Humusoft Data Acquisition Library Reference Manual

Generated by Doxygen 1.4.6-NO

Wed May 3 16:25:28 2006

Contents

Chapter 1

Humusoft Data Acquisition Library

1.1 Introduction

Humusoft Data Acquisition Library is a library of functions that allows the user to access Humusoft data acquisition boards from the Win32 API. It is not designed with any specific programming language in mind, but it is expected that C or C++ will be the most commonly used ones.

1.2 Installation

The Humusoft Data Acquisition Library binary file is named `hudaqlib.dll` and is installed together with the driver into the operating system directory. No additional installation steps need to be performed besides installing the driver.

1.3 Basic concepts of working with the library

When using the Humusoft Data Acquisition Library, each data acquisition device is represented by its handle. So the first necessary action before accessing a device is to open a handle for it. Then, the device has several subsystems, like Analog Input or Digital Output. Each subsystem usually provides several channels of the particular type - so a device has for example eight analog inputs, four counter inputs and one digital input. The channels are accessed by their numbers and are numbered starting from zero - that is, if a device has eight channels, the last channel has number 7.

1.4 Building applications with the library

The application that uses the library needs to be dynamically linked against the `hudaqlib.dll` which contains all the functions described later in this documentation. Programs in C or C++ should include the file **hudaqlib.h**(p. **??**) that contain prototypes for the the functions. Users of Microsoft compilers can then link against the file `hudaqlib.lib`, which is the import library for `hudaqlib.dll`. Users of other programming languages and compilers will need to dynamically link against the necessary functions and pass parameters according to the documentation for the individual functions. The individual functions are documented in the **Modules** section. All the

functions in the library use the `__stdcall` calling convention, similar to functions available in the Win32 API used for generic Microsoft Windows programming.

Before starting a new project, it is usually best to copy the files **hudaqlib.h**(p. **??**) and `hudaqlib.dll` into the project directory. Then you can either create a project in your favorite development environment or you can use a command-line compiler to build the project.

To build any of the examples, please copy the example files into the directory where you have copied the files **hudaqlib.h**(p. **??**) and hudaqlib.dll. Then, you can either create a project in your favorite development environment or you can use a command-line compiler to build the example. For example, building the AIRead example with Microsoft Visual C using the command-line compiler would be done using this command:

```
cl AIRead.c hudaqlib.lib
```

Author:

Chapter 2

Humusoft Data Acquisition Library Module Index

2.1 Humusoft Data Acquisition Library Modules

Here is a list of all modules:

Chapter 3

Humusoft Data Acquisition Library Module Documentation

3.1 Board Initialization

3.1.1 Detailed Description

The board initialization and cleanup routines are intended for establishing communication with a device and for closing the communication handle after the communication is finished.

Each successful opening of the device must be followed by closing the device before the application exits. It is not recommended to open a handle to the same device multiple times in the same application.

Functions

- **HUDAQHANDLE HudaqOpenDevice** (const char *devicename, int deviceorder, int options)

  Open a data acquisition device.

- **HUDAQSTATUS HudaqResetDevice** (**HUDAQHANDLE** handle)

  Reset a data acquisition device.

- void **HudaqCloseDevice** (**HUDAQHANDLE** handle)

  Close a data acquisition device handle.

3.1.2 Function Documentation

3.1.2.1 void HudaqCloseDevice (HUDAQHANDLE *handle*)

Close a data acquisition device handle.

The device handle becomes invalid after this call and must not be used any more. The device state is not changed when its handle is closed. If it is required that the device is set to a specific state before exitting the application, it must be done explicitly before calling this function.

Parameters:
← ***handle*** Device handle.

Examples:
AIRead.c, **AIReadMultiple.c**, **AOWrite.c**, **AOWriteMultiple.c**, **CtrRead.c**, **DIRead.c**, **DIReadBit.c**, **DOWrite.c**, **DOWriteBit.c**, **EncCtrRead.c**, **EncRead.c**, and **PWMWrite.c**.

3.1.2.2 HUDAQHANDLE HudaqOpenDevice (const char * *devicename*, int *deviceorder*, int *options*)

Open a data acquisition device.

Parameters:
← ***devicename*** Device name. This parameter is the device type as a string, for example "MF624".
← ***deviceorder*** Device order. One-based index to distinguish between devices of identical type.
← ***options*** Reserved, must be zero.

Returns:
Device handle or zero on failure.

Examples:
AIRead.c, **AIReadMultiple.c**, **AOWrite.c**, **AOWriteMultiple.c**, **CtrRead.c**, **DIRead.c**, **DIReadBit.c**, **DOWrite.c**, **DOWriteBit.c**, **EncCtrRead.c**, **EncRead.c**, and **PWMWrite.c**.

3.1.2.3 HUDAQSTATUS HudaqResetDevice (HUDAQHANDLE *handle*)

Reset a data acquisition device.

This function puts the device into initial state.

Parameters:
← ***handle*** Device handle.

Returns:
HUDAQSUCCESS(p. 23) on success, other values on failure.

3.2 Analog Input

3.2.1 Detailed Description

The Analog Input routines read signal values from the analog inputs of the data acquisition device.

The signal value read is returned in volts. It is possible to read a single input channel or multiple channels by a single function. Using a single function to read multiple channels at once is faster than multiple calls reading one channel each.

Device capabilities:

- AD622 has 8 analog input channels.
- MF624 has 8 analog input channels.
- AD612 has 8 analog input channels.
- MF614 has 8 analog input channels.

