Humusoft Data Acquisition Library

Generated by Doxygen 1.8.14

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 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` 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 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 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 2002-2007 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.

Chapter 2

Feature list of AD612 device.

2.1 Features

- Eight single-ended 12-bit analog input channels, see HudaqAIRead, HudaqAIReadMultiple.
- Programmable A/D ranges, see HudaqGetParameter, HudaqSetParameter
- Four 12-bit analog output channels, see HudaqAOWrite, HudaqAOWriteMultiple.
- 8 digital inputs, see HudaqDIRead, HudaqDIReadMultiple.
- 8 digital outputs, see HudaqDOWrite, HudaqDOWriteMultiple.

- Sampling rate up to 108kHz (one channel); 13kHz (eight channels)

2.2 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

2.3 Specifications

Chapter 3

Feature list of MF614 device.

3.1 Features

- Eight single-ended 12-bit analog input channels, see HudaqAIRead, HudaqAIReadMultiple.
- Programmable A/D input ranges, see HudaqSetParameter, HudaqGetParameter
- Four 12-bit analog output channels, see HudaqAOWrite, HudaqAOWriteMultiple.
- 8 digital inputs aggregated in one channel, see HudaqDIRead, HudaqDIReadMultiple.
- 8 digital outputs aggregated in one channel, see HudaqDOWrite, HudaqDOWriteMultiple.
- Four quadrature encoder inputs (diffrerential), see HudaqEncRead, HudaqEncReset
- Five counters/timers, see HudaqCtrRead, HudaqCtrReset
- Sampling rate up to 108kHz (one channel); 13kHz (eight channels)

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

3.3 Specifications

Chapter 4

Feature list of MF624 device.

4.1 Features

- Eight single-ended 14-bit analog input channels, see HudaqAIRead, HudaqAIReadMultiple.
- Eight 14-bit analog output channels, see HudaqAOWrite, HudaqAOWriteMultiple.
- 8 digital inputs aggregated in one channel, see HudaqDIRead, HudaqDIReadMultiple.
- 8 digital outputs aggregated in one channel, see HudaqDOWrite, HudaqDOWriteMultiple.
- Four quadrature encoder inputs (diffrerential), see HudaqEncRead, HudaqEncReset
- Four counters/timers, see HudaqCtrRead, HudaqCtrReset
- Fast conversion rate; up to 250kHz (one channel); 88kHz (eight channels)

4.2 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

4.3 Specifications

Chapter 5

Feature list of MF634 device.

5.1 Features

- Eight single-ended 14-bit analog input channels, see HudaqAIRead, HudaqAIReadMultiple.
- Eight 14-bit analog output channels, see HudaqAOWrite, HudaqAOWriteMultiple.
- 8 digital inputs aggregated in one channel, see HudaqDIRead, HudaqDIReadMultiple.
- 8 digital outputs aggregated in one channel, see HudaqDOWrite, HudaqDOWriteMultiple.
- Four quadrature encoder inputs (diffrerential), see HudaqEncRead, HudaqEncReset
- Four counters/timers, see HudaqCtrRead, HudaqCtrReset
- Fast conversion rate; up to 250kHz (one channel); 88kHz (eight channels)

5.2 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

5.3 Specifications

Chapter 6

Feature list of MF644 device.

6.1 Features

- Eight single-ended 14-bit analog input channels, see HudaqAIRead, HudaqAIReadMultiple.
- Eight 14-bit analog output channels, see HudaqAOWrite, HudaqAOWriteMultiple.
- 8 digital inputs aggregated in one channel, see HudaqDIRead, HudaqDIReadMultiple.
- 8 digital outputs aggregated in one channel, see HudaqDOWrite, HudaqDOWriteMultiple.
- Four quadrature encoder inputs (diffrerential), see HudaqEncRead, HudaqEncReset
- Four counters/timers, see HudaqCtrRead, HudaqCtrReset
- Fast conversion rate; up to 250kHz (one channel); 88kHz (eight channels)

6.2 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 Specifications

Chapter 7

Module Index

7.1 Modules

Here is a list of all modules:

Chapter 8

Module Documentation

8.1 Board Initialization

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.

Functions

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

  UNDOCUMENTED.
- HUDAQSTATUS HudaqResetDevice (HUDAQHANDLE handle)

  Reset a data acquisition device.
- void HudaqCloseDevice (HUDAQHANDLE handle)

  Close a data acquisition device handle.

8.1.1 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.

8.1.2 Function Documentation

8.1.2.1 HudaqOpenDevice()

```
HUDAQHANDLE HudaqOpenDevice (
              const char ∗ devicename,
              int deviceorder,
              int options )
```

UNDOCUMENTED.

Open a data acquisition device. The device is put into initial state when being opened.

Parameters

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

Returns

Device handle or zero on failure.

Examples:

AIRead.c, AIReadEx.c, AIReadMultiple.c, AOWrite.c, AOWriteMultiple.c, CtrRead.c, DIRead.c, DIReadBit.c, DOWrite.c, DOWriteBit.c, DOWriteMultipleBits.c, DumpBAR.c, EncConfig.c, EncRead.c, IRCRead.c, ListClassic.c, ListDevices.c, ProbeDevices.c, PWM3Write.c, and PWMWrite.c.

8.1.2.2 HudaqResetDevice()

```
HUDAQSTATUS HudaqResetDevice (
            HUDAQHANDLE handle )
```

Reset a data acquisition device.

This function puts the device into initial state.

Parameters

in	*handle*	Device handle.

Returns

HUDAQSUCCESS on success, other values on failure.

8.1.2.3 HudaqCloseDevice()

```
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

in	*handle*	Device handle.

Examples:

AIRead.c, AIReadEx.c, AIReadMultiple.c, AOWrite.c, AOWriteMultiple.c, CtrRead.c, DIRead.c, DIReadBit.c, DOWrite.c, DOWriteBit.c, DOWriteMultipleBits.c, DumpBAR.c, EncConfig.c, EncRead.c, IRCRead.c, List↩ Classic.c, ListDevices.c, ProbeDevices.c, PWM3Write.c, and PWMWrite.c.

8.2 Analog Input

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

Functions

- double HudaqAIRead (HUDAQHANDLE handle, unsigned channel)

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

  Read data from multiple analog input channels.

8.2.1 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.
- MF625 has 8 analog input channels.

8.2.2 Function Documentation

8.2.2.1 HudaqAIRead()

```
double HudaqAIRead (
            HUDAQHANDLE handle,
            unsigned channel )
```

UNDOCUMENTED.

Read data from a single analog input channel.

Parameters

in	*handle*	Device handle.
in	*channel*	Analog input channel number.

Returns

Value read from the analog input channel.

Examples:

AIRead.c, AIReadEx.c, DumpBAR.c, and ProbeDevices.c.

8.2.2.2 HudaqAIReadMultiple()

```
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 this feature.

