The number of samples to collect for each active channel

Returns

A non-zero number if successful indicating the number of values returned,

0 if the call failed or no values available

Ordering of returned data (example)

When two analog channels (e.g. 1 and 5) are enabled and a digital channel is set as an input, the data are returned in the following order:

```
Sample No: 0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 . n-3 n-2 n-1
Channel:   DI 1  5  DI 1  5  DI 1  5  DI 1  5  DI 1  5  . DI  1   5
```

where n represents the value returned by the function and DI the digital inputs.

The channels are always ordered from channel 1 up to the maximum channel number (ADC-24: channel 16, ADC-20: channel 8). If one or more digital channels are set as inputs then the first sample contains the digital channels.

Digital inputs

The digital channels are represented by a binary bit pattern with 0 representing off, and 1 representing on. Digital input 1 is in bit 0.

5.4.8   HRDLGetUnitInfo

```
int16_t HRDLGetUnitInfo
(
    int16_t handle,
    int8_t  * string,
    int16_t stringLength,
    int16_t info
)
```

This function writes information about the data logger to a character string. If the logger fails to open, only `info = HRDL_ERROR (7)` is available to explain why the last open unit call failed. When retrieving the driver version, the handle value is ignored.

Arguments

`handle`   Handle to the device from which information is required. If an invalid handle is passed, the error code from the last unit that failed to open is returned (as if `info = HRDL_ERROR`), unless `info = HRDL_DRIVER_VERSION` and then the driver version is returned.

`string`   Pointer to the `int8_t` string buffer in the calling function where the unit information string (selected with `info`) will be stored. If a null pointer is passed, no information will be written.

`stringLength` Length of the `int8_t` string buffer. If the string is not long enough to accept all of the information, only the first `stringLength` characters are returned.

`info`   Enumerated type (listed below) specifying what information is required from the driver.

Returns

The length of the string written to the `int8_t` string buffer, `string,` by the function.

If one of the parameters is out of range, or a null pointer is passed for `string,` the function will return zero.

Values of `info`

Value of `info`	Description	Example
`HRDL_DRIVER_VERSION (0)`	The version of `picohrdl.dll`, `libpicohrdl.dylib` or `libpicohrdl.so`	1.0.0.1
`HRDL_USB_VERSION (1)`	The type of USB to which the unit is connected	1.1
`HRDL_HARDWARE_VERSION (2)`	The hardware version of the HRDL attached	1
`HRDL_VARIANT_INFO (3)`	Information about the type of HRDL attached	24
`HRDL_BATCH_AND_SERIAL (4)`	Batch and serial numbers of the unit	CMY02/116
`HRDL_CAL_DATE (5)`	Calibration date of the unit	09Sep05
`HRDL_KERNEL_DRIVER_VERSION (6)`	Kernel driver version	

Value of `info`	Description	Example
`HRDL_ERROR (7)`	One of the error codes listed in Error codes below	4
`HRDL_SETTINGS (8)`	One of the error codes listed in Settings Error Codes below	

Error codes (when `info` = `HRDL_ERROR`)

Error code	Description
`HRDL OK (0)`	The unit is functioning correctly
`HRDL_KERNEL_DRIVER (1)`	The `picopp.sys` file is to old to support this product
`HRDL NOT FOUND (2)`	No data logger could be found
`HRDL CONFIG FAIL (3)`	Unable to download firmware
`HRDL_ERROR_OS_NOT_SUPPORTED (4)`	The operating system is not supported by this device
`HRDL_MAX_DEVICES (5)`	The maximum number of units allowed are already open

Settings Error Codes (when `info` = `HRDL_SETTINGS`)

Settings Error Code	Description
`SE_CONVERSION_TIME_OUT_OF_RANGE (0)`	The conversion time parameter is out of range
`SE_SAMPLEINTERVAL_OUT_OF_RANGE (1)`	The sample time interval is out of range
`SE_CONVERSION_TIME_TOO_SLOW (2)`	The conversion time chosen is not fast enough to convert all channels within the sample interval
`SE_CHANNEL_NOT_AVAILABLE (3)`	The channel being set is valid but not currently available
`SE_INVALID_CHANNEL (4)`	The channel being set is not valid for this device
`SE_INVALID_VOLTAGE_RANGE (5)`	The voltage range being set for this device s not valid
`SE INVALID PARAMETER (6)`	One or more parameters are invalid
`SE_CONVERSION_IN_PROGRESS (7)`	A conversion is in progress for a single asynchronous operation
`SE_COMMUNICATION_FAILED (8)`	The PC has lost communication with the device
`SE_OK (9)`	All settings have been completed successfully