Functions

- double **HudaqAIRead** (**HUDAQHANDLE** handle, unsigned channel)

  Read data from a single analog input channel.

- **HUDAQSTATUS HudaqAIReadMultiple** (**HUDAQHANDLE** handle, unsigned number, const unsigned ∗channels, double ∗values)

  Read data from multiple analog input channels.

3.2.2 Function Documentation

3.2.2.1 double HudaqAIRead (HUDAQHANDLE *handle*, unsigned *channel*)

Read data from a single analog input channel.

Parameters:

← ***handle*** Device handle.

← ***channel*** Analog input channel number.

Returns:

Value read from the analog input channel.

Examples:

AIRead.c.

3.2.2.2 HUDAQSTATUS HudaqAIReadMultiple (HUDAQHANDLE *handle*, unsigned *number*, const unsigned ∗ *channels*, double ∗ *values*)

Read data from multiple analog input channels.

The analog input channels are read with the minimum possible interval between individual channels or simultaneously if the device supports it.

Parameters:

← ***handle*** Device handle.

← ***number*** Number of channels to read.

← ***channels*** Array of numbers of channels that will be read. The array must contain the specified number of channel numbers.

→ ***values*** Pointer to the array to be filled with values read from the analog input channels. The array is allocated by the caller and must contain enough space to hold all the values read.

Returns:

HUDAQSUCCESS(p. 23) on success, other values on failure.

Examples:

AIReadMultiple.c.

3.3 Analog Output

3.3.1 Detailed Description

The Analog Output routines write signal values to the analog outputs of the data acquisition device.

The signal value to be written is specified in volts. It is possible to write a single output channel or multiple channels by a single function. Using a single function to write multiple channels at once is faster than multiple calls writing one channel each.

Device capabilities:

- AD622 has 8 analog output channels.
- MF624 has 8 analog output channels.
- AD612 has 4 analog output channels.
- MF614 has 4 analog output channels.

Functions

- void **HudaqAOWrite** (**HUDAQHANDLE** handle, unsigned channel, double value)

  Write data to a single analog output channel.

- **HUDAQSTATUS HudaqAOWriteMultiple** (**HUDAQHANDLE** handle, unsigned number, const unsigned ∗channels, const double ∗values)

  Write data to multiple analog output channels.

3.3.2 Function Documentation

3.3.2.1 void HudaqAOWrite (HUDAQHANDLE *handle*, unsigned *channel*, double *value*)

Write data to a single analog output channel.

Parameters:

← ***handle*** Device handle.

← ***channel*** Analog output channel number.

← ***value*** Value to write to the analog output channel.

Examples:

AOWrite.c.

3.3.2.2 HUDAQSTATUS HudaqAOWriteMultiple (HUDAQHANDLE *handle*, unsigned *number*, const unsigned ∗ *channels*, const double ∗ *values*)

Write data to multiple analog output channels.

The analog output channels are updated with the minimum possible interval between individual channels or simultaneously if the device supports it.

Parameters:

← ***handle*** Device handle.

← ***number*** Number of channels to be written.

← ***channels*** Array of numbers of channels that will be written. The array must contain the specified number of channel numbers.

← ***values*** Array of values to be written to the analog output channels. The array must contain the specified number of values.

Returns:

HUDAQSUCCESS(p. 23) on success, other values on failure.

Examples:

AOWriteMultiple.c.

3.4 Digital Input

3.4.1 Detailed Description

Digital input routines read logical values from digital inputs.

The values can be read as individual bits, as the whole channel, or from multiple channels at once.

Device capabilities:

- AD622 has one 8-bit digital input channel.
- MF624 has one 8-bit digital input channel.
- AD612 has one 8-bit digital input channel.
- MF614 has one 8-bit digital input channel.

Functions

- int **HudaqDIReadBit** (**HUDAQHANDLE** handle, unsigned channel, unsigned bit)
  Read a single bit from a digital input channel.
- int **HudaqDIRead** (**HUDAQHANDLE** handle, unsigned channel)
  Read data from a single digital input channel.
- **HUDAQSTATUS HudaqDIReadMultiple** (**HUDAQHANDLE** handle, unsigned number, const unsigned ∗channels, unsigned ∗values)
  Read data from multiple digital input channels.

3.4.2 Function Documentation

3.4.2.1 int HudaqDIRead (HUDAQHANDLE *handle*, unsigned *channel*)

Read data from a single digital input channel.

The number of bits in a digital input channel is device specific.

Parameters:

← ***handle*** Device handle.

← ***channel*** Digital input channel number.

Returns:

Value read from the digital input channel.

Examples:

DIRead.c.

3.4.2.2 int HudaqDIReadBit (HUDAQHANDLE *handle*, unsigned *channel*, unsigned *bit*)

Read a single bit from a digital input channel.

Parameters:

← ***handle*** Device handle.

← ***channel*** Digital input channel number.

← ***bit*** Bit position in the channel.

Returns:

Bit value read from the specified bit.

Examples:

DIReadBit.c.

3.4.2.3 HUDAQSTATUS HudaqDIReadMultiple (HUDAQHANDLE *handle*, unsigned *number*, const unsigned * *channels*, unsigned * *values*)

Read data from multiple digital input channels.

Parameters:

← ***handle*** Device handle.

← ***number*** Number of channels to read.

← ***channels*** Array of numbers of channels that will be read. The array must contain the specified number of channel numbers.

→ ***values*** Pointer to the array to be filled with values read from the digital input channels. The array is allocated by the caller and must contain enough space to hold all the values read.

Returns:

HUDAQSUCCESS(p. 23) on success, other values on failure.