Parameters

in	*handle*	Device handle.
in	*number*	Number of channels to read.
in	*channels*	Array of channel numbers that will be read. The array must contain the specified number of channel numbers.
out	*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 on success, other values on failure.

Examples:

AIReadMultiple.c.

8.3 Analog Output

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

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.

8.3.1 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.
- MF625 has 8 analog output channels.

8.3.2 Function Documentation

8.3.2.1 HudaqAOWrite()

```
void HudaqAOWrite (
            HUDAQHANDLE handle,
            unsigned channel,
            double value )
```

Write data to a single analog output channel.

Parameters

in	*handle*	Device handle.
in	*channel*	Analog output channel number.
in	*value*	Value to write to the analog output channel.

Examples:

AOWrite.c.

8.3.2.2 HudaqAOWriteMultiple()

```
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

in	*handle*	Device handle.
in	*number*	Number of channels to be written.
in	*channels*	Array of channel numbers that will be written. The array must contain the specified number of channel numbers.
in	*values*	Array of values to be written to the analog output channels. The array must contain the specified number of values.

Returns

HUDAQSUCCESS on success, other values on failure.

Examples:

AOWriteMultiple.c.

8.4 Digital Input

Digital input routines read logical values from digital inputs.

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.

8.4.1 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.
- MF625 has one 8-bit digital input channel.

8.4.2 Function Documentation

8.4.2.1 HudaqDIReadBit()

```
int HudaqDIReadBit (
            HUDAQHANDLE handle,
            unsigned channel,
            unsigned bit )
```

Read a single bit from a digital input channel.

Parameters

in	*handle*	Device handle.
in	*channel*	Digital input channel number.
in	*bit*	Bit position in the channel.

Returns

Bit value read from the specified bit.

Examples:

DIReadBit.c.

8.4.2.2 HudaqDIRead()

```
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

in	*handle*	Device handle.
in	*channel*	Digital input channel number.

Returns

Value read from the digital input channel.

Examples:

DIRead.c, DumpBAR.c, and ProbeDevices.c.

8.4.2.3 HudaqDIReadMultiple()

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

Read data from multiple digital input channels.

Parameters

in	*handle*	Device handle.
in	*number*	Number of channels to read.
in	*channels*	Array of channel numbers that will be read. The array must contain the specified number of channel numbers.
out	*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 on success, other values on failure.

8.5 Digital Output

Digital output routines write logical values to digital outputs.

Functions

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

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

8.5.1 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.
- MF625 has one 8-bit digital output channel.

8.5.2 Function Documentation

8.5.2.1 HudaqDOWriteBit()

```
void HudaqDOWriteBit (
             HUDAQHANDLE handle,
             unsigned channel,
             unsigned bit,
             int value )
```

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

Parameters

in	*handle*	Device handle.
in	*channel*	Digital output channel number.
in	*bit*	Bit position in channel specified
in	*value*	Bit value to be written to the specified bit.

Examples:

DOWriteBit.c.

8.5.2.2 HudaqDOWrite()

```
HUDAQSTATUS 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

in	*handle*	Device handle.
in	*channel*	Digital output channel number.
in	*value*	Value to be written into digital output.

Returns

HUDAQSUCCESS on success, other values on failure.

Examples:

DOWrite.c.

8.5.2.3 HudaqDOWriteMultipleBits()

```
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

in	*handle*	Device handle.
in	*channel*	Digital output channel number.
in	*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.
in	*value*	Value to write. Only the bits specified by `mask` are modified, the other bits are ignored.

Examples:

DOWriteMultipleBits.c.

8.5.2.4 HudaqDOWriteMultiple()

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

Write data to multiple digital output channels.

Parameters

in	*handle*	Device handle.
in	*number*	Number of channels to be written.
in	*channels*	Array of channel numbers that will be written. The array must contain the specified number of channel numbers.
in	*values*	Array of values to be written to the digital output channels. The array must contain the specified number of values.

Returns

HUDAQSUCCESS on success, other values on failure.

8.6 Counter Input

Counter input routines read the counter pulse count.

Functions

- void HudaqCtrReset (HUDAQHANDLE handle, unsigned channel)

  Reset counter pulse count.
- int HudaqCtrRead (HUDAQHANDLE handle, unsigned channel)

  Read counter pulse count.

8.6.1 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) is selected as the default counter input. The input of individual counter channels can then be changed by a call to HudaqSetParameter. 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 counter input channel.
- MF614 has 4 counter input channels, the hardware is shared with PWM output channels.
- AD622 has no counter input channel.
- MF624 has 4 counter input channels, the hardware is shared with PWM output channels.
- MF625 has no counter input channels.

8.6.2 Function Documentation

8.6.2.1 HudaqCtrReset()

```
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

in	*handle*	Device handle.
in	*channel*	Number of counter channel.

Examples:

CtrRead.c.

8.6.2.2 HudaqCtrRead()

```
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

in	*handle*	Device handle.
in	*channel*	Counter input channel number.

Returns

Value read from the counter input channel.

Examples:

CtrRead.c, DumpBAR.c, and ProbeDevices.c.

8.7 Encoder Input

Encoder input routines read the encoder pulse count.

Functions

- void HudaqEncReset (HUDAQHANDLE handle, unsigned channel)

    Reset encoder pulse count.
- int HudaqEncRead (HUDAQHANDLE handle, unsigned channel)

    Read encoder pulse count.

8.7.1 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) the *A* and *B* inputs expect signal from quadrature encoder pulse ouputs 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.

The *I* input works in any of the encoder modes and its functionality is programmable - see HudaqEncReset↩ Mode 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.
- MF625 has 4 encoder input channels.

8.7.2 Function Documentation

8.7.2.1 HudaqEncReset()

```
void HudaqEncReset (
             HUDAQHANDLE handle,
             unsigned channel )
```

Reset encoder pulse count.

Parameters

in	*handle*	Device handle.
in	*channel*	Encoder input channel number.

Examples:

IRCRead.c.