5.4.9 HRDLGetValues

```
int32_t HRDLGetValues
(
    int16_t handle,
    int32_t * values,
    int16_t * overflow,
    int32_t noOfValues
)
```

This function returns the requested number of samples for each enabled channel, so the size of the **values** array needs to be (number of values) x (number of enabled channels). When one or more of the digital IOs are enabled as inputs, they count as one additional channel. The function informs the user if the voltages of any of the enabled channels have overflowed.

Arguments

`handle`   Returned by HRDLOpenUnit.

`values`   Pointer to an `int32_t` where the sample values are written. If more than one channel is active, the samples are interleaved. If digital channels are enabled then they are always the first value. See table below for the order in which data are returned.

`overflow`   Pointer to an `int16_t` indicating any inputs that have exceeded their maximum voltage range. Channels with overvoltages are indicated by a high bit, with the LSB indicating channel 1 and the MSB channel 16.

`noOfValues`   The number of samples to collect for each active channel

Returns

A non-zero number if successful indicating the number of values returned, or

0 if the call failed or no values available

Ordering of returned data (example)

When two analog channels (e.g. 1 and 5) are enabled and a digital channel is set as an input, the data are returned in the following order.

```
Sample No: 0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 . n-3 n-2 n-1
Channel:   DI 1  5  DI 1  5  DI 1  5  DI 1  5  DI 1  5  . DI  1   5
```

where **n** represents the value returned by the function and **DI** the digital inputs.

The channels are always ordered from channel 1 up to the maximum channel number (ADC-24: channel 16, ADC-20: channel 8). If one or more digital channels are set as inputs then the first sample contains the digital channels.

Digital inputs

The digital channels are represented by a binary bit pattern with 0 representing off, and 1 representing on. Digital input 1 is in bit 0.

5.4.10   HRDLOpenUnit

```
int16_t HRDLOpenUnit
(
    void
)
```

This function opens a [data logger](). The API driver can support up to 20 units.

Arguments

None

Returns

-1   if the unit fails to open
0   if no unit is found
>= 1   handle to the device opened

5.4.11   HRDLOpenUnitAsync

```
int16_t HRDLOpenUnitAsync
(
    void
)
```

Opens a [data logger](#) without blocking the calling thread.

Arguments

None

Returns

0   if there is already an open operation in progress
1   if the open operation has been initiated

5.4.12   HRDLOpenUnitProgress

```
int16_t HRDLOpenUnitProgress
(
    int16_t * handle,
    int16_t * progress
)
```

Checks the progress of an <u>asynchronous</u> open operation.

Arguments

handle   Pointer to an `int16_t` where the unit handle is to be written:
            -1: if the unit fails to open
            0: if no unit is found
            >0: a handle to the device opened (this handle is not valid unless the
            function returns true)

progress   Pointer to an `int16_t` to which the percentage progress is to be written.
            100% implies that the open operation is complete

Returns

0   if open operation is still in progress
1   if the open operation is complete

5.4.13   HRDLReady

```
int16_t HRDLReady
(
    int16_t handle
)
```

This function indicates when the readings are ready to be retrieved from the driver.

Arguments

`handle`   Handle returned by [HRDLOpenUnit](#).

Returns

0 if not ready, or failed
1 if ready

5.4.14   HRDLRun

```
int16_t HRDLRun
(
    int16_t handle,
    int32_t nValues,
    int16_t method
)
```

This function starts the device sampling and storing the samples in the driver's buffer. See Streaming recording methods for help on using this function.

Arguments

handle   Handle returned by HRDLOpenUnit.

nValues   Number of samples to collect for each active channel.

method   Sampling method. This should be one of the values listed below.