3.5 Digital Output

3.5.1 Detailed Description

Digital output routines write logical values to digital outputs.

The values can be written as individual bits, as multiple bits in one channel, as the whole channel, or to multiple channels at once.

Device capabilities:

- AD622 has one 8-bit digital output channel.
- MF624 has one 8-bit digital output channel.
- AD612 has one 8-bit digital output channel.
- MF614 has one 8-bit digital output channel.

Functions

- void **HudaqDOWriteBit** (**HUDAQHANDLE** handle, unsigned channel, unsigned bit, unsigned value)

  Write data to a single bit of a digital output channel.

- void **HudaqDOWrite** (**HUDAQHANDLE** handle, unsigned channel, unsigned value)

  Write data to a single digital output channel.

- void **HudaqDOWriteMultipleBits** (**HUDAQHANDLE** handle, unsigned channel, unsigned mask, unsigned value)

  Modify multiple bits in a single digital output channel.

- **HUDAQSTATUS HudaqDOWriteMultiple** (**HUDAQHANDLE** handle, unsigned number, const unsigned ∗channels, const unsigned ∗values)

  Write data to multiple digital input channels.

3.5.2 Function Documentation

3.5.2.1 void HudaqDOWrite (HUDAQHANDLE *handle*, unsigned *channel*, unsigned *value*)

Write data to a single digital output channel.

The number of bits in a digital output channel is device specific.

Parameters:

← ***handle*** Device handle.

← ***channel*** Digital output channel number.

← ***value*** Value to be written into digital output.

← ***channel*** digital input channel

Examples:

DOWrite.c.

3.5.2.2 void HudaqDOWriteBit (HUDAQHANDLE *handle*, unsigned *channel*, unsigned *bit*, unsigned *value*)

Write data to a single bit of a digital output channel.

Parameters:

← ***handle*** Device handle.

← ***channel*** Digital output channel number.

← ***bit*** Bit position in channel specified

← ***value*** Bit value to be written to the specified bit.

Examples:

DOWriteBit.c.

3.5.2.3 HUDAQSTATUS HudaqDOWriteMultiple (HUDAQHANDLE *handle*, unsigned *number*, const unsigned * *channels*, const unsigned * *values*)

Write data to multiple digital input channels.

Parameters:

← ***handle*** Device handle.

← ***number*** Number of channels to be written.

← ***channels*** Array of numbers of channels that will be written. The array must contain the specified number of channel numbers.

← ***values*** Array of values to be written to the digital output channels. The array must contain the specified number of values.

Returns:

HUDAQSUCCESS(p. 23) on success, other values on failure.

3.5.2.4 void HudaqDOWriteMultipleBits (HUDAQHANDLE *handle*, unsigned *channel*, unsigned *mask*, unsigned *value*)

Modify multiple bits in a single digital output channel.

All the bits are modified simultaneously.

Parameters:

← ***handle*** Device handle.

← ***channel*** Digital output channel number.

← ***mask*** Mask that specifies bits that will be modified. Bits that are 1 will be assigned the corresponding bits of `value`, bits that are 0 will be left untouched.

← ***value*** Value to write. Only the bits specified by `mask` are modified, the other bits are ignored.

3.6 Counter Input

3.6.1 Detailed Description

Counter input routines read the count of pulse on the counter input.

Because the counter hardware can be shared among multiple subsystems of the device, not all channels may be available when functions from other subsystems are utilized.

Device capabilities:

- AD622 has no counter input channel.
- MF624 has 4 counter input channels, the hardware is shared with PWM output channels.
- AD612 has no counter input channel.
- MF614 has 4 counter input channels, the hardware is shared with PWM output channels.

Enumerations

- enum **HudaqCtrClockSources** {

  HudaqCtrCLOCK50MHz = 0, **HudaqCtrCLOCK10MHz** = 1, **HudaqCtrCLOCK1MHz** = 2, **HudaqCtrCLOCK100kHz** = 3,

  HudaqCtrCLOCKRISEIN = 5, **HudaqCtrCLOCKFALLIN** = 6, **HudaqCtrCLOCKEITERIN** = 7, **HudaqCtrCLOCKRISEN_1** = 9,

  HudaqCtrCLOCKFALLN_1 = 10, **HudaqCtrCLOCKEITHERN_1** = 11, **HudaqCtrCLOCKRISEN1** = 13, **HudaqCtrCLOCKFALLN1** = 14,

  HudaqCtrCLOCKEITHERN1 = 15 }

  Clock sources list for counter.

Functions

- void **HudaqCtrReset** (**HUDAQHANDLE** handle, unsigned channel)

  Reset internal counter value.

- int **HudaqCtrRead** (**HUDAQHANDLE** handle, unsigned channel)

  Read data from a counter channel.

3.6.2 Enumeration Type Documentation

3.6.2.1 enum HudaqCtrClockSources

Clock sources list for counter.

Enumerator:

HudaqCtrCLOCK50MHz Use internal clock 50MHz.

HudaqCtrCLOCK10MHz Use internal clock 10MHz.

HudaqCtrCLOCK1MHz Use internal clock 1MHz.

HudaqCtrCLOCK100kHz Use internal clock 100kHz.

HudaqCtrCLOCKRISEIN Use external input, count on rise edge.

HudaqCtrCLOCKFALLIN Use external input, count on fall edge.

HudaqCtrCLOCKEITERIN Use external input, count on either edge.

HudaqCtrCLOCKRISEN_1 Use output of previous counter, count on rise edge.

HudaqCtrCLOCKFALLN_1 Use output of previous counter, count on fall edge.

