

# Humusoft Data Acquisition Library Reference Manual

Generated by Doxygen 1.4.6-NO

Thu Jul 13 10:09:41 2006



# Contents

<b>1</b>	<b>Humusoft Data Acquisition Library</b>	<b>1</b>
1.1	Introduction . . . . .	1
1.2	Installation . . . . .	1
1.3	Basic concepts of working with the library . . . . .	1
1.4	Building applications with the library . . . . .	1
<b>2</b>	<b>Humusoft Data Acquisition Library Module Index</b>	<b>3</b>
2.1	Humusoft Data Acquisition Library Modules . . . . .	3
<b>3</b>	<b>Humusoft Data Acquisition Library Module Documentation</b>	<b>5</b>
3.1	Board Initialization . . . . .	5
3.2	Analog Input . . . . .	7
3.3	Analog Output . . . . .	9
3.4	Digital Input . . . . .	11
3.5	Digital Output . . . . .	13
3.6	Counter Input . . . . .	15
3.7	Encoder Input . . . . .	17
3.8	PWM Output . . . . .	20
3.9	Configuration of channels . . . . .	21
3.10	General purpose Constants and Data Types . . . . .	23
<b>4</b>	<b>Humusoft Data Acquisition Library Example Documentation</b>	<b>25</b>
4.1	AIRead.c . . . . .	25
4.2	AIReadMultiple.c . . . . .	26
4.3	AOWrite.c . . . . .	27
4.4	AOWriteMultiple.c . . . . .	28
4.5	CtrRead.c . . . . .	29
4.6	DRead.c . . . . .	30
4.7	DReadBit.c . . . . .	31

4.8	DOWrite.c . . . . .	32
4.9	DOWriteBit.c . . . . .	33
4.10	EncCtrRead.c . . . . .	34
4.11	EncRead.c . . . . .	35
4.12	PWMWrite.c . . . . .	36

# 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 e.g. `C:\WIN2000\system32\`. 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 - i.e., if a device has eight channels, the last channel has number 7.

For information about device capabilities see device documentation.

### 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 respective 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.lib** 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.lib**. 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 can be done using this command:

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

## 1.5 Copyright

### Author:

Copyright 2002-2006 Humusoft s.r.o.

No part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written consent of HUMUSOFT s.r.o.

Limited Warranty: HUMUSOFT s.r.o. disclaims all liability for any direct or indirect damages caused by use or misuse of the HUDAQ library or this documentation.

HUMUSOFT is a registered trademark of HUMUSOFT s.r.o.

Other brand and product names are trademarks or registered trademarks of their respective holders.

Chapter 2

Humusoft Data Acquisition Library  
Module Index

2.1 Humusoft Data Acquisition Library Modules

Here is a list of all modules:

Board Initialization . . . . .	5
Analog Input . . . . .	7
Analog Output . . . . .	9
Digital Input . . . . .	11
Digital Output . . . . .	13
Counter Input . . . . .	15
Encoder Input . . . . .	17
PWM Output . . . . .	20
Channel Configuration . . . . .	??
General Purpose Constants and Data Types . . . . .	23





Chapter 3

Humusoft Data Acquisition Library  
Page Index

3.1 Humusoft Data Acquisition Library Related Pages

Here is a list of all related documentation pages:

Feature list of AD612 device. . . . .	??
Feature list of MF614 device. . . . .	??
Feature list of MF624 device. . . . .	??



## Chapter 4

# Humusoft Data Acquisition Library Module Documentation

### 4.1 Board Initialization

#### 4.1.1 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.*

#### 4.1.2 Function Documentation

##### 4.1.2.1 HUDAQHANDLE HudaqOpenDevice (const char \* *devicename*, int *deviceorder*, int *options*)

Open a data acquisition device.

The device is put into initial state when being opened.

**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, DOWriteMultipleBits.c, EncConfig.c, EncRead.c, and PWMWrite.c.

**4.1.2.2 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.

**4.1.2.3 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 closing 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, DOWriteMultipleBits.c, EncConfig.c, EncRead.c, and PWMWrite.c.