Returns

0 if failed,
1 if successful

Sampling methods

Value of method	Description
BM BLOCK (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

5.4.15   HRDLSetAnalogInChannel

```
int16_t HRDLSetAnalogInChannel
(
    int16_t handle,
    int16_t channel,
    int16_t enabled,
    int16_t range,
    int16_t singleEnded
)
```

This function enables or disables the selected analog channel. If you wish to enable an odd-numbered channel in differential mode, you must first make sure that its corresponding even-numbered channel is disabled. (For example, to set channel 1 to differential mode, first ensure that channel 2 is disabled).

Arguments

`handle`   Handle returned by <u>HRDLOpenUnit</u>.

`channel`   The channel that will be enabled or disabled.

                 ADC-20: 1 to 8
                 ADC-24: 1 to 16

`enabled`   Sets the channel active or dormant.

                 0: dormant
                 <> 0: active

`range`   The voltage range to be used during sampling. Applies only to selected channel. See Voltage ranges below.

`singleEnded`   Non-zero to measure a single-ended voltage.
                 Zero for a differential voltage.

Returns

0 if failed
1 if successful

If the function fails, call <u>HRDLGetUnitInfo</u> with `info = HRDL_SETTINGS_ERROR (8)` to obtain the specific settings error.

Voltage ranges

Value of `range`	Voltage range	Availability
`HRDL 2500 MV (0)`	±2500 mV	ADC-20 and ADC-24
`HRDL 1250 MV (1)`	±1250 mV	ADC-20 and ADC-24
`HRDL 625 MV (2)`	±625 mV	ADC-24 only
`HRDL 313 MV (3)`	±312.5 mV	ADC-24 only
`HRDL 156 MV (4)`	±156.25 mV	ADC-24 only
`HRDL 78 MV (5)`	±78.125 mV	ADC-24 only
`HRDL 39 MV (6)`	±39.0625 mV	ADC-24 only

5.4.16   HRDLSetDigitalIOChannel (ADC-24 only)

```
int16_t HRDLSetDigitalIOChannel
(
    int16_t handle,
    int16_t directionOut,
    int16_t digitalOutPinState,
    int16_t enabledDigitalIn
)
```

Sets up the digital input/output channels. If the direction is "output" then the pin can be set high (on) or low (off). While the device is sampling, the direction cannot be changed but the value of an output can.

Arguments

handle   Handle returned by HRDLOpenUnit.

directionOut   The directions of the digital IO pins, either input or output. The four least significant bits must be a combination of HRDL_DIGITAL_IO_CHANNEL constants (see below).

digitalOutPinState   If the pin is set as an output, it can be set high or low by a combination of HRDL_DIGITAL_IO_CHANNEL constants (see below).

enabledDigitalIn   Sets the digital input as active. Use a combination of HRDL_DIGITAL_IO_CHANNEL constants (see below).

Returns

0 if failed,
1 if successful

If the function fails, call HRDLGetUnitInfo with info = HRDL_SETTINGS_ERROR (8) to obtain the specific setting error.

Pin values for directionOut, digitalOutPinState and enabledDigitalIn

Pin value	Description
HRDL DIGITAL IO CHANNEL 1 (1)	IO Pin 1
HRDL DIGITAL IO CHANNEL 2 (2)	IO Pin 2
HRDL DIGITAL IO CHANNEL 3 (4)	IO Pin 3
HRDL DIGITAL IO CHANNEL 4 (8)	IO Pin 4

Examples:

- To set digital channels 1 and 2 to input and digital channels 3 and 4 to output:
  `directionOut = `
  `HRDL_DIGITAL_IO_CHANNEL_4 (8) + HRDL_DIGITAL_IO_CHANNEL_3 (4) = `
  `12`
- To set digital channel 4 high and digital channel 3 low:
  `digitalOutPinState = HRDL_DIGITAL_IO_CHANNEL_4 (8) = 8`
- To set only digital channel 3 high:
  `digitalOutPinState = HRDL_DIGITAL_IO_CHANNEL_3 (4) = 4`
- To turn both digital channels 3 and 4 on:
  `digitalOutPinState = `
  `HRDL_DIGITAL_IO_CHANNEL_4 (8) + HRDL_DIGITAL_IO_CHANNEL_3 (4) = `
  `12`

Example bit patterns for `directionOut` parameter:

Decimal	Bit Pattern	Digital Channel 4	Digital Channel 3	Digital Channel 2	Digital Channel 1
1	0001	Input	Input	Input	Output
10	1010	Output	Input	Output	Input
12	1100	Output	Output	Input	Input
13	1101	Output	Output	Input	Output

The above is a selection of the 16 different options available for the `directionOut` parameter. When a digital channel has been selected as an output, it can then be set on or off with the `digitalOutputPinState` parameter, again using binary bit patterns to represent the different digital channels.