HudaqCtrCLOCKEITHERN_1 Use output of previous counter, count on either edge.

HudaqCtrCLOCKRISEN1 Use output of next counter, count on rise edge.

HudaqCtrCLOCKFALLN1 Use output of next counter, count on fall edge.

HudaqCtrCLOCKEITHERN1 Use output of next counter, count on either edge.

3.6.3 Function Documentation

3.6.3.1 int HudaqCtrRead (HUDAQHANDLE *handle*, unsigned *channel*)

Read data from a counter channel.

The value is the number of pulses counted by the counter since reset.

Parameters:

← ***handle*** Device handle.

← ***channel*** Counter input channel number.

Returns:

Value read from the counter input channel.

Examples:

CtrRead.c.

3.6.3.2 void HudaqCtrReset (HUDAQHANDLE *handle*, unsigned *channel*)

Reset internal counter value.

If a selected counter is not in a counting mode it is switched to it.

Parameters:

← ***handle*** Device handle.

← ***channel*** Number of encoder.

Examples:

CtrRead.c.

3.7 Encoder Input

3.7.1 Detailed Description

Encoder input routines read inputs from special encoder counters.

These counters have three inputs A, B and I, which enable to directly connect quadrature encoders with index inputs. A and B inputs are connected to IRC and I input is connected to reset pulse. Resetting counter on index pulse could be optionally turned off, see **HudaqEncResModes**(p. 18).

Encoders could be configured as normal counters with one input A and no output, see **HudaqEncModes**(p. 18).

Device capabilities:

- AD622 has no encoder input channel.
- MF624 has 4 encoder input channels.
- AD612 has no encoder input channel.
- MF614 has 4 encoder input channels.

Enumerations

- enum **HudaqEncModes** { **HudaqEncMODEIRC** = 0, **HudaqEncMODERISING**, **HudaqEncMODEFALLING**, **HudaqEncMODEEITHER** }

  Types for Encoder channel option **HudaqEncMODE**(p. 22) *used by* **HudaqEncRead**(p. 19) *function.*

- enum **HudaqEncBlocks** { **HudaqEncBLOCKENABLE** = 0, **HudaqEncBLOCKDISABLE**, **HudaqEncBLOCKI0**, **HudaqEncBLOCKI1** }

  Types for Encoder channel option **HudaqEncBLOCK**(p. 22) *used by* **HudaqEncRead**(p. 19) *function.*

- enum **HudaqEncResModes** {

  HudaqEncRESENABLE = 0, **HudaqEncRESRESET**, **HudaqEncRESI0**, **HudaqEncRESI1**,

  HudaqEncRESIRISING, **HudaqEncRESIFALLING**, **HudaqEncRESIEITHER** }

  Types for Encoder channel option **HudaqEncRESMODE**(p. 22) *used by* **HudaqEncRead**(p. 19) *function.*

Functions

- void **HudaqEncReset** (**HUDAQHANDLE** handle, unsigned channel)

  Reset internal encoder value.

- int **HudaqEncRead** (**HUDAQHANDLE** handle, unsigned channel)

  Read data from an encoder channel.

3.7.2 Enumeration Type Documentation

3.7.2.1 enum HudaqEncBlocks

Types for Encoder channel option **HudaqEncBLOCK**(p. 22) used by **HudaqEncRead**(p. 19) function.

Blocking of counter allow to conditionally disable or enable counting depending on external signal.

Enumerator:

HudaqEncBLOCKENABLE No blocking used, encoder runs forewer. (MF614 & MF624).

HudaqEncBLOCKDISABLE Encoder is stopped. (MF624).

HudaqEncBLOCKI0 Count only when input I=0 (MF624).

HudaqEncBLOCKI1 Count only when input I=1 (MF624).

3.7.2.2 enum HudaqEncModes

Types for Encoder channel option **HudaqEncMODE**(p. 22) used by **HudaqEncRead**(p. 19) function.

Modes are influencing what to count.

Enumerator:

HudaqEncMODEIRC Decode IRC connected to input A and B (default) (MF614 & MF624).

HudaqEncMODERISING Count up on A rising edge (MF614 & MF624).

HudaqEncMODEFALLING Count up on A falling edge (MF624).

HudaqEncMODEEITHER Count up on A either edge (MF624).

3.7.2.3 enum HudaqEncResModes

Types for Encoder channel option **HudaqEncRESMODE**(p. 22) used by **HudaqEncRead**(p. 19) function.

Resetting of counter allow to conditionally reset encoder depending on external signal.

Enumerator:

HudaqEncRESENABLE Encoder is not reset depending on external signal (MF614 + MF624).

HudaqEncRESRESET Permanent RESET (MF614 + MF624).

HudaqEncRESI0 Reset Encoder when I=0 (MF614 + MF624).

HudaqEncRESI1 Reset Encoder when I=1 (MF614 + MF624).

HudaqEncRESIRISING Reset Encoder on rising edge I (MF624).

HudaqEncRESIFALLING Reset Encoder on falling edge I (MF624).

HudaqEncRESIEITHER Reset Encoder on either edges I (MF624).

3.7.3 Function Documentation

3.7.3.1 int HudaqEncRead (HUDAQHANDLE *handle*, unsigned *channel*)

Read data from an encoder channel.

Parameters:

← ***handle*** Device handle.

← ***channel*** Encoder input channel number.

Returns:

Value read from the encoder input channel.

Examples:

EncCtrRead.c, and **EncRead.c**.