## 4.2 Analog Input

### 4.2.1 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:

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

### Functions

- double **HudaqARead** (**HUDAQHANDLE** handle, unsigned channel)  
*Read data from a single analog input channel.*
- **HUDAQSTATUS** **HudaqAReadMultiple** (**HUDAQHANDLE** handle, unsigned number, const unsigned \*channels, double \*values)  
*Read data from multiple analog input channels.*

### 4.2.2 Function Documentation

#### 4.2.2.1 double HudaqARead (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:

**ARead.c.**

#### 4.2.2.2 HUDAQSTATUS HudaqAReadMultiple (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 this feature.

**Parameters:**

- ← *handle* Device handle.
- ← *number* Number of channels to read.
- ← *channels* Array of channel numbers 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:**

**AIRReadMultiple.c.**

## 4.3 Analog Output

### 4.3.1 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:

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

### 4.3.2 Function Documentation

#### 4.3.2.1 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.

#### 4.3.2.2 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 this feature.

**Parameters:**

- ← *handle* Device handle.
- ← *number* Number of channels to be written.
- ← *channels* Array of channel numbers 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.**



## 4.4 Digital Input

### 4.4.1 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:**

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

### 4.4.2 Function Documentation

#### 4.4.2.1 **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.**

#### 4.4.2.2 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.

#### 4.4.2.3 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 channel numbers 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.

## 4.5 Digital Output

### 4.5.1 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:**

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

### Functions

- void **HudaqDOWriteBit** (**HUDAQHANDLE** handle, unsigned channel, unsigned bit, int 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 output channels.*

### 4.5.2 Function Documentation

#### 4.5.2.1 void HudaqDOWriteBit (**HUDAQHANDLE** *handle*, unsigned *channel*, unsigned *bit*, int *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.

#### 4.5.2.2 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.

##### Examples:

DOWrite.c.

#### 4.5.2.3 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.

##### Examples:

DOWriteMultipleBits.c.

#### 4.5.2.4 HUDAQSTATUS HudaqDOWriteMultiple (HUDAQHANDLE *handle*, unsigned *number*, const unsigned \* *channels*, const unsigned \* *values*)

Write data to multiple digital output channels.

##### Parameters:

- ← *handle* Device handle.
- ← *number* Number of channels to be written.
- ← *channels* Array of channel numbers 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.

## 4.6 Counter Input

### 4.6.1 Detailed Description

Counter input routines read the counter pulse count.

After the counter hardware is switched to counting mode, external input on rising edge (input mode **HudaqCtrCLOCKINRISING**(p.??)) is selected as the default counter input. The input of individual counter channels can then be changed by a call to **HudaqSetParameter**(p.??). 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.

### Functions

- void **HudaqCtrReset** (**HUDAQHANDLE** handle, unsigned channel)  
*Reset counter pulse count.*
- int **HudaqCtrRead** (**HUDAQHANDLE** handle, unsigned channel)  
*Read counter pulse count.*

### 4.6.2 Function Documentation

#### 4.6.2.1 void HudaqCtrReset (HUDAQHANDLE *handle*, unsigned *channel*)

Reset counter pulse count.

If the counter hardware is used by another subsystem, it is switched to counting mode and the default input is selected. If the counter is already in counting mode, its input selection is not changed.

#### Parameters:

- ← *handle* Device handle.
- ← *channel* Number of counter channel.

#### Examples:

**CtrRead.c.**

#### 4.6.2.2 int HudaqCtrRead (HUDAQHANDLE *handle*, unsigned *channel*)

Read counter pulse count.

The returned 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`.

## 4.7 Encoder Input

### 4.7.1 Detailed Description

Encoder input routines read the encoder pulse count.

Each encoder channel has three inputs *A*, *B* and *I*, which allow direct connection of quadrature encoders with index output. In default mode (mode **HudaqEncMODEIRC**(p.??)) the *A* and *B* inputs expect signal from quadrature encoder pulse outputs and the *I* input connects to the encoder index pulse output.

Encoders could be also configured as bidirectional counters. In this mode, that count pulses on input *A* and input *B* specifies count direction. For details see **HudaqEncMode**(p.??).

The *I* input works in any of the encoder modes and its functionality is programmable - see **HudaqEncResetMode**(p.??) for details.