8.7.2.2 HudaqEncRead()

```
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

in	*handle*	Device handle.
in	*channel*	Encoder input channel number.

Returns

Value read from the encoder input channel.

Examples:

DumpBAR.c, EncConfig.c, EncRead.c, IRCRead.c, and ProbeDevices.c.

8.8 PWM Output

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

Functions

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

  Generate pulse-width modulation signal on a counter output.
- HUDAQSTATUS HudaqPWM3Write (HUDAQHANDLE handle, unsigned channel, double frequency, double duty1, double duty2, double duty3)

  Generate 3 phases + their inversions pulse-width modulation signal on a specialized counter.

8.8.1 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.
- MF625 has 1 three phases + inversions (6 phases) specialised PWM output channel. Maximum output frequency is 12.5MHz.

8.8.2 Function Documentation

8.8.2.1 HudaqPWMWrite()

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

Generate pulse-width modulation signal on a counter output.

Parameters

in	*handle*	Device handle.
in	*channel*	Counter output channel number.
in	*frequency*	Output frequency in Hz.
in	*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.

Returns

HUDAQSUCCESS on success, other values on failure.

Examples:

PWMWrite.c.

8.8.2.2 HudaqPWM3Write()

```
HUDAQSTATUS HudaqPWM3Write (
            HUDAQHANDLE handle,
            unsigned channel,
            double frequency,
            double duty1,
            double duty2,
            double duty3 )
```

Generate 3 phases + their inversions pulse-width modulation signal on a specialized counter.

Parameters

in	*handle*	Device handle.
in	*channel*	Counter output channel number.
in	*frequency*	Output frequency in Hz.
in	*duty1*	Output signal duty for first phase.
in	*duty2*	Output signal duty for second phase.
in	*duty3*	Output signal duty for third phase. 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 on success, other values on failure.

Examples:

PWM3Write.c.

8.9 Channel Configuration

Some channels can perform different functions based on their parameters.

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,
  HudaqCtrOUTPUTCONTROL,
  HudaqCtrREPETITION,
  HudaqCtrLOADTOGGLE,
  HudaqCtrDIRECTION,
  HudaqCtrOUTTOGGLE,
  HudaqCtrTRIGSOURCE,
  HudaqCtrTRIGTYPE,
  HudaqCtrRETRIGGER,
  HudaqCtrGATESOURCE,
  HudaqCtrGATEPOLARITY,
  HudaqCtrFILTER,
  HudaqPwmPHASES,
  HudaqPwmUPDOWN,
  HudaqPwmINVERSIONS,
  HudaqPwmDEADBAND,
  HudaqPwmOUTPUTCONTROL,
  HudaqPwmCLOCKSOURCE,
  HudaqPwmFILTER,
  HudaqPwmGATESOURCE,
  HudaqPwmTRANSPARENT,
  HudaqPwmEMERGENCY,
  HudaqPwmGATEPOLARITY,
  HudaqPwmOUTPUTUDCONTROL }

  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,
HudaqCtrCLOCK20MHz,
HudaqCtrCLOCK2MHz,
HudaqCtrCLOCK200kHz,
HudaqCtrCLOCK20kHz,
HudaqCtrCLOCK2kHz }

Counter clock sources.

- enum HudaqCtrOutputControl {
HudaqCtrOUTPUTNORMAL = 0,
HudaqCtrOUTPUTINVERTED = 1,
HudaqCtrOUTPUT_0 = 2,
HudaqCtrOUTPUT_1 = 3 }

  Counter output control.

- enum HudaqCtrTrigSource {
HudaqCtrTRIGDISABLE = 0,
HudaqCtrTRIGINPUT = 1,
HudaqCtrTRIGPREV = 2,
HudaqCtrTRIGNEXT = 3 }

  Counter trigger source.

- enum HudaqCtrTrigType {
HudaqCtrTRIGNONE = 0,
HudaqCtrTRIGRISING = 1,
HudaqCtrTRIGFALLING = 2,
HudaqCtrTRIGEITHER = 3 }

  Counter trigger type.

- enum HudaqCtrGateSource {
HudaqCtrGATEHIGH = 0,
HudaqCtrGATEINPUT = 1,
HudaqCtrGATEPREV = 2,
HudaqCtrGATENEXT = 3 }

  Counter gate source.

- 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)

  UNDOCUMENTED.
- 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.

8.9.1 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.

8.9.2 Enumeration Type Documentation

8.9.2.1 HudaqSubsystem

```
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.

8.9.2.2 HudaqParameter

```
enum HudaqParameter
```

Parameter identifiers.

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

Enumerator

HudaqAIRANGE	UNDOCUMENTED. 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.
HudaqAORANGE	UNDOCUMENTED. UNDOCUMENTED 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.
HudaqEncRESETONREAD	UNDOCUMENTED. UNDOCUMENTED UNDOCUMENTED 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.
HudaqEncCOUNTCONTROL	Encoder count control; for possible values see HudaqEncCountControl.
HudaqEncRESETMODE	Encoder reset mode; for possible values see HudaqEncResetMode.
HudaqCtrRESETONREAD	UNDOCUMENTED. 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.
HudaqCtrOUTPUTCONTROL	Counter output signal; for possible values see HudaqCtrOutputControl.
HudaqCtrREPETITION	0 - counter stops after terminal count; 1 - counter reloads and continues counting.
HudaqCtrLOADTOGGLE	0 - always load from register A; 1 - alternate load registers A and B.
HudaqCtrDIRECTION	0 - counter counts down; 1 - counter counts up.
HudaqCtrOUTTOGGLE	0 - output is directly connected to TC; 1 - use flipflop that is toggled on every TC.
HudaqCtrTRIGSOURCE	Counter trigger source; for possible values see HudaqCtrTrigSource.
HudaqCtrTRIGTYPE	Counter trigger edge; for possible values see HudaqCtrTrigType.
HudaqCtrRETRIGGER	Counter can be retriggered: 0 - only when stopped; 1 - anytime.
HudaqCtrGATESOURCE	Counter gate source; for possible values see HudaqCtrGateSource.
HudaqCtrGATEPOLARITY	Counter gate polarity: 0 - gate low disables counting; 1 - gate high disables counting.
HudaqCtrFILTER	Filter counter inputs with a lowpass filter; possible values are 0 (off) or 1 (on).

Enumerator

HudaqPwmPHASES	UNDOCUMENTED. UNDOCUMENTED Read bits that coresponds to output phases. Only HudaqGetParameter is supported. (MF625 only)
HudaqPwmUPDOWN	Read UpDown flag. Only HudaqGetParameter is supported. (MF625 only)
HudaqPwmINVERSIONS	Read bits, that corresponds to inversions in all phases. (MF625 only)
HudaqPwmDEADBAND	Set dead band between phase and inverted phase (MF625 only)
HudaqPwmOUTPUTCONTROL	Output phase signal; for possible values see HudaqCtrOutputControl.
HudaqPwmCLOCKSOURCE	Counter clock source; for possible values see HudaqCtrClockSource.
HudaqPwmFILTER	Filter counter inputs with a lowpass filter; possible values are 0 (off) or 1 (on).
HudaqPwmGATESOURCE	Counter gate source; for possible values see HudaqCtrGateSource.
HudaqPwmTRANSPARENT	0 - Dual buffer is turned on, 1 - Dual buffer is turned off. (MF625 only)
HudaqPwmEMERGENCY	Define condition to block PWM output; for possible values see ::HudaqPwmEmergency (MF625 only)
HudaqPwmGATEPOLARITY	Counter gate polarity: 0 - gate low disables counting; 1 - gate high disables counting.
HudaqPwmOUTPUTUDCONTROL	Output up/down signal; for possible values see HudaqCtrOutputControl.