The default setting for the digital channels is "output, off".

5.4.17   HRDLSetInterval

```
int16_t HRDLSetInterval
(
    int16_t handle,
    int32_t sampleInterval_ms,
    int16_t conversionTime
)
```

This sets the sampling time interval. The number of channels active must be able to convert within the specified interval.

Arguments

handle   Handle returned by [HRDLOpenUnit](#).

sampleInterval_ms   Time interval in milliseconds within which all conversions must take place before the next set of conversions starts.

conversionTime   The amount of time given to one channel's conversion. This must be one of the constants below.

Returns

0 if failed
1 if successful

If the function fails, call [HRDLGetUnitInfo](#) with `info = HRDL_SETTINGS_ERRORS` for the specific settings error.

Conversion times

Value of conversionTime	Conversion time
HRDL 60MS (0)	60 ms
HRDL 100MS (1)	100 ms
HRDL 180MS (2)	180 ms
HRDL 340MS (3)	340 ms
HRDL 660MS (4)	660 ms

5.4.18   HRDLSetMains

```
int16_t HRDLSetMains
(
    int16_t handle,
    int16_t sixtyHertz
)
```

This function configures the mains noise rejection setting. Rejection takes effect the next time sampling occurs.

Arguments

handle   Handle returned by [HRDLOpenUnit](#).

sixtyHertz   Specifies whether 50 Hz or 60 Hz noise rejection is applied.

                    0: reject 50 Hz
                    <> 0: reject 60 Hz

Returns

0 if failed
1 if successful

5.4.19   HRDLStop

```
void HRDLStop
(
    int16_t handle
)
```

This function stops the device from sampling data.

When running the device in windowed or streaming mode, you will need to call this function to end data collection. This is particularly important in streaming mode, to ensure that the scope is ready for the next capture.

When running the device in block mode, you can call this function to interrupt data capture.

Arguments

handle   Handle returned by HRDLOpenUnit.

5.5   Programming examples

Your Pico Technology SDK installation includes programming examples in various languages and development environments. Please refer to the SDK for details.

adc20.en r6

5.6 Sequence of calls and data flow

The C sample program, `hrdlcon.c`, demonstrates the use of all the functions of the API driver, and includes examples showing each mode of operation.

5.6.1 Streaming recording methods

5.6.1.1 Collecting a block of data

This method collects a single block of data and then stops.

- Open the data logger with one of the HRDLOpenUnit calls
- Set mains noise rejection with HRDLSetMains
- Set the analog or/and digital channels
- Set the sample interval with HRDLSetInterval
- Start the unit collecting samples by calling HRDLRun with `method = BM_BLOCK`
- Loop
- Repeat Loop until ready (HRDLReady)
- Collect data with HRDLGetValues
- Repeat from "Start the unit" until you have finished collecting data.
- Close the connection to the unit with HRDLCloseUnit.
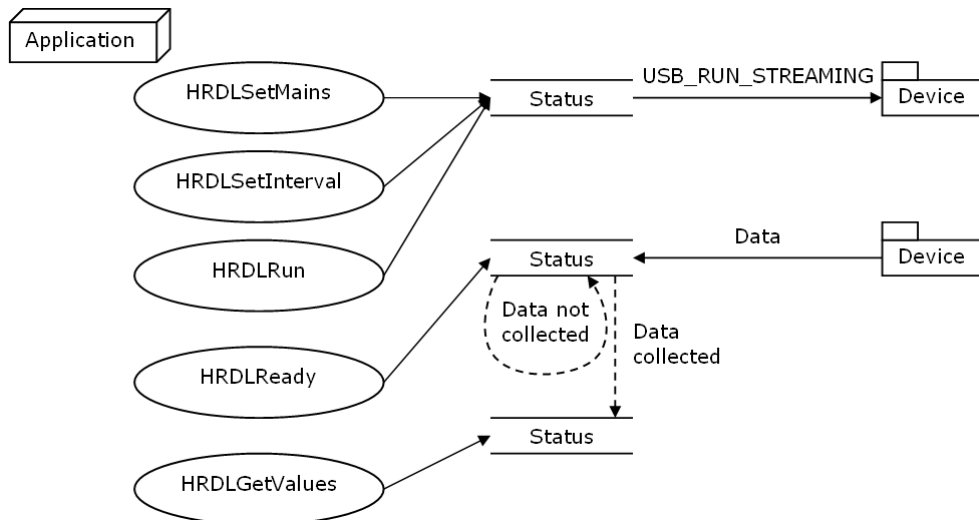
5.6.1.2   Collecting windowed or streaming data