#### Device capabilities:

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

### Functions

- void **HudaqEncReset** (**HUDAQHANDLE** handle, unsigned channel)  
*Reset encoder pulse count.*
- int **HudaqEncRead** (**HUDAQHANDLE** handle, unsigned channel)  
*Read encoder pulse count.*

### 4.7.2 Function Documentation

#### 4.7.2.1 void HudaqEncReset (HUDAQHANDLE handle, unsigned channel)

Reset encoder pulse count.

##### Parameters:

- ← *handle* Device handle.
- ← *channel* Encoder input channel number.

#### 4.7.2.2 int HudaqEncRead (HUDAQHANDLE handle, unsigned channel)

Read encoder pulse count.

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

##### Parameters:

- ← *handle* Device handle.

← *channel* Encoder input channel number.

**Returns:**

Value read from the encoder input channel.

**Examples:**

**EncConfig.c**, and **EncRead.c**.



## 4.8 PWM Output

### 4.8.1 Detailed Description

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

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:**

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

### Functions

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

*Generate pulse-width modulation signal on a counter output.*

### 4.8.2 Function Documentation

#### 4.8.2.1 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.

## 4.9 Channel Configuration

### 4.9.1 Detailed Description

Some channels can perform different functions based on their parameters.

The parameters are specific for different subsystems and their values are specific to the respective parameter. The channel parameter setting persists until changed or until the device is reset.

A particular device does not necessarily support all parameters and parameter values. See description of individual parameters for details.

### Enumerations

- enum **HudaqSubsystem** {  
    **HudaqAI** = 0x1000,  
    **HudaqAO** = 0x2000,  
    **HudaqDI** = 0x3000,  
    **HudaqDO** = 0x4000,  
    **HudaqEnc** = 0x5000,  
    **HudaqCtr** = 0x6000,  
    **HudaqPWM** = 0x7000 }  
    *Subsystem identifiers.*
- enum **HudaqParameter** {  
    **HudaqAIRange**,  
    **HudaqAORange**,  
    **HudaqEncRESETONREAD** = **HudaqEnc**,  
    **HudaqEncFILTER**,  
    **HudaqEncMODE**,  
    **HudaqEncCOUNTCONTROL**,  
    **HudaqEncRESETMODE**,  
    **HudaqCtrRESETONREAD** = **HudaqCtr**,  
    **HudaqCtrCLOCKSOURCE** }  
    *Parameter identifiers.*
- enum **HudaqCtrClockSource** {  
    **HudaqCtrCLOCK50MHz** = 0,  
    **HudaqCtrCLOCK10MHz** = 1,  
    **HudaqCtrCLOCK1MHz** = 2,  
    **HudaqCtrCLOCK100kHz** = 3,  
    **HudaqCtrCLOCKINRISING** = 5,  
    **HudaqCtrCLOCKINFALLING** = 6,  
    **HudaqCtrCLOCKINEITHER** = 7,  
    **HudaqCtrCLOCKPREVRISING** = 9,

```

HudaqCtrCLOCKPREVFALLING = 10,
HudaqCtrCLOCKPREVEITHER = 11,
HudaqCtrCLOCKNEXTRISING = 13,
HudaqCtrCLOCKNEXTFALLING = 14,
HudaqCtrCLOCKNEXTEITHER = 15 }

```

*Counter clock sources.*

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

*Encoder counting modes.*

- enum **HudaqEncCountControl** {  
**HudaqEncCOUNTENABLE** = 0,  
**HudaqEncCOUNTDISABLE**,  
**HudaqEncCOUNTI0**,  
**HudaqEncCOUNTI1** }

*Encoder count control.*

- enum **HudaqEncResetMode** {  
**HudaqEncRESNONE** = 0,  
**HudaqEncRESPERMANENT**,  
**HudaqEncRESI0**,  
**HudaqEncRESI1**,  
**HudaqEncRESIRISING**,  
**HudaqEncRESIFALLING**,  
**HudaqEncRESIEITHER** }

*Encoder reset mode.*

## Functions

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

*Reads single channel configuration for a given subsystem.*

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

*Configures single channel of a given subsystem.*

- const **HudaqRange** \* **HudaqQueryRange** (**HUDAQHANDLE** handle, **HudaqSubsystem** S, unsigned item)

*Query voltage ranges by their indices.*

## 4.9.2 Enumeration Type Documentation

### 4.9.2.1 enum HudaqSubsystem

Subsystem identifiers.