3.7.3.2 void HudaqEncReset (HUDAQHANDLE *handle*, unsigned *channel*)

Reset internal encoder value.

Parameters:

← ***handle*** Device handle.

← ***channel*** Number of encoder.

3.8 PWM Output

3.8.1 Detailed Description

PWM output routines set the counter output pins to generate pulse-width modulation signal with given frequency and duty.

Because the counter hardware can be shared among multiple subsystems of the device, not all channels may be available when functions from other subsystems are utilized.

Device capabilities:

- AD622 has no PWM output channel.
- MF624 has 4 PWM output channels, the hardware is shared with counter input channels. Maximal output frequency is limited to 25MHz.
- AD612 has no PWM output channel.
- MF614 has 4 PWM output channels, the hardware is shared with counter input channels. Maximal output frequency is limited to 10MHz.

Functions

- **HUDAQSTATUS HudaqPWMWrite** (**HUDAQHANDLE** handle, unsigned channel, double frequency, double duty)

  Generate pulse-width modulation signal on a counter output.

3.8.2 Function Documentation

3.8.2.1 HUDAQSTATUS HudaqPWMWrite (HUDAQHANDLE *handle*, unsigned *channel*, double *frequency*, double *duty*)

Generate pulse-width modulation signal on a counter output.

Parameters:

← ***handle*** Device handle.

← ***channel*** Counter output channel number.

← ***frequency*** Output frequency in Hz.

← ***duty*** Output signal duty. The duty is the fraction of signal period during which the signal is at high level.

Value 0 means permanent logical low on the counter output.

Value 0.5 means periodic signal with the counter output for half of the period at logical low and for half of the period at logical high.

Value 1 means permanent logical high on the counter output.

Returns:

HUDAQSUCCESS(p. 23) on success, other values on failure.

Examples:

PWMWrite.c.

3.9 Configuration of channels

3.9.1 Detailed Description

Some channels have configurable parameters - these can be configured by a function designed for this purpose.

The channel parameter setting then persists until changed or until the device is reset.

All parameter identifiers for one channel have same prefix.

Please note, that a particular device do not necesarily support all parameters and their values. For example AD612 do not support encoders and counters. Thus all identifiers prefixed by HudaqEnc..., HudaqPWM... or HudaqCtr... do not succeed. Other options could be device specific.

Enumerations

- enum **HudaqParameter** {

  HudaqAIUNITS = HudaqAI, **HudaqAOUNITS** = HudaqAO, **HudaqEncRESETONREAD** = HudaqEnc, **HudaqEncFILTER**,

  HudaqEncMODE, **HudaqEncBLOCK**, **HudaqEncRESMODE**, **HudaqEncI**,

  HudaqCtrRESETONREAD = HudaqCtr, **HudaqCtrCLOCKSOURCE** }

  Constants for confguration of all Hudaq channels.

Functions

- **HUDAQSTATUS HudaqSetParameter** (**HUDAQHANDLE** handle, unsigned channel, **HudaqParameter** param, double value)

  Set parameters for given subsystem.

- double **HudaqGetParameter** (**HUDAQHANDLE** handle, unsigned channel, **HudaqParameter** param)

  Read one separate parameter from a given subsystem.

3.9.2 Enumeration Type Documentation

3.9.2.1 enum HudaqParameter

Constants for confguration of all Hudaq channels.

These constants could be used inside **HudaqGetParameter**(p. 22) and **HudaqSetParameter**(p. 22) functions. Please note, that a particular devices do not neccesarily support all functions.

Enumerator:

HudaqAIUNITS set units of channel, internal type 'int'. See **HudaqAIOUnits**(p. 23).

HudaqAOUNITS Set units of channel, internal type 'int'. See **HudaqAIOUnits**(p. 23).

.

HudaqEncRESETONREAD Reset encoder value after every read; option type 'int', values 0 (off), 1 (on).

HudaqEncFILTER Filter IRC inputs; option type 'int', values 0 (off), 1 (on).

HudaqEncMODE Encoder mode see **HudaqEncModes**(p. 18); internal option type 'int'.

HudaqEncBLOCK Conditional block counting; ; see **HudaqEncBlocks**(p. 18); internal type 'int'.

HudaqEncRESMODE Auto reset mode depending on external signal; see **HudaqEncResModes**(p. 18); internal option type 'int'.

HudaqEncI Read I signal. Only **HudaqGetParameter**(p. 22) is supported; internal option type int.

.

HudaqCtrRESETONREAD Reset counter value after every read; internal option type 'int', values 0 (off), 1 (on).

HudaqCtrCLOCKSOURCE See enum **HudaqCtrClockSources**(p. 15) for available values.

3.9.3 Function Documentation

3.9.3.1 double HudaqGetParameter (HUDAQHANDLE *handle*, unsigned *channel*, HudaqParameter *param*)

Read one separate parameter from a given subsystem.

Please note, that a particular device do not neccesarily support all parameters and their values.

Parameters:

← ***handle*** Device handle.

← ***channel*** number of counter

← ***param*** Identifier of parameter, use **HudaqParameter**(p. 21) for their list.

Returns:

parameter specified promoted to double.

3.9.3.2 HUDAQSTATUS HudaqSetParameter (HUDAQHANDLE *handle*, unsigned *channel*, HudaqParameter *param*, double *value*)

Set parameters for given subsystem.

Parameters:

← ***handle*** Device handle.

← ***channel*** number of counter

← ***param*** Identifier of parameter, use **HudaqParameter**(p. 21) for their list.

← ***value*** Variable that contains parameter specified.

Returns:

HUDAQSUCCESS(p. 23) on success, other values on failure.