8.9.2.3 HudaqCtrClockSource

```
enum HudaqCtrClockSource
```

Counter clock sources.

Enumerator

HudaqCtrCLOCK50MHz	Internal clock 50MHz. (MF624 and MF625)
HudaqCtrCLOCK10MHz	Internal clock 10MHz. (MF624 and MF625)
HudaqCtrCLOCK1MHz	Internal clock 1MHz. (MF624 and MF625)
HudaqCtrCLOCK100kHz	Internal clock 100kHz. (MF624 and MF625)
HudaqCtrCLOCKINRISING	External input, count on rising edge. (MF614, MF624 and MF625)
HudaqCtrCLOCKINFALLING	External input, count on falling edge.(MF614, MF624 and MF625)
HudaqCtrCLOCKINEITHER	External input, count on either edge. (MF624 and MF625)
HudaqCtrCLOCKPREVRISING	Output of previous counter, count on rising edge. (MF614, MF624 and MF625)
HudaqCtrCLOCKPREVFALLING	Output of previous counter, count on falling edge. (MF614, MF624 and MF625)
HudaqCtrCLOCKPREVEITHER	Output of previous counter, count on either edge. (MF624 and MF625)
HudaqCtrCLOCKNEXTRISING	Output of next counter, count on rising edge. (MF624 and MF625)
HudaqCtrCLOCKNEXTFALLING	Output of next counter, count on falling edge. (MF624 and MF625)
HudaqCtrCLOCKNEXTEITHER	Output of next counter, count on either edge. (MF624 and MF625)
HudaqCtrCLOCK20MHz	Internal clock 20MHz. (MF614 only)
HudaqCtrCLOCK2MHz	Internal clock 2MHz. (MF614 only)

Enumerator

HudaqCtrCLOCK200kHz	Internal clock 200kHz. (MF614 only)
HudaqCtrCLOCK20kHz	Internal clock 20kHz. (MF614 only)
HudaqCtrCLOCK2kHz	Internal clock 2kHz. (MF614 only)

8.9.2.4 HudaqCtrOutputControl

```
enum HudaqCtrOutputControl
```

Counter output control.

Enumerator

HudaqCtrOUTPUTNORMAL	Normal counter output. (MF614 and MF624)
HudaqCtrOUTPUTINVERTED	Inverted counter output. (MF624 only)
HudaqCtrOUTPUT_0	Force 0 on output, counter is still counting. (MF624 only)
HudaqCtrOUTPUT_1	Force 1 on output,counter is still counting. (MF624 only)

8.9.2.5 HudaqCtrTrigSource

```
enum HudaqCtrTrigSource
```

Counter trigger source.

Enumerator

HudaqCtrTRIGDISABLE	Trigger is disabled.
HudaqCtrTRIGINPUT	Trigger by counter input (TxIn).
HudaqCtrTRIGPREV	Trigger by previous counter output.
HudaqCtrTRIGNEXT	Trigger by next counter output.

8.9.2.6 HudaqCtrTrigType

```
enum HudaqCtrTrigType
```

Counter trigger type.

Enumerator

HudaqCtrTRIGNONE	Trigger is inactive.

Enumerator

HudaqCtrTRIGRISING	Trigger by rising edge of trigger signal.
HudaqCtrTRIGFALLING	Trigger by falling edge of trigger signal.
HudaqCtrTRIGEITHER	Trigger by either edge of trigger signal.

8.9.2.7 HudaqCtrGateSource

```
enum HudaqCtrGateSource
```

Counter gate source.

Please note HudaqCtrGATEPOLARITY for full undrestanding gate functionality.

Enumerator

HudaqCtrGATEHIGH	Gate is forced to logical high. (MF614, MF624 and MF625)
HudaqCtrGATEINPUT	Gate by counter input (TxIn). (MF614, MF624 and MF625)
HudaqCtrGATEPREV	Gate by previous counter output. (MF624 and MF625)
HudaqCtrGATENEXT	Gate by next counter output. (MF624 and MF625)

8.9.2.8 HudaqEncMode

```
enum HudaqEncMode
```

Encoder counting modes.

Enumerator

HudaqEncMODEIRC	Count IRC pulses on inputs *A* and *B* (MF614, MF624 and MF625)
HudaqEncMODERISING	Count pulses on *A*, rising edge, *B* specifies direction (MF614, MF624 and MF625)
HudaqEncMODEFALLING	Count pulses on *A*, falling edge, *B* specifies direction (MF624 and MF625)
HudaqEncMODEEITHER	Count pulses on *A*, either edge, *B* specifies direction (MF624 and MF625)

8.9.2.9 HudaqEncCountControl

```
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, MF624 and MF625)
HudaqEncCOUNTDISABLE	Encoder counting is disabled. (MF624 and MF625)
HudaqEncCOUNTI0	Encoder counting is enabled when input *I* is at logical low, disabled when input *I* is at logical high. (MF624 and MF625)
HudaqEncCOUNTI1	Encoder counting is enabled when input *I* is at logical high, disabled when input *I* is at logical low. (MF624 and MF625)

8.9.2.10 HudaqEncResetMode

```
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, MF624 and MF625)
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, MF624 and MF625)
HudaqEncRESI1	Reset encoder pulse count when input *I* is at logical high. (MF624 and MF625)
HudaqEncRESIRISING	Reset encoder pulse count on rising edge of input *I* . (MF614, MF624 and MF625)
HudaqEncRESIFALLING	Reset encoder pulse count on falling edge of input *I* . (MF614, MF624 and MF625)
HudaqEncRESIEITHER	Reset encoder pulse count on either edge of input *I* . (MF624 and MF625)

8.9.3 Function Documentation

8.9.3.1 HudaqGetParameter()

```
double HudaqGetParameter (
            HUDAQHANDLE handle,
            unsigned channel,
            HudaqParameter param )
```

UNDOCUMENTED.

Reads single channel configuration for a given subsystem. Not all devices support all the parameters.

Parameters

in	*handle*	Device handle.
in	*channel*	Channel number. The channel type is determined by the parameter identifier `param`.
in	*param*	Parameter identifier; see HudaqParameter for possible values.

Returns

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

Examples:

AIReadEx.c, DumpBAR.c, and ProbeDevices.c.

8.9.3.2 HudaqSetParameter()

```
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

in	*handle*	Device handle.
in	*channel*	Channel number. The channel type is determined by the parameter identifier `param`.
in	*param*	Parameter identifier; see HudaqParameter for possible values.
in	*value*	Value of the parameter; see individual parameter descriptions for possible values.

Returns

HUDAQSUCCESS on success, other values on failure.

Examples:

AIReadEx.c, EncConfig.c, and PWM3Write.c.

8.9.3.3 HudaqQueryRange()

```
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 to set a voltage range by its index; use HudaqGetParameter to get the index of the currently set voltage range.