Used to identify individual subsystems of the board.

**Enumerator:**

*HudaqAI* Analog Input.  
*HudaqAO* Analog Output.  
*HudaqDI* Digital Input.  
*HudaqDO* Digital Output.  
*HudaqEnc* Encoder.  
*HudaqCtr* Counter.  
*HudaqPWM* Pulse-Width Modulation Output.

### 4.9.2.2 enum HudaqParameter

Parameter identifiers.

They are used as the third parameter to **HudaqGetParameter(p.??)** and **HudaqSetParameter(p.??)** functions to specify which parameter should be configured. Please note that a particular device does not necessarily support all functions.

**Enumerator:**

*HudaqAIRange* Select analog input voltage range. The range is selected by its index; the actual voltages corresponding to the range can be obtained by calling **HudaqQueryRange(p.??)**.

*HudaqAORange* Select analog output voltage range. The range is selected by its index; the actual voltages corresponding to the range can be obtained by calling **HudaqQueryRange(p.??)**.

*HudaqEncRESETONREAD* Automatically reset encoder pulse count after it is read; possible values are 0 (off) or 1 (on).

*HudaqEncFILTER* Filter encoder inputs with a lowpass filter; possible values are 0 (off) or 1 (on).

*HudaqEncMODE* Encoder mode; for possible values see **HudaqEncMode(p.??)**.

*HudaqEncCOUNTCONTROL* Encoder count control; for possible values see **HudaqEncCountControl(p.??)**.

*HudaqEncRESETMODE* Encoder reset mode; for possible values see **HudaqEncResetMode(p.??)**.

*HudaqCtrRESETONREAD* Automatically reset counter pulse count after it is read; possible values are 0 (off) or 1 (on).

*HudaqCtrCLOCKSOURCE* Counter clock source; for possible values see **HudaqCtrClockSource(p.??)**.

### 4.9.2.3 enum HudaqCtrClockSource

Counter clock sources.

Enumerator:

- HudaqCtrCLOCK50MHz*** Internal clock 50MHz. (MF624 only).
- HudaqCtrCLOCK10MHz*** Internal clock 10MHz. (MF624 only).
- HudaqCtrCLOCK1MHz*** Internal clock 1MHz. (MF624 only).
- HudaqCtrCLOCK100kHz*** Internal clock 100kHz. (MF624 only).
- HudaqCtrCLOCKINRISING*** External input, count on rising edge. (MF614 and MF624).
- HudaqCtrCLOCKINFALLING*** External input, count on falling edge.(MF614 and MF624).
- HudaqCtrCLOCKINEITHER*** External input, count on either edge. (MF624 only).
- HudaqCtrCLOCKPREVRISING*** Output of previous counter, count on rising edge. (MF614 and MF624).
- HudaqCtrCLOCKPREVFALLING*** Output of previous counter, count on falling edge. (MF614 and MF624).
- HudaqCtrCLOCKPREVEITHER*** Output of previous counter, count on either edge. (MF624 only).
- HudaqCtrCLOCKNEXTRISING*** Output of next counter, count on rising edge. (MF624 only).
- HudaqCtrCLOCKNEXTFALLING*** Output of next counter, count on falling edge. (MF624 only).
- HudaqCtrCLOCKNEXTEITHER*** Output of next counter, count on either edge. (MF624 only).

### 4.9.2.4 enum HudaqEncMode

Encoder counting modes.

Enumerator:

- HudaqEncMODEIRC*** Count IRC pulses on inputs *A* and *B* (MF614 and MF624).
- HudaqEncMODERISING*** Count pulses on *A*, rising edge, *B* specifies direction (MF614 and MF624).
- HudaqEncMODEFALLING*** Count pulses on *A*, falling edge, *B* specifies direction (MF624 only).
- HudaqEncMODEEITHER*** Count pulses on *A*, either edge, *B* specifies direction (MF624 only).

### 4.9.2.5 enum HudaqEncCountControl

Encoder count control.