Examples:

EncCtrRead.c.

3.10 General purpose Constants and Data Types

3.10.1 Detailed Description

This section documents constants and data types used throughout the Humusoft Data Acquisition Library.

Typedefs

- typedef size_t **HUDAQHANDLE**

  The HUDAQ device handle data type.

Enumerations

- enum **HUDAQSTATUS** { **HUDAQSUCCESS** = 0 }

  Return codes for HUDAQ functions.

- enum **HudaqSubsystems** {

  HudaqDI = 0x1000, **HudaqDO** = 0x2000, **HudaqAI** = 0x3000, **HudaqAO** = 0x4000,

  HudaqEnc = 0x5000, **HudaqCtr** = 0x6000, **HudaqStep** = 0x7000, **HudaqSubsystemMASK** = 0xFFFFF000 }

  Constants for identification of subsystems.

- enum **HudaqAIOUnits** { **HudaqAIOVolts** = 0, **HudaqAIORaw** = 1 }

  Allowed values for HudaqAOUNITS.

3.10.2 Enumeration Type Documentation

3.10.2.1 enum HudaqAIOUnits

Allowed values for HudaqAOUNITS.

Used in both Analog inputs and Analog outputs channels. See **HudaqAIUNITS**(p. 21) and **HudaqAOUNITS**(p. 21).

Enumerator:

HudaqAIOVolts measure analog value in volts

HudaqAIORaw get raw value of A/D

3.10.2.2 enum HUDAQSTATUS

Return codes for HUDAQ functions.

Enumerator:

HUDAQSUCCESS Success. All other values mean failure.

3.10.2.3 enum HudaqSubsystems

Constants for identification of subsystems.

Used inside HudaqParameter.

Enumerator:

HudaqDI Digital inputs.

HudaqDO Digital outputs.

HudaqAI Analog inputs.

HudaqAO Analog outputs.

HudaqEnc Encoders.

HudaqCtr Counters.

HudaqStep Stepper motors.

HudaqSubsystemMASK Mask that spreads subsystem number and its parameter.

Chapter 4

Humusoft Data Acquisition Library Example Documentation

4.1 AIRead.c

```
1 /* Humusoft data acquisition library.
2  * Example that shows reading of analog input channels
3  * using the function to read a single channel.
4  */
5
6 /* Copyright 2002-2006 Humusoft s.r.o. */
7
8 #include <stdio.h>
9
10 #include "hudaqlib.h"
11
12
13 int main(int argc, char* argv[])
14 {
15   HUDAQHANDLE h;
16   unsigned i;
17   double value;
18
19   /* open a handle to the first MF624 device in the system */
20   h = HudaqOpenDevice("MF624", 1, 0);
21   if (h==0)
22   {
23     printf("\nData acquisition device not found.\n");
24     return(-1);
25   }
26
27   /* read all the 8 analog inputs in a loop, print their values */
28   for (i=0; i<8; i++)
29   {
30     value = HudaqAIRead(h,i);
31     printf("Analog channel %d, value read %fV.\n", i, value);
32   }
33
34   /* close the device handle */
35   HudaqCloseDevice(h);
36
37   return(0);
38 }
39
```

4.2 AIReadMultiple.c

```
1 /* Humusoft data acquisition library.
2  *    Copyright 2002-2006 Humusoft s.r.o.
3  *
4  * Example that shows reading of analog input channels
5  * using the function to read multiple channels together.
6  */
7
8 #include <stdio.h>
9
10 #include "hudaqlib.h"
11
12
13 int main(int argc, char* argv[])
14 {
15   HUDAQHANDLE h;
16   unsigned i;
17         /* Buffer for channel numbers. Order of channels is not signifficant.
18            Duplicated channels are also tolerated. */
19   unsigned channels[8] = {4,5,6,7,0,1,2,3};
20         /* Buffer for receiving values read. Is size must correspond to
21            buffer of channels. */
22   double values[8];
23
24   /* open a handle to the first MF624 device in the system */
25   h = HudaqOpenDevice("MF624", 1, 0);
26   if (h==0)
27   {
28     printf("\nData acquisition device not found.\n");
29     return(-1);
30   }
31
32   /* read all the 8 analog inputs together */
33   HudaqAIReadMultiple(h,8,channels,values);
34
35   /* print values read */
36   for (i=0; i<8; i++)
37   {
38     printf("Analog channel %d, value read %fV.\n", channels[i], values[i]);
39   }
40
41   /* close the device handle */
42   HudaqCloseDevice(h);
43
44   return(0);
45 }
46
```

4.3 AOWrite.c

```
1 /* Humusoft data acquisition library.
2  * Example that shows writing to analog output channels
3  * using the function to write a single channel.
4  */
5
6 /* Copyright 2002-2006 Humusoft s.r.o. */
7
8 #include <stdio.h>
9
10 #include "hudaqlib.h"
11
12
13 int main(int argc, char* argv[])
14 {
15   HUDAQHANDLE h;
16   unsigned i;
17   double value;
18
19   /* open a handle to the first MF624 device in the system */
20   h = HudaqOpenDevice("MF624", 1, 0);
21   if (h==0)
22   {
23     printf("\nData acquisition device not found.\n");
24     return(-1);
25   }
26
27   /* write all the 8 analog outputs in a loop */
28   /* the voltage written to the output is computed as (channel number - 5) */
29   for (i=0; i<8; i++)
30   {
31     value = i-5;
32     HudaqAOWrite(h, i, value);
33     printf("Analog output channel %d, value written %fV.\n", i, value);
34   }
35
36   /* close the device handle */
37   HudaqCloseDevice(h);
38
39   return(0);
40 }
41
```