This method causes the device to start sampling. Samples are stored in the driver's buffer. In windowed mode, the buffer will always contain the requested number of samples, but generally only a subset of these are new data. In streaming mode, new data are returned continuously.

- Open the data logger with one of the HRDLOpenUnit calls
- Set mains noise rejection with HRDLSetMains
- Set the analog or/and digital channels
- Set the sample interval with HRDLSetInterval
- Start the unit collecting samples by calling HRDLRun with `method = BM_WINDOW` or `BM_STREAM`
- Loop
- Repeat Loop until ready (HRDLReady)
- Collect data whenever you want with HRDLGetValues
- Call HRDLStop to end data collection
- Close the connection to the unit with HRDLCloseUnit
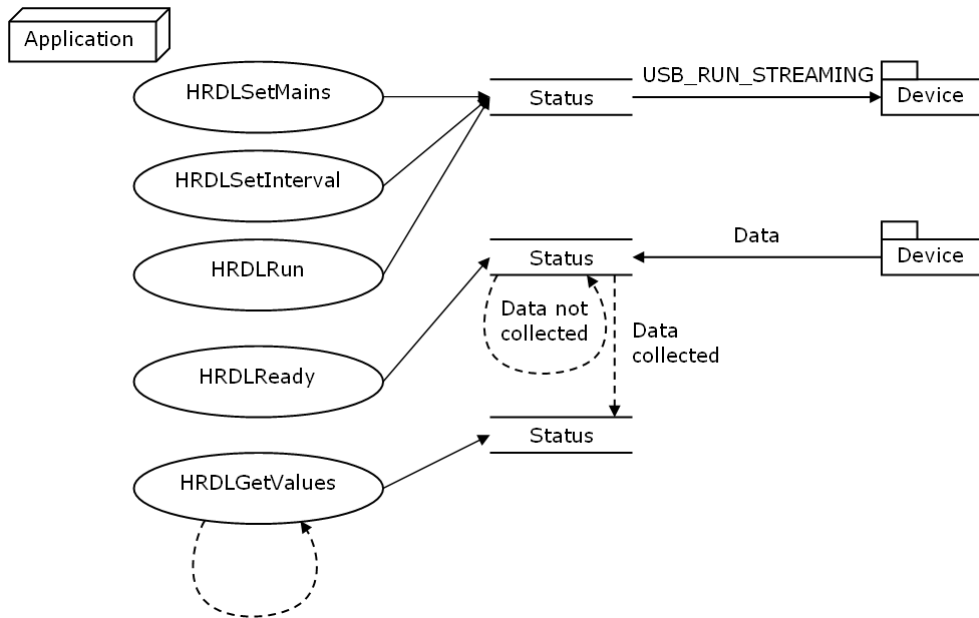
5.6.2 Single-value recording methods

5.6.2.1 Collecting a single reading, blocking

This method collects a single reading and blocks the calling thread.

- Open the data logger with one of the HRDLOpenUnit calls
- Set mains noise rejection with HRDLSetMains
- Get a single reading (one channel only at a time) with HRDLGetSingleValue
- Close the connection to the unit with HRDLCloseUnit
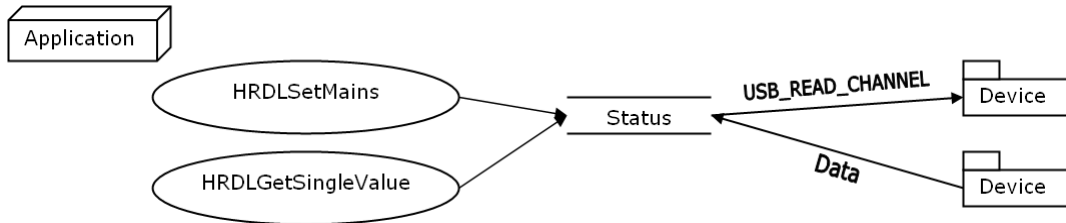
5.6.2.2 Collecting a single reading, non-blocking

This method collects a single reading without blocking the calling thread.