Encoder count control allows enabling or disabling pulse counting based on software or hardware conditions.

**Enumerator:**

**HudaqEncCOUNTENABLE** Encoder counting is enabled. (MF614 and MF624).

**HudaqEncCOUNTDISABLE** Encoder counting is disabled. (MF624 only).

**HudaqEncCOUNTI0** Encoder counting is enabled when input *I* is at logical low, disabled when input *I* is at logical high. (MF624 only).

**HudaqEncCOUNTI1** Encoder counting is enabled when input *I* is at logical high, disabled when input *I* is at logical low. (MF624 only).

**4.9.2.6 enum HudaqEncResetMode**

Encoder reset mode.

Encoder reset mode allows enabling the functionality of resetting the encoder pulse count by external signal.

**Enumerator:**

**HudaqEncRESNONE** Encoder is not being reset by external signal. (MF614 and MF624).

**HudaqEncRESPERMANENT** Encoder is permanently reset; the encoder does not count in this mode. (MF624 only).

**HudaqEncRESI0** Reset encoder pulse count when input *I* is at logical low. (MF614 and MF624).

**HudaqEncRESI1** Reset encoder pulse count when input *I* is at logical high. (MF614 and MF624).

**HudaqEncRESIRISING** Reset encoder pulse count on rising edge of input *I* . (MF624 only).

**HudaqEncRESIFALLING** Reset encoder pulse count on falling edge of input *I* . (MF624 only).

**HudaqEncRESIEITHER** Reset encoder pulse count on either edge of input *I* . (MF624 only).

**4.9.3 Function Documentation****4.9.3.1 double HudaqGetParameter (HUDAQHANDLE *handle*, unsigned *channel*, HudaqParameter *param*)**

Reads single channel configuration for a given subsystem.

Not all devices support all the parameters.

**Parameters:**

← ***handle*** Device handle.

← ***channel*** Channel number. The channel type is determined by the parameter identifier ***param***.

← ***param*** Parameter identifier; see **HudaqParameter(p.??)** for possible values.

**Returns:**

value Value of the parameter; see individual parameter descriptions for description of the values.

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

Configures single channel of a given subsystem.

Not all devices support all the parameters and all the parameter values.

#### Parameters:

- ← *handle* Device handle.
- ← *channel* Channel number. The channel type is determined by the parameter identifier *param*.
- ← *param* Parameter identifier; see **HudaqParameter**(p. ??) for possible values.
- ← *value* Value of the parameter; see individual parameter descriptions for possible values.

#### Returns:

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

#### Examples:

EncConfig.c.

### 4.9.3.3 const HudaqRange\* HudaqQueryRange (HUDAQHANDLE *handle*, HudaqSubsystem *S*, unsigned *item*)

Query voltage ranges by their indices.

This function translates the voltage range index to actual voltage range limits for the device. No change is done to device configuration; use **HudaqSetParameter**(p. ??) to set a voltage range by its index; use **HudaqGetParameter**(p. ??) to get the index of the currently set voltage range.

#### Parameters:

- ← *handle* Device handle.
- ← *S* Subsystem identifier; possible values are **HudaqAI**(p. ??) or **HudaqAO**(p. ??).
- ← *item* Zero-based voltage range index.

#### Returns:

Pointer to a structure containing the voltage range limits for the specified index, or NULL for an invalid range index. The structure pointed to by the returned pointer is defined like this:

```
typedef struct
{
    double Lo;
    double Hi;
} HudaqRange;
```

## 4.10 General Purpose Constants and Data Types

### 4.10.1 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.*

### 4.10.2 Enumeration Type Documentation

#### 4.10.2.1 enum HUDAQSTATUS

Return codes for HUDAQ functions.