4.4 AOWriteMultiple.c

```
1 /* Humusoft data acquisition library.
2  *     Copyright 2002-2006 Humusoft s.r.o.
3  *
4  * Example that shows writing to analog output channels
5  * using the function to write multiple channels together.
6  */
7
8 #include <stdio.h>
9
10 #include "hudaqlib.h"
11
12
13 int main(int argc, char* argv[])
14 {
15   HUDAQHANDLE h;
16         /* Buffer for channel numbers. Order of channels is not signifficant.
17            Duplicated channels are also tolerated. */
18   unsigned channels[8] = {4,5,6,7,0,1,2,3};
19         /* Buffer that contains values to be written.
20            Is size must correspond to buffer of channels. */
21   double values[8] = {5.0, 6.0, 7.0, 8.0, 1.0, 2.0, 3.0, 4.0};
22
23   /* Open a handle to the first MF624 device in the system. */
24   h = HudaqOpenDevice("MF624", 1, 0);
25   if (h==0)
26   {
27     printf("\nData acquisition device not found.\n");
28     return(-1);
29   }
30
31   /* Write all the 8 analog outputs in one call. */
32   if(HudaqAOWriteMultiple(h, 8, channels, values)==HUDAQSUCCESS)
33   {
34     printf("\nData has been written.\n");
35   }
36
37   /* Close the device handle. */
38   HudaqCloseDevice(h);
39
40   return(0);
41 }
42
```

4.5 CtrRead.c

```
1 /* Humusoft data acquisition library.
2  *     Copyright 2002-2006 Humusoft s.r.o.
3  *
4  * Example that shows reading of counters and counting pulses.
5  */
6
7 #include <stdio.h>
8
9 #include "hudaqlib.h"
10
11
12 int main(int argc, char* argv[])
13 {
14   HUDAQHANDLE h;
15   unsigned i;
16   int value;
17
18   /* open a handle to the first MF624 device in the system */
19   h = HudaqOpenDevice("MF624", 1, 0);
20   if (h==0)
21   {
22     printf("\nData acquisition device not found.\n");
23     return(-1);
24   }
25
26   /* Do reset of counters. Each counter is switched to counting mode
27      after its first ussage. */
28   for (i=0; i<4; i++)
29   {
30     HudaqCtrReset(h,i);
31   }
32
33   printf("Counting external pulses by counters, press Enter to continue.\n", i, value);
34   getchar();
35
36   /* read all the 4 counters in a loop, print their values */
37   for (i=0; i<4; i++)
38   {
39     value = HudaqCtrRead(h,i);
40     printf("Counter channel %d, value read %d.\n", i, value);
41   }
42
43   /* close the device handle */
44   HudaqCloseDevice(h);
45
46   return(0);
47 }
48
```

4.6 DIRead.c

```
1 /* Humusoft data acquisition library.
2  * Example that shows reading of digital input channels
3  * using the function to read a single channel.
4  */
5
6 /* Copyright 2002-2006 Humusoft s.r.o. */
7
8 #include <stdio.h>
9
10 #include "hudaqlib.h"
11
12
13 int main(int argc, char* argv[])
14 {
15   HUDAQHANDLE h;
16   unsigned i;
17   double value;
18
19   /* open a handle to the first MF624 device in the system */
20   h = HudaqOpenDevice("MF624", 1, 0);
21   if (h==0)
22   {
23     printf("\nData acquisition device not found.\n");
24     return(-1);
25   }
26
27   /* read whole digital channel at once */
28   value = HudaqDIRead(h,0);
29   printf("\nValue read from digital channel 0: %Xh ", value);
30
31   /* close the device handle */
32   HudaqCloseDevice(h);
33
34   return(0);
35 }
36
```

4.7 DIReadBit.c

```
1 /* Humusoft data acquisition library.
2  * Example that shows reading of digital input channels using
3  * the function to read separate bits from a single channel.
4  */
5
6 /* Copyright 2002-2006 Humusoft s.r.o. */
7
8 #include <stdio.h>
9
10 #include "hudaqlib.h"
11
12
13 int main(int argc, char* argv[])
14 {
15   HUDAQHANDLE h;
16   unsigned i;
17   double value;
18
19   /* open a handle to the first MF624 device in the system */
20   h = HudaqOpenDevice("MF624", 1, 0);
21   if (h==0)
22   {
23     printf("\nData acquisition device not found.\n");
24     return(-1);
25   }
26
27   /* read all 8 bits from digital inputs in a loop, print their values */
28   for(i=0; i<8; i++)
29   {
30     value = HudaqDIReadBit(h,0,i);  /* Read one bit from digital input */
31     printf("bit:%d, %d ", i, value);
32   }
33
34   /* close the device handle */
35   HudaqCloseDevice(h);
36
37   return(0);
38 }
39
```

4.8 DOWrite.c

```
1 /* Humusoft data acquisition library.
2  * Example that shows writing all bits to a digital output channels
3  * using the function to write a single channel.
4  */
5
6 /* Copyright 2002-2006 Humusoft s.r.o. */
7
8 #include <stdio.h>
9
10 #include "hudaqlib.h"
11
12
13 int main(int argc, char* argv[])
14 {
15   HUDAQHANDLE h;
16   unsigned i;
17
18   /* open a handle to the first MF624 device in the system */
19   h = HudaqOpenDevice("MF624", 1, 0);
20   if (h==0)
21   {
22     printf("\nData acquisition device not found.\n");
23     return(-1);
24   }
25
26   /* write 0xFF to whole digital channel at once */
27   HudaqDOWrite(h,0,0xFF);
28
29   printf("\n0xFF was writen to digital output. Press any key to continue.");
30   getchar();
31
32   /* write 0x00 to whole digital channel at once */
33   HudaqDOWrite(h,0,0x0);
34   printf("\n0x00 has been writen to digital output.");
35
36   /* close the device handle */
37   HudaqCloseDevice(h);
38
39   return(0);
40 }
41
```