- Open the data logger with one of the HRDLOpenUnit calls
- Set mains noise rejection with HRDLSetMains
- Start the conversion for a single reading with HRDLCollectSingleValueAsync
- Wait until the reading is ready (HRDLReady)
- Get the reading from the driver with HRDLGetSingleValueAsync
- Close the connection to the unit with HRDLCloseUnit
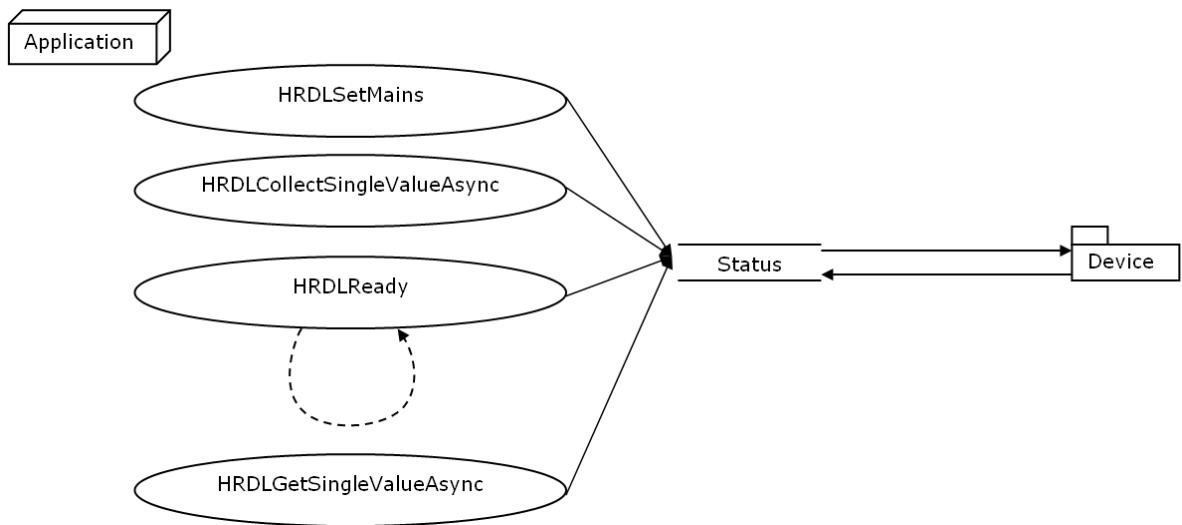
6   Glossary

Asynchronous. In asynchronous data collection, your application requests data from the driver, and the driver immediately returns without blocking the application. The application must then poll a status function until the data is ready.

Common-mode rejection ratio. The ratio by which the data logger attenuates a common-mode voltage (see below). It is defined as:

CMRR(dB) = 20 log10 (Vin/Vmeas),

where CMRR(dB) is the common-mode rejection ratio in decibels, Vin is the common-mode voltage present at the input, and Vmeas is the common-mode voltage visible in the measured data.

Common mode voltage. A differential signal fed into the data logger consists of a positive input (Vp) and a negative input (Vn), and the logger measures the difference (Vdiff = Vp - Vn) between the two inputs. This means that any offset in ground potential between the signal source and the data logger adds a constant voltage, called the common mode voltage (Vcm), to both inputs equally, so ideally it does not affect Vdiff. In practice, however, the data logger cannot make an accurate measurement if Vcm is too large, and even small values of Vcm may affect the reading slightly.

Data logger. A measuring instrument that monitors one or more analog signals, samples them at pre-programmed intervals, then accurately converts the samples to digital data and stores them in memory. The ADC-20 and ADC-24 use your PC for storage and display.

DLL. Dynamic Link Library. A DLL is a file containing a collection of Windows functions designed to perform a specific class of operations.