##### Enumerator:

**HUDAQSUCCESS** Success. All other values mean failure.



## Chapter 5

# Humusoft Data Acquisition Library Example Documentation

### 5.1 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
```

## 5.2 AIReadMultiple.c

```
1 /* Humusoft data acquisition library.
2 *
3 * Example that shows reading of analog input channels
4 * using the function to read multiple channels together.
5 */
6
7 /* Copyright 2002-2006 Humusoft s.r.o. */
8
9 #include <stdio.h>
10
11 #include "hudaqlib.h"
12
13
14 int main(int argc, char* argv[])
15 {
16     HUDAQHANDLE h;
17     unsigned i;
18     /* Buffer for channel numbers. Order of channels is not significant.
19      * Duplicated channels are also supported. */
20     unsigned channels[8] = {4,5,6,7,0,1,2,3};
21     /* Buffer for receiving values read. Its size must correspond to
22      * buffer of channels. */
23     double values[8];
24
25     /* open a handle to the first MF624 device in the system */
26     h = HudaqOpenDevice("MF624", 1, 0);
27     if (h==0)
28     {
29         printf("\nData acquisition device not found.\n");
30         return(-1);
31     }
32
33     /* read all the 8 analog inputs together */
34     HudaqAIReadMultiple(h,8,channels,values);
35
36     /* print values read */
37     for (i=0; i<8; i++)
38     {
39         printf("Analog channel %d, value read %fV.\n", channels[i], values[i]);
40     }
41
42     /* close the device handle */
43     HudaqCloseDevice(h);
44
45     return(0);
46 }
47
```

## 5.3 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
```

## 5.4 AOWriteMultiple.c

```
1 /* Humusoft data acquisition library.
2 *
3 * Example that shows writing to analog output channels
4 * using the function to write multiple channels together.
5 */
6
7 /* Copyright 2002-2006 Humusoft s.r.o. */
8
9 #include <stdio.h>
10
11 #include "hudaqlib.h"
12
13
14 int main(int argc, char* argv[])
15 {
16     HUDAQHANDLE h;
17     /* Buffer for channel numbers. Order of channels is not significant.
18        Duplicated channels are also supported. */
19     unsigned channels[8] = {4,5,6,7,0,1,2,3};
20     /* Buffer that contains values to be written.
21        Is size must correspond to buffer of channels. */
22     double values[8] = {5.0, 6.0, 7.0, 8.0, 1.0, 2.0, 3.0, 4.0};
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     /* Write all the 8 analog outputs in one call. */
33     if(HudaqAOWriteMultiple(h, 8, channels, values)==HUDAQSUCCESS)
34     {
35         printf("\nData has been written.\n");
36     }
37
38     /* Close the device handle. */
39     HudaqCloseDevice(h);
40
41     return(0);
42 }
43
```

## 5.5 CtrRead.c

```
1 /* Humusoft data acquisition library.
2 *
3 * Example that shows reading of counters and counting pulses.
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     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     /* Do reset of counters. Each counter is switched to counting mode
28        after its first usage. */
29     for (i=0; i<4; i++)
30     {
31         HudaqCtrReset(h,i);
32     }
33
34     printf("Counting external pulses by counters, press Enter to continue.\n", i, value);
35     getchar();
36
37     /* read all the 4 counters in a loop, print their values */
38     for (i=0; i<4; i++)
39     {
40         value = HudaqCtrRead(h,i);
41         printf("Counter channel %d, value read %d.\n", i, value);
42     }
43
44     /* close the device handle */
45     HudaqCloseDevice(h);
46
47     return(0);
48 }
49
```

## 5.6 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
```

## 5.7 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
```

## 5.8 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
```



## 5.9 DOWriteBit.c

```
1 /* Humusoft data acquisition library.
2 *
3 * Example that shows writing separate bits to a digital output channel
4 * using the function to write a single bit.
5 */
6
7 /* Copyright 2002-2006 Humusoft s.r.o. */
8
9 #include <stdio.h>
10
11 #include "hudaqlib.h"
12
13
14 int main(int argc, char* argv[])
15 {
16     HUDAQHANDLE h;
17     unsigned i;
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     /* HudaqOpenDevice initialises all digital output bits to 0 */
28     for(i=0; i<8; i++)
29     {
30         printf("\nPress any key to set a bit %d to '1'", i);
31         getchar();
32         HudaqDOWriteBit(h, 0, i, 1);
33     }
34
35     /* close the device handle */
36     HudaqCloseDevice(h);
37
38     return(0);
39 }
40
```

## 5.10 DOWriteMultipleBits.c