Parameters

in	*handle*	Device handle.
in	*S*	Subsystem identifier; possible values are HudaqAI or HudaqAO.
in	*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;
```

8.10 General Purpose Constants and Data Types

UNDOCUMENTED.

Typedefs

- typedef size_t HUDAQHANDLE

  UNDOCUMENTED.

Enumerations

- enum HUDAQSTATUS {
  HUDAQSUCCESS = 0,
  HUDAQIrqPending = 5 }

  Return codes for HUDAQ functions.

8.10.1 Detailed Description

UNDOCUMENTED.

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

8.10.2 Typedef Documentation

8.10.2.1 HUDAQHANDLE

```
typedef size_t HUDAQHANDLE
```

UNDOCUMENTED.

UNDOCUMENTED The HUDAQ device handle data type.

8.10.3 Enumeration Type Documentation

8.10.3.1 HUDAQSTATUS

```
enum HUDAQSTATUS
```

Return codes for HUDAQ functions.

Enumerator

HUDAQSUCCESS	Success. All other values mean failure.
HUDAQIrqPending	UNDOCUMENTED.

Chapter 9

Example Documentation

9.1 AIRead.c

```
/* Humusoft data acquisition library.
 * Example that shows reading of analog input channels
 * using the function to read a single channel.
 */

/* Copyright 2002-2006 Humusoft s.r.o. */

#include <stdio.h>

#include "hudaqlib.h"

#define DAQ_DEVICE      "MF634"

int main(int argc, char* argv[])
{
  HUDAQHANDLE h;
  unsigned i;
  double value;

  /* open a handle to the first MF624 device in the system */
  h = HudaqOpenDevice(DAQ_DEVICE, 1, 0);
  if (h==0)
  {
    printf("\nData acquisition device %s not found.\n",DAQ_DEVICE);
    return(-1);
  }

  /* read all the 8 analog inputs in a loop, print their values */
  for (i=0; i<8; i++)
  {
    value = HudaqAIRead(h,i);
    printf("Analog channel %u, value read %fV.\n", i, value);
  }

  /* close the device handle */
  HudaqCloseDevice(h);

  return(0);
}
```

9.2 AIReadEx.c

```
/* AIReadEx.c:
 *   This demo demonstrates how to read analog inputs. It shows how to get
 *   a handle to Hudaq device, how to read data analog inputs. */

#include <windows.h>
#include <stdio.h>
#include <conio.h>

#include "../hudaqlib.h"
```

```
void cls(void);

int main(int argc, char* argv[])
{
HUDAQHANDLE h;
unsigned i;
double ValueRead;
int NoAnalogIn;

double range=10;

  h = HudaqOpenDevice("MF634", 1, 0);  /* Get first MF634 device found. */
  if(h==0)
    {
    printf("\nNo Hudaq device found!\n");
    return -1;                         /* No Hudaq device - return from application */
    }

  i=1;
  NoAnalogIn = HudaqGetParameter(h,0,HudaqAINUMCHANNELS);
  //HudaqAISetParameter(h,0,HudaqAIBipolar,&i);
  //HudaqAISetParameter(h,1,HudaqAIVolts,&range);
  HudaqSetParameter(h,0,HudaqAIRange,0);

  while(!kbhit())
    {
    cls();
    for(i=0; i<NoAnalogIn; i++)
      {
      ValueRead = HudaqAIRead(h,i);      /* Read from analog input. */
      printf("Analog channel %d, value read %fV\n", i, ValueRead);
      }
    Sleep(10);
    }

 HudaqCloseDevice(h);                  /* Closing a Hudaq device. */

return 0;
}


  /* Auxiliarry function that clears a Windows console screen. */
void cls(void)
{
HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
   COORD coordScreen = { 0, 0 };    /* home for the cursor */
   DWORD cCharsWritten;
   CONSOLE_SCREEN_BUFFER_INFO csbi;
   DWORD dwConSize;
        /* Get the number of character cells in the current buffer. */
   if( !GetConsoleScreenBufferInfo( hConsole, &csbi )) return;
   dwConSize = csbi.dwSize.X * csbi.dwSize.Y;
        /* Fill the entire screen with blanks. */
   if( !FillConsoleOutputCharacter( hConsole, (TCHAR) ' ',dwConSize, coordScreen, &cCharsWritten )) return;
        /* Get the current text attribute. */
   if( !GetConsoleScreenBufferInfo( hConsole, &csbi )) return;
        /* Set the buffer's attributes accordingly. */
   if( !FillConsoleOutputAttribute( hConsole, csbi.wAttributes,dwConSize, coordScreen, &cCharsWritten ))
      return;
        /* Put the cursor at its home coordinates. */
   SetConsoleCursorPosition( hConsole, coordScreen );
}
```

9.3 AIReadMultiple.c

```
/* Humusoft data acquisition library.
 *
 * Example that shows reading of analog input channels
 * using the function to read multiple channels together.
 */

/* Copyright 2002-2006 Humusoft s.r.o. */

#include <stdio.h>

#include "hudaqlib.h"


#define DAQ_DEVICE      "MF634"
```

```
int main(int argc, char* argv[])
{
  HUDAQHANDLE h;
  unsigned i;
        /* Buffer for channel numbers. Order of channels is not significant.
           Duplicated channels are also supported. */
  unsigned channels[8] = {4,5,6,7,0,1,2,3};
        /* Buffer for receiving values read. Is size must correspond to
           buffer of channels. */
  double values[8];

  /* open a handle to the first MF624 device in the system */
  h = HudaqOpenDevice(DAQ_DEVICE, 1, 0);
  if (h==0)
  {
    printf("\nData acquisition device %s not found.\n",DAQ_DEVICE);
    return(-1);
  }

  /* read all the 8 analog inputs together */
  HudaqAIReadMultiple(h,8,channels,values);

  /* print values read */
  for (i=0; i<8; i++)
  {
    printf("Analog channel %u, value read %fV.\n", channels[i], values[i]);
  }

  /* close the device handle */
  HudaqCloseDevice(h);

  return(0);
}
```

9.4 AOWrite.c

```
/* Humusoft data acquisition library.
 * Example that shows writing to analog output channels
 * using the function to write a single channel.
 */

/* Copyright 2002-2007 Humusoft s.r.o. */

#include <stdio.h>

#include "hudaqlib.h"


int main(int argc, char* argv[])
{
  HUDAQHANDLE h;
  unsigned i;
  double value;

  /* open a handle to the first MF624 device in the system */
  h = HudaqOpenDevice("MF624", 1, 0);
  if (h==0)
  {
    printf("\nData acquisition device not found.\n");
    return(-1);
  }