Driver. A driver is a computer program that acts as an interface, generally between a hardware component and a computer system, the hardware in this case being the data logger.

EMC. Electromagnetic compatibility. The ability of a device to operate in proximity with other devices without causing or suffering undue interference from electromagnetic fields or conducted electrical noise.

Gain error. Gain error is the worst deviation of a measurement from the true value, measured over the whole input range and expressed as a percentage.

Galvanic isolation. A barrier between two parts of an electrical circuit that prevents noise and voltage offsets in one part from affecting the other part.

Input impedance. This is the impedance of the input channel of the data logger. Impedance is the ratio of the voltage across the input to the current flowing through it, and at low frequencies can be considered as a pure resistance. The larger the impedance, the more accurate the measurement.

Input voltage range. The input voltage range is the range of voltages that an analog channel can convert without an overload error. The maximum input voltage range of the ADC-20 and ADC-24 is therefore -2.5 V to +2.5 V. Furthermore, you should not inject voltages outside the range -5 V to +5 V, as this can cause measurement errors on all channels. You will not damage the unit unless you exceed the overvoltage protection voltage range.

LSB. Least significant bit. In a binary word, the least significant bit has the value 1.

MSB. Most significant bit. In an n-bit binary word, the most significant bit has the value $2^{(n-1)}$.

Noise-free resolution. Any measurement is subject to noise. In a digital measuring instrument, a result with a resolution of n bits may include m bits of noise. The noise-free resolution is then $n-m$ bits.

Noise rejection. The ability of the data logger to attenuate noise in a given frequency range. The ADC-20/ADC-24 can be programmed to reject noise at either 50 hertz or 60 hertz. The noise rejection ratio is defined as:

NRR(dB) = 20 log10 (Vin/Vmeas)

where NRR(dB) is the noise rejection ratio in decibels, Vin is the noise voltage at the input, and Vmeas is the noise voltage that appears in the measurement.

Overvoltage protection. Overvoltage protection is characterized by the maximum voltage that can be applied across the inputs of the data logger without causing damage to it. The ADC-20 and ADC-24 are protected to ±30 V.

Resolution. A value, in bits, indicating the number of unique digital values that the converter can produce. If the resolution is n bits, then the number of unique values is 2 to the power n.

RH. Relative Humidity. RH is the amount of water vapor present in the air, expressed as a percentage of the amount needed for saturation at the same temperature.

USB. Universal Serial Bus. This is a standard port that enables you to connect external devices to PCs. A full-speed USB 2.0 port operates at up to 480 megabits per second. The PicoLog 1000 Series is also compatible with any USB port from USB 1.1 upwards.

7 Appendix A: Connecting the data logger in Windows XP

You must install the software before plugging in the data logger. When you have installed the software, connect the data logger's USB cable to a spare USB port on your computer.

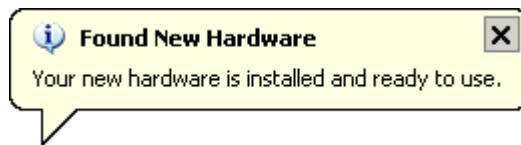When you first plug in the device, your computer should show this message:

and then display the New Hardware Found Wizard.
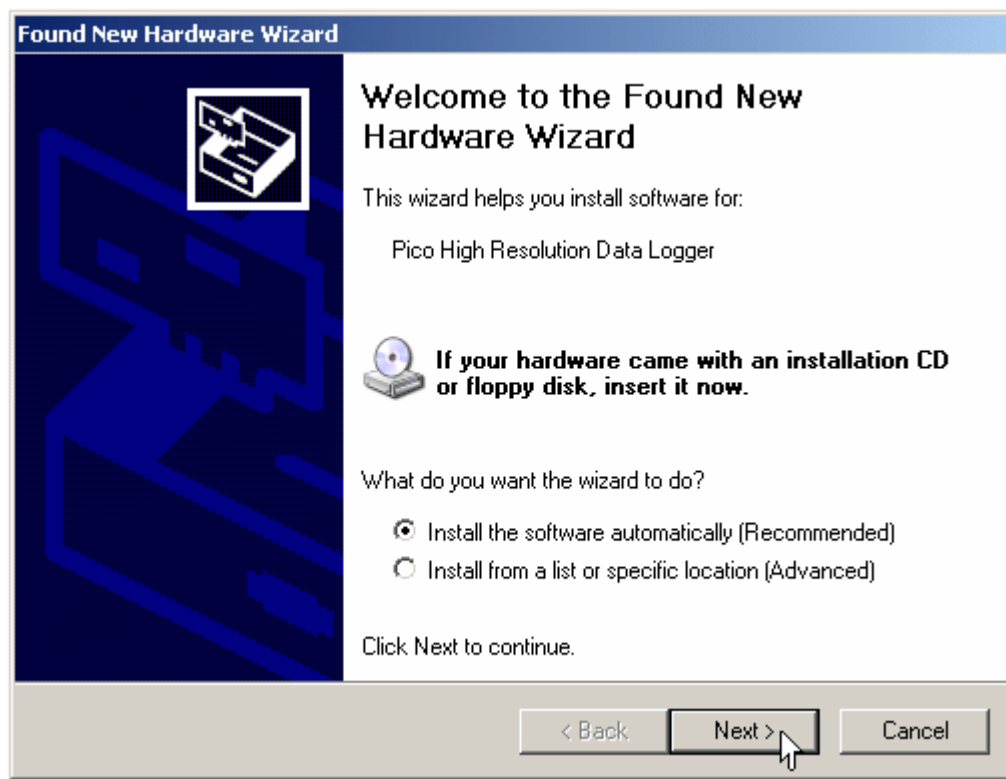
- In the "Welcome to the New Hardware Found Wizard" dialog (above), click Next>.
- Wait while the wizard installs the software.
- A dialog will appear like the one below.
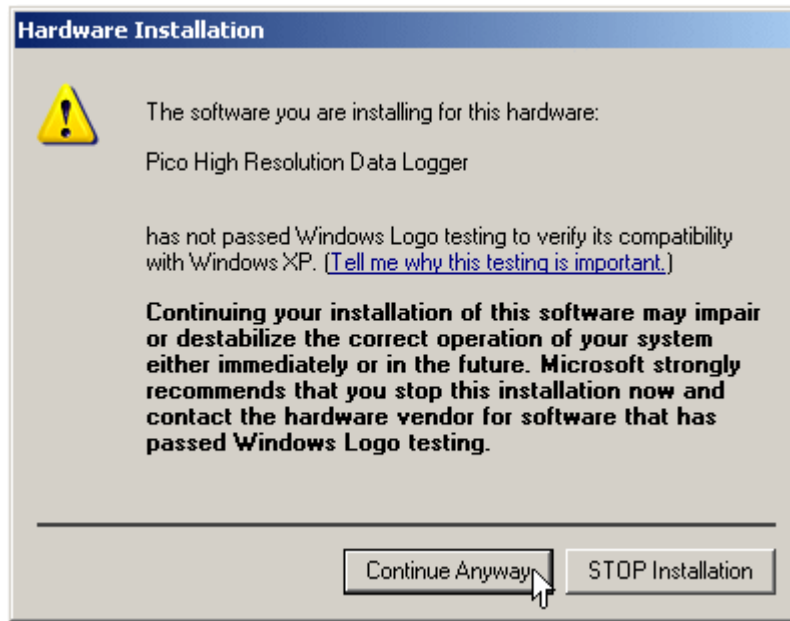
- Click Continue Anyway.
- Continue to wait while the wizard installs the software.
- When instructed, click Finish to close the wizard.
- After a few seconds you should see this message:

- Your ADC-20 or ADC-24 Data Logger is now properly installed.

Index

UK headquarters

Pico Technology
James House
Colmworth Business Park
St. Neots
Cambridgeshire
PE19 8YP
United Kingdom

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

sales@picotech.com
support@picotech.com

www.picotech.com

USA headquarters

Pico Technology
320 N Glenwood Blvd
Tyler
Texas 75702
United States of America

Tel: +1 800 591 2796
Fax: +1 620 272 0981

Elso Philips Service
Jilemnického 2; 911 01 Trenčín
tel: +421 32 6582410, 7431690
fax: +421 32 6582592
email: elso@elso.sk
web: www.elso.sk

adc20.en r6 2016-05-31