```
1 /* Humusoft data acquisition library.
2 *
3 * Example that demonstrates using HudaqDOWriteMultipleBits.
4 * This function allows to influence only selected bits from
5 * digital outputs.
6 */
7
8 /* Copyright 2002-2006 Humusoft s.r.o. */
9
10 #include <stdio.h>
11
12 #include "hudaqlib.h"
13
14
15 int main(int argc, char* argv[])
16 {
17     HUDAQHANDLE h;
18     unsigned i;
19
20     /* open a handle to the first MF624 device in the system */
21     h = HudaqOpenDevice("MF624", 1, 0);
22     if (h==0)
23     {
24         printf("\nData acquisition device not found.\n");
25         return(-1);
26     }
27
28     /* HudaqOpenDevice initialises all digital output bits to 0 */
29
30     /* Set first and fifth bits to value '1'. */
31     HudaqDOWriteMultipleBits(h, 0, 0x11, 0x11);
32     printf("\nBits 1 and 5 are set. Press any key to continue.");
33     getchar();
34
35     /* Reset bit 1 and set bit 6. */
36     HudaqDOWriteMultipleBits(h, 0, 0x21, 0x20);
37     printf("\nBit 1 is reset and bit 6 is set. Press any key to continue.");
38     getchar();
39
40
41     /* close the device handle */
42     HudaqCloseDevice(h);
43
44     return(0);
45 }
46
```

## 5.11 EncConfig.c

```
1 /* Humusoft data acquisition library.
2 *
3 * Example that shows configuration of
4 * encoder.
5 */
6
7 /* Copyright 2002-2006 Humusoft s.r.o. */
8
9 #include <stdio.h>
10
11 #include "hudaqlib.h"
12
13
14 int main(int argc, char* argv[])
15 {
16     HUDAQHANDLE h;
17     unsigned i;
18     int value;
19
20     /* open a handle to the first MF624 device in the system */
21     h = HudaqOpenDevice("MF624", 1, 0);
22     if (h==0)
23     {
24         printf("\nData acquisition device not found.\n");
25         return(-1);
26     }
27
28     /* Configure encoder to count input pulses. */
29     if(HudaqSetParameter(h, 0, HudaqEncMODE, HudaqEncMODERISING) != HUDAQSUCCESS)
30     {
31         printf("\nCannot switch encoder to counting mode.\n");
32         HudaqCloseDevice(h);
33         return(-2);
34     }
35
36     /* Turn on hardware filter for input signal. */
37     if(HudaqSetParameter(h, 0, HudaqEncFILTER, 1) != HUDAQSUCCESS)
38     {
39         printf("\nCannot filter input signal.\n");
40     }
41
42     printf("Counting external pulses on input A by encoder, press Enter to continue.\n");
43     getchar();
44
45     /* Read encoder 0 value, print it. */
46     value = HudaqEncRead(h,0);
47     printf("Encoder channel 0, value read %d.\n", 0, value);
48
49     /* close the device handle */
50     HudaqCloseDevice(h);
51
52     return(0);
53 }
54
```

## 5.12 EncRead.c

```
1 /* Humusoft data acquisition library.
2 *
3 * Example that shows decoding IRC position
4 * using the function to read a single encoder.
5 */
6
7 /* Copyright 2002-2006 Humusoft s.r.o. */
8
9 #include <stdio.h>
10
11 #include "hudaqlib.h"
12
13
14 int main(int argc, char* argv[])
15 {
16     HUDAQHANDLE h;
17     unsigned i;
18     int value;
19
20     /* open a handle to the first MF624 device in the system */
21     h = HudaqOpenDevice("MF624", 1, 0);
22     if (h==0)
23     {
24         printf("\nData acquisition device not found.\n");
25         return(-1);
26     }
27
28     printf("Counting external IRC pulses by encoders, press Enter to continue.\n");
29     getchar();
30
31     /* Read all the 4 encoder values in a loop, print them. */
32     for (i=0; i<4; i++)
33     {
34         value = HudaqEncRead(h,i);
35         printf("Encoder channel %d, value read %d.\n", i, value);
36     }
37
38     /* Close the device handle. */
39     HudaqCloseDevice(h);
40
41     return(0);
42 }
43
```

## 5.13 PWMWrite.c