  /* write all the 8 analog outputs in a loop */
  /* the voltage written to the output is computed as (channel number - 5) */
  for (i=0; i<8; i++)
  {
    value = i-5.0;
    HudaqAOWrite(h, i, value);
    printf("Analog output channel %u, value written %fV.\n", i, value);
  }

  /* close the device handle */
  HudaqCloseDevice(h);

  return(0);
}
```

9.5 AOWriteMultiple.c

```
/* Humusoft data acquisition library.
 *
 * Example that shows writing to analog output channels
 * using the function to write multiple channels together.
 */

/* Copyright 2002-2006 Humusoft s.r.o. */

#include <stdio.h>

#include "hudaqlib.h"


int main(int argc, char* argv[])
{
  HUDAQHANDLE h;
        /* Buffer for channel numbers. Order of channels is not significant.
           Duplicated channels are also supported. */
  unsigned channels[8] = {4,5,6,7,0,1,2,3};
        /* Buffer that contains values to be written.
           Is size must correspond to buffer of channels. */
  double values[8] = {5.0, 6.0, 7.0, 8.0, 1.0, 2.0, 3.0, 4.0};

  /* Open a handle to the first MF624 device in the system. */
  h = HudaqOpenDevice("MF624", 1, 0);
  if (h==0)
  {
    printf("\nData acquisition device not found.\n");
    return(-1);
  }

  /* Write all the 8 analog outputs in one call. */
  if(HudaqAOWriteMultiple(h, 8, channels, values)==
      HUDAQSUCCESS)
  {
    printf("\nData has been written.\n");
  }

  /* Close the device handle. */
  HudaqCloseDevice(h);

  return(0);
}
```

9.6 CtrRead.c

```
/* Humusoft data acquisition library.
 *
 * Example that shows reading of counters and counting pulses.
 */

/* Copyright 2002-2006 Humusoft s.r.o. */

#include <stdio.h>

#include "hudaqlib.h"


int main(int argc, char* argv[])
{
  HUDAQHANDLE h;
  unsigned i;
  int value;

  /* open a handle to the first MF624 device in the system */
  h = HudaqOpenDevice("MF624", 1, 0);
  if (h==0)
  {
    printf("\nData acquisition device not found.\n");
    return(-1);
  }

  /* Do reset of counters. Each counter is switched to counting mode
     after its first usage. */
  for (i=0; i<4; i++)
  {
    HudaqCtrReset(h,i);
  }

  printf("Counting external pulses by counters, press Enter to continue.\n");
  getchar();

  /* read all the 4 counters in a loop, print their values */
  for (i=0; i<4; i++)
  {
    value = HudaqCtrRead(h,i);
    printf("Counter channel %u, value read %d.\n", i, value);
  }

  /* close the device handle */
  HudaqCloseDevice(h);

  return(0);
}
```

9.7 DIRead.c

```
/* Humusoft data acquisition library.
 * Example that shows reading of digital input channels
 * using the function to read a single channel.
 */

/* Copyright 2002-2007 Humusoft s.r.o. */

#include <stdio.h>

#include "hudaqlib.h"


int main(int argc, char* argv[])
{
  HUDAQHANDLE h;
  unsigned value;

  /* open a handle to the first MF624 device in the system */
  h = HudaqOpenDevice("PCT7303B", 1, 0);
//  h = HudaqOpenDevice("PCD7004", 1, 0);
  if (h==0)
  {
    printf("\nData acquisition device not found.\n");
    return(-1);
  }

  /* read whole digital channel at once */
  value = HudaqDIRead(h,0);
  printf("\nValue read from digital channel 0: %Xh ", value);

  /* close the device handle */
  HudaqCloseDevice(h);

  return(0);
}
```

9.8 DIReadBit.c

```
/* Humusoft data acquisition library.
 * Example that shows reading of digital input channels using
 * the function to read separate bits from a single channel.
 */

/* Copyright 2002-2007 Humusoft s.r.o. */

#include <stdio.h>

#include "hudaqlib.h"


int main(int argc, char* argv[])
{
  HUDAQHANDLE h;
  unsigned i;
  double value;

  /* open a handle to the first MF624 device in the system */
  h = HudaqOpenDevice("MF624", 1, 0);
  if (h==0)
  {
    printf("\nData acquisition device not found.\n");
    return(-1);
  }

  /* read all 8 bits from digital inputs in a loop, print their values */
  for(i=0; i<8; i++)
  {
    value = HudaqDIReadBit(h,0,i);  /* Read one bit from digital input */
    printf("bit:%u, %.0f ", i, value);
  }
  printf("\n");

  /* close the device handle */
  HudaqCloseDevice(h);

  return(0);
}
```

9.9 DOWrite.c

```
/* Humusoft data acquisition library.
 * Example that shows writing all bits to a digital output channels
 * using the function to write a single channel.
 */

/* Copyright 2002-2007 Humusoft s.r.o. */

#include <stdio.h>

#include "hudaqlib.h"


int main(int argc, char* argv[])
{
  HUDAQHANDLE h;

  /* open a handle to the first MF624 device in the system */
  h = HudaqOpenDevice("MF624", 1, 0);
  if (h==0)
  {
    printf("\nData acquisition device not found.\n");
    return(-1);
  }

  /* write 0xFF to whole digital channel at once */
  HudaqDOWrite(h,0,0xFF);

  printf("\n0xFF was writen to digital output. Press any key to continue.");
  getchar();

  /* write 0x00 to whole digital channel at once */
  HudaqDOWrite(h,0,0x0);
  printf("\n0x00 has been writen to digital output.");

  /* close the device handle */
  HudaqCloseDevice(h);

  return(0);
}
```

9.10 DOWriteBit.c

```
/* Humusoft data acquisition library.
 *
 * Example that shows writing separate bits to a digital output channel
 * using the function to write a single bit.
 */
```

```
/* Copyright 2002-2006 Humusoft s.r.o. */

#include <stdio.h>

#include "hudaqlib.h"


int main(int argc, char* argv[])
{
  HUDAQHANDLE h;
  unsigned i;

  /* open a handle to the first MF624 device in the system */
  h = HudaqOpenDevice("MF624", 1, 0);
  if (h==0)
  {
    printf("\nData acquisition device not found.\n");
    return(-1);
  }

  /* HudaqOpenDevice initialises all digital output bits to 0 */
  for(i=0; i<8; i++)
  {
    printf("\nPress any key to set a bit %u to '1'", i);
    getchar();
    HudaqDOWriteBit(h, 0, i, 1);
  }

  /* close the device handle */
  HudaqCloseDevice(h);

  return(0);
}
```

9.11 DOWriteMultipleBits.c

```
/* Humusoft data acquisition library.
 *
 * Example that demonstrates using HudaqDOWriteMultipleBits.
 * This function allows to influence only selected bits from
 * digital outputs.
 */