4.9 DOWriteBit.c

```
1 /* Humusoft data acquisition library.
2  *     Copyright 2002-2006 Humusoft s.r.o.
3  *
4  * Example that shows writing separate bits to a digital output channel
5  * using the function to write a single bit.
6  */
7
8 #include <stdio.h>
9
10 #include "hudaqlib.h"
11
12
13 int main(int argc, char* argv[])
14 {
15   HUDAQHANDLE h;
16   unsigned i;
17
18   /* open a handle to the first MF624 device in the system */
19   h = HudaqOpenDevice("MF624", 1, 0);
20   if (h==0)
21   {
22     printf("\nData acquisition device not found.\n");
23     return(-1);
24   }
25
26   /* HudaqOpenDevice initialises all digital output bits to 0 */
27   for(i=0; i<8; i++)
28   {
29     printf("\nPress any key to set a bit %d to '1'", i);
30     getchar();
31     HudaqDOWriteBit(h, 0, i, 1);
32   }
33
34   /* close the device handle */
35   HudaqCloseDevice(h);
36
37   return(0);
38 }
39
```

4.10 EncCtrRead.c

```
1 /* Humusoft data acquisition library.
2  *    Copyright 2002-2006 Humusoft s.r.o.
3  *
4  * Example that shows configuration of
5  * encoder.
6  */
7
8 #include <stdio.h>
9
10 #include "hudaqlib.h"
11
12
13 int main(int argc, char* argv[])
14 {
15   HUDAQHANDLE h;
16   unsigned i;
17   int value;
18
19   /* open a handle to the first MF624 device in the system */
20   h = HudaqOpenDevice("MF624", 1, 0);
21   if (h==0)
22   {
23     printf("\nData acquisition device not found.\n");
24     return(-1);
25   }
26
27   /* Configure encoder to count input pulses. */
28   if(HudaqSetParameter(h, 0, HudaqEncMODE, HudaqEncMODERISING) != HUDAQSUCCESS)
29   {
30     printf("\nCannot switch counting mode.\n");
31     HudaqCloseDevice(h);
32     return(-2);
33   }
34
35   /* Turn on hardware filter for input signal. */
36   if(HudaqSetParameter(h, 0, HudaqEncFILTER, 1) != HUDAQSUCCESS)
37   {
38     printf("\nCannot filter input signal.\n");
39   }
40
41   printf("Counting external IRC pulses by encoders, press Enter to continue.\n", i, value);
42   getchar();
43
44   /* Read encoder 0 value, print it. */
45   value = HudaqEncRead(h,0);
46   printf("Encoder channel 0, value read %d.\n", 0, value);
47
48   /* close the device handle */
49   HudaqCloseDevice(h);
50
51   return(0);
52 }
53
```

4.11 EncRead.c

```
1 /* Humusoft data acquisition library.
2  *    Copyright 2002-2006 Humusoft s.r.o.
3  *
4  * Example that shows decoding IRC position
5  * using the function to read a single encoder.
6  */
7
8 #include <stdio.h>
9
10 #include "hudaqlib.h"
11
12
13 int main(int argc, char* argv[])
14 {
15   HUDAQHANDLE h;
16   unsigned i;
17   int value;
18
19   /* open a handle to the first MF624 device in the system */
20   h = HudaqOpenDevice("MF624", 1, 0);
21   if (h==0)
22   {
23     printf("\nData acquisition device not found.\n");
24     return(-1);
25   }
26
27   printf("Counting external IRC pulses by encoders, press Enter to continue.\n", i, value);
28   getchar();
29
30   /* read all the 4 encoder values in a loop, print them */
31   for (i=0; i<4; i++)
32   {
33     value = HudaqEncRead(h,i);
34     printf("Encoder channel %d, value read %d.\n", i, value);
35   }
36
37   /* close the device handle */
38   HudaqCloseDevice(h);
39
40   return(0);
41 }
42
```

4.12 PWMWrite.c

```
1 /* Humusoft data acquisition library.
2  *    Copyright 2002-2006 Humusoft s.r.o.
3  *
4  * Example that shows writing to analog output channels
5  * using the function to write a single channel.
6  */
7
8 #include <stdio.h>
9
10 #include "hudaqlib.h"
11
12
13 int main(int argc, char* argv[])
14 {
15   HUDAQHANDLE h;
16   unsigned i;
17   double value;
18
19   /* open a handle to the first MF624 device in the system */
20   h = HudaqOpenDevice("MF624", 1, 0);
21   if (h==0)
22   {
23     printf("\nData acquisition device not found.\n");
24     return(-1);
25   }
26
27   /* set first PWM channel to frequency 1.5kHz with duty cycle 0.5 */
28   HudaqPWMWrite(h,0,1500,0.5);
29
30   /* set second PWM channel to frequency 2.5kHz with duty cycle 0.75 */
31   HudaqPWMWrite(h,1,2500,0.75);
32
33   /* close the device handle */
34   HudaqCloseDevice(h);
35
36   return(0);
37 }
38
```