```
1 /* Humusoft data acquisition library.
2 *
3 * Example that shows using of PWM output channels.
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     /* 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
```



## Chapter 6

# Humusoft Data Acquisition Library Page Documentation

### 6.1 Feature list of AD612 device.

#### 6.1.1 Features

- Eight single-ended 12-bit analog input channels, see **HudaqAIRead**(p. 7), **HudaqAIReadMultiple**(p. 7).
- Programmable A/D ranges, see **HudaqGetRange**, **HudaqSetParameter**(p. ??)
- Four 12-bit analog output channels, see **HudaqAOWrite**(p. 9), **HudaqAOWriteMultiple**(p. 9).
- 8 digital inputs, see **HudaqDIRead**(p. 11), **HudaqDIReadMultiple**(p. 12).
- 8 digital outputs, see **HudaqDOWrite**(p. 13), **HudaqDOWriteMultiple**(p. 14).
- Sampling rate up to 108kHz (one channel); 13kHz (eight channels)

#### 6.1.2 Applications

- DC voltage measurement
- Transducer and sensor interfacing
- Vibration and transient analysis
- Process monitoring and control
- Multichannel data acquisition
- Real-time simulation
- Programmable voltage output

#### 6.1.3 Specifications

## 6.2 Feature list of MF614 device.

### 6.2.1 Features

- Eight single-ended 12-bit analog input channels, see **HudaqAIRead**(p. 7), **HudaqAIRead-Multiple**(p. 7).
- Programmable A/D input ranges, see **HudaqGetRange**, **HudaqSetParameter**(p. ??)
- Four 12-bit analog output channels, see **HudaqAOWrite**(p. 9), **HudaqAOWrite-Multiple**(p. 9).
- 8 digital inputs aggregated in one channel, see **HudaqDIRead**(p. 11), **HudaqDIRead-Multiple**(p. 12).
- 8 digital outputs aggregated in one channel, see **HudaqDOWrite**(p. 13), **HudaqDOWrite-Multiple**(p. 14).
- Four quadrature encoder inputs (differential), see **HudaqEncRead**(p. 19), **HudaqEnc-Reset**(p. 19)
- Five counters/timers, see **HudaqCtrRead**(p. 16), **HudaqCtrReset**(p. 16)
- Sampling rate up to 108kHz (one channel); 13kHz (eight channels)

### 6.2.2 Applications

- DC voltage measurement
- Transducer and sensor interfacing
- Vibration and transient analysis
- Process monitoring and control
- Waveform acquisition and analysis
- Multichannel data acquisition
- Real-time simulation
- Programmable voltage output
- Position measurements
- Servo systems
- PWM
- Frequency measurements
- Time measurements
- Pulse/frequency generation
- Pulse counting

### 6.2.3 Specifications



## 6.3 Feature list of MF624 device.

### 6.3.1 Features

- Eight single-ended 14-bit analog input channels, see **HudaqAIRead**(p. 7), **HudaqAIRead-Multiple**(p. 7).
- Eight 14-bit analog output channels, see **HudaqAOWrite**(p. 9), **HudaqAOWrite-Multiple**(p. 9).
- 8 digital inputs aggregated in one channel, see **HudaqDIRead**(p. 11), **HudaqDIRead-Multiple**(p. 12).
- 8 digital outputs aggregated in one channel, see **HudaqDOWrite**(p. 13), **HudaqDOWrite-Multiple**(p. 14).
- Four quadrature encoder inputs (differential), see **HudaqEncRead**(p. 19), **HudaqEnc-Reset**(p. 19)
- Four counters/timers, see **HudaqCtrRead**(p. 16), **HudaqCtrReset**(p. 16)
- Fast conversion rate; up to 250kHz (one channel); 88kHz (eight channels)

### 6.3.2 Applications

- DC voltage measurement
- Transducer and sensor interfacing
- Vibration and transient analysis
- Process monitoring and control
- Waveform acquisition and analysis
- Multichannel data acquisition
- Real-time simulation
- Programmable voltage output
- Position measurements
- Servo systems
- PWM
- Frequency measurements
- Time measurements
- Pulse/frequency generation
- Pulse counting

### 6.3.3 Specifications