/* Copyright 2002-2007 Humusoft s.r.o. */

#include <stdio.h>

#include "hudaqlib.h"


int main(int argc, char* argv[])
{
  HUDAQHANDLE h;

  /* open a handle to the first MF624 device in the system */
  h = HudaqOpenDevice("MF624", 1, 0);
  if (h==0)
  {
    printf("\nData acquisition device not found.\n");
    return(-1);
  }

  /* HudaqOpenDevice initializes all digital output bits to 0 */

  /* Set first and fifth bits to value '1'. */
  HudaqDOWriteMultipleBits(h, 0, 0x11, 0x11);
  printf("\nBits 1 and 5 are set. Press any key to continue.");
  getchar();

  /* Reset bit 1 and set bit 6. */
  HudaqDOWriteMultipleBits(h, 0, 0x21, 0x20);
  printf("\nBit 1 is reset and bit 6 is set. Press any key to continue.");
  getchar();


  /* close the device handle */
  HudaqCloseDevice(h);

  return(0);
}
```

9.12 DumpBAR.c

```
/* ProbeDevices.c:
 *   This demo demonstrates extract information from all available devices.
 *   (c)2007-2020 Jaroslav Fojtik
 *
 *   This WILL NOT work with original Humusofts library!
 *    HudaqOpenDevice("",) with argument "" is not supported with original hudaqlib.
 */
#include <stdio.h>
#include "hudaqlib.h"


static __inline __int32 GetDword(size_t Ptr, int Offset)
{
  return ((volatile unsigned __int32 *)Ptr)[Offset/4];
}


int main(void)
{
HUDAQHANDLE h;
const HudaqResourceInfo *HRI;
int i,j;
double value;
int NoAnalogIn,NoDigitalIn,NoEncoders,NoCounters;
int dev = 1;
#ifdef _MSC_VER
unsigned __int64 DMA_MemRaw;
void *Pointer;
size_t BlockSz;
#endif
        /* Open first device found of any name. */
  h = HudaqOpenDevice("",1,0);
  if(h==0)
    {printf("No HUDAQ device found\n"); return -1;}

  while(h!=0)
  {
    printf("\n================ DEVICE FOUND ======================");

    HRI = HudaqGetDeviceResources(h);
    printf("\nBus number %d, Slot number %d.",HRI->BusNumber, HRI->SlotNumber);
    printf("\nVendorID %Xh, DeviceID %Xh.",HRI->VendorID,HRI->DeviceID);

    for(i=0; i<HRI->NumMemResources; i++)
    {
      printf("\n Memory resource %d: Base:%Xh, Length:%Xh",
          i, HRI->MemResources[i].Base, HRI->MemResources[i].Length);

      if(HRI->MemResources[i].Base != 0)
      {
        printf("\n");
        for(j=0; j<HRI->MemResources[i].Length/4; j++)
        {
          printf("%4X ",GetDword(HRI->MemResources[i].Base,j));
          if(j>16) break;
        }
      }
    }

    for(i=0; i<HRI->NumIOResources; i++)
    {
      printf("\n IO resource %d: Base:%Xh, Length:%Xh",
        i, HRI->IOResources[i].Base, HRI->IOResources[i].Length);
    }

    NoAnalogIn = HudaqGetParameter(h,0,HudaqAINUMCHANNELS);
    printf("\nAnalog channels AI:%d / AO:%d", NoAnalogIn, (int)HudaqGetParameter(h,0,
      HudaqAONUMCHANNELS));
    for (i=0; i<NoAnalogIn; i++)
    {
      value = HudaqAIRead(h,i);
      printf("\n  Analog channel %d, value read %fV.", i, value);
    }

    NoDigitalIn = HudaqGetParameter(h,0,HudaqDINUMCHANNELS);
    printf("\nDigital channels DI:%d / DO:%d", NoDigitalIn, (int)
      HudaqGetParameter(h,0,HudaqDONUMCHANNELS));
    for (i=0; i<NoDigitalIn; i++)
    {
      printf("\n  Digital input %d: %d",i,HudaqDIRead(h,i));
    }

    NoEncoders = HudaqGetParameter(h,0,HudaqEncNUMCHANNELS);
    printf("\nEncoder channels %d", NoEncoders);
    for (i=0; i<NoEncoders; i++)
    {
      printf("\n  Encoder value %d: %d",i,HudaqEncRead(h,i));
    }

    NoCounters = HudaqGetParameter(h,0,HudaqCtrNUMCHANNELS);
    printf("\nCounter channels %d", NoCounters);
    for (i=0; i<NoCounters; i++)
    {
      printf("\n  Counted value %d: %d",i,HudaqCtrRead(h,i));
    }

    printf("\nPWM channels %d", (int)HudaqGetParameter(h, 0, HudaqPwmNUMCHANNELS));

#ifdef _MSC_VER
    printf("\nIRQ counter: %g (%g)",
             HudaqGetParameter(h,0,HudaqIRQ), HudaqGetParameter(h, 0,
      HudaqIRQ+1));

    HudaqGetDMAinfo(h,&DMA_MemRaw,&Pointer,&BlockSz);
    printf("\nDMA mem raw %lX, pointer %p, size: %u.", DMA_MemRaw,Pointer,(unsigned)BlockSz);
#endif

    HudaqCloseDevice(h);
    h = HudaqOpenDevice("",++dev,0);
  }

  //  HudaqSetParameter(h, 0, HudaqIRQ, 1);
  //i = HudaqGetParameter(h, 0, HudaqIRQ);
  printf("\n");
return 0;
}
```

9.13 EncConfig.c

```
/* Humusoft data acquisition library.
 *
 * Example that shows configuration of
 * encoder.
 */

/* Copyright 2002-2007 Humusoft s.r.o. */

#include <stdio.h>

#include "hudaqlib.h"


int main(int argc, char* argv[])
{
  HUDAQHANDLE h;
  int value;

  /* open a handle to the first MF624 device in the system */
  h = HudaqOpenDevice("MF624", 1, 0);
  if (h==0)
  {
    printf("\nData acquisition device not found.\n");
    return(-1);
  }

  /* Configure encoder to count input pulses. */
  if(HudaqSetParameter(h, 0, HudaqEncMODE,
      HudaqEncMODERISING) != HUDAQSUCCESS)
  {
    printf("\nCannot switch encoder to counting mode.\n");
    HudaqCloseDevice(h);
    return(-2);
  }

  /* Turn on hardware filter for input signal. */
  if(HudaqSetParameter(h, 0, HudaqEncFILTER, 1) !=
      HUDAQSUCCESS)
  {
    printf("\nCannot filter input signal.\n");
  }

  printf("Counting external pulses on input A by encoder, press Enter to continue.\n");
  getchar();

  /* Read encoder 0 value, print it. */
  value = HudaqEncRead(h,0);
  printf("Encoder channel 0, value read %d.\n", value);

  /* close the device handle */
  HudaqCloseDevice(h);

  return(0);
}
```

9.14 EncRead.c

```
/* Humusoft data acquisition library.
 *
 * Example that shows decoding IRC position
 * using the function to read a single encoder.
 */

/* Copyright 2002-2006 Humusoft s.r.o. */

#include <stdio.h>

#include "hudaqlib.h"


int main(int argc, char* argv[])
{
  HUDAQHANDLE h;
  unsigned i;
  int value;

  /* open a handle to the first MF624 device in the system */
  h = HudaqOpenDevice("MF624", 1, 0);
  if (h==0)
  {
    printf("\nData acquisition device not found.\n");
    return(-1);
  }

  printf("Counting external IRC pulses by encoders, press Enter to continue.\n");
  getchar();

  /* Read all the 4 encoder values in a loop, print them. */
  for (i=0; i<4; i++)
  {
    value = HudaqEncRead(h,i);
    printf("Encoder channel %u, value read %d.\n", i, value);
  }

  /* Close the device handle. */
  HudaqCloseDevice(h);

  return(0);
}
```

9.15 IRCRead.c

```
/* IRCRead.c:
 *   This demo demonstrates how to read IRC from Encoder channels. It shows how to get
 *   a handle to Hudaq device, how to read data from encoder and how to optionally reset
 *   encoder value.
 */

#include <Windows.h>
#include <conio.h>
#include <stdio.h>

#include "..\hudaqlib.h"

void cls(void);


int main(int argc, char* argv[])
{
HUDAQHANDLE h;
unsigned i;
char Key;
int ValueFromEncoder;

  h = HudaqOpenDevice("MF614", 1, 0);     /* Device must be opened before usage. */
  if(h==0)
    {                                     /* No Hudaq device - return from application */
    printf("\nNo Hudaq device found!\n");
    return -1;
    }

  do
    {
    while(!kbhit())
      {
      cls();

      for(i=0; i<4; i++)
        {
        ValueFromEncoder = HudaqEncRead(h,i);       /* Read value from encoder */
        printf("Encoder IRC channel %d, value read %d\n", i, ValueFromEncoder); /* Print results on screen.
      */
        }

      printf("\n\npress ’0’ for reset channel 0, ’1’ -> 1, ’2’ -> 2; ’3’ -> 3.\n");
      printf("\npress any other key to exit this demo.\n");

      Sleep(60);
      }

    Key=getch();
    switch(Key)
      {
      case ’0’: HudaqEncReset(h,0);   /* Reset value inside given channel */
              break;
      case ’1’: HudaqEncReset(h,1);
              break;
      case ’2’: HudaqEncReset(h,2);
              break;
      case ’3’: HudaqEncReset(h,3);
              break;
      default:Key = 0;                  /* any other key stroke causes loop continuing */
      }

    } while(Key!=0);


 HudaqCloseDevice(h);           /* It is also neccessary to close device upon exit
       application. */

return 0;
}


  /* Auxiliarry function that clears a Windows console screen. */
void cls(void)
{
HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
   COORD coordScreen = { 0, 0 };    /* home for the cursor */
   DWORD cCharsWritten;
   CONSOLE_SCREEN_BUFFER_INFO csbi;
   DWORD dwConSize;
        /* Get the number of character cells in the current buffer. */
   if( !GetConsoleScreenBufferInfo( hConsole, &csbi )) return;
   dwConSize = csbi.dwSize.X * csbi.dwSize.Y;
        /* Fill the entire screen with blanks. */
   if( !FillConsoleOutputCharacter( hConsole, (TCHAR) ’ ’,dwConSize, coordScreen, &cCharsWritten )) return;
        /* Get the current text attribute. */
   if( !GetConsoleScreenBufferInfo( hConsole, &csbi )) return;
        /* Set the buffer’s attributes accordingly. */
   if( !FillConsoleOutputAttribute( hConsole, csbi.wAttributes,dwConSize, coordScreen, &cCharsWritten ))
      return;
        /* Put the cursor at its home coordinates. */
   SetConsoleCursorPosition( hConsole, coordScreen );
}
```

9.16 ListClassic.c

```
/* ListDevices.c:
 *   This demo demonstrates how to enumerate multiple devices.
 *   (c)2007-2020 Jaroslav Fojtik
 */
```

```
#include <stdio.h>

#include "hudaqlib.h"


typedef struct
{
  const char *DevName;
  int DevOrder;
} DeviceStruct;

int ListDevices(DeviceStruct *D)
{
int index=0;
int i,n;
HUDAQHANDLE h;
static const char *DevNames[]={"AD612","MF614","AD622","MF624","MF625","MF634",
                        "PCD-7004", "PCD-7006C", "PCD-8006", "PCI1753", "PCT-7303B",
                        "AD1753"};

  for(n=0; n<sizeof(DevNames)/sizeof(char*); n++)
    for(i=1; i<8; i++)
      {
      h = HudaqOpenDevice(DevNames[n], i, HudaqOpenNOINIT);  //Open a device
      if(h==0) break;
      HudaqCloseDevice(h);
      D[index].DevName = DevNames[n];
      D[index].DevOrder = i;
      index++;
      }
return index;
}


int main(int argc, char* argv[])
{
int i, count;
char ch;
HUDAQHANDLE h;
DeviceStruct ds[10];

  count = ListDevices(ds);
  if(count<=0) {printf("\nNo Hudaqdevice found");return(-1);}

  if(count>1)
    {
    for(i=0;i<count;i++)
      printf("\nPlease hit %c to choose %s card [%d]:",i+'A',ds[i].DevName,ds[i].DevOrder);
    printf("\n");
    i=toupper(getchar());
    i-='A';
    if (i<0 || i>=10) return -1;
    }
  else
    i=0;

  h = HudaqOpenDevice(ds[i].DevName,ds[i].DevOrder, HudaqOpenNOINIT);
  if(h==0)
    return -3;                         /* Device cannot be openned. */


  printf("\nDevice %s[%d] has been succesfully opened.", ds[i].DevName, ds[i].DevOrder);

  HudaqCloseDevice(h);                 /* Close handle */

return 0;
}
```

9.17 ListDevices.c

```
/* ListDevices.c:
 *   This demo demonstrates how to list devices.
 *
 *   This WILL NOT work with original Humusofts library!
 *    HudaqOpenDevice("",) with argument "" is not supported with original hudaqlib.
 */
#include <malloc.h>
#include <string.h>
#include <stdio.h>

#include "hudaqlib.h"
```

```
typedef struct
{
  char *DevName;
  int DevOrder;
} DeviceStruct;


int ListDevices(DeviceStruct *D)
{
int index=0;
int i;
HUDAQHANDLE h;

  do
  {
    h = HudaqOpenDevice("", index+1, HudaqOpenNOINIT);  //Open a device
    if(h==0) break;

    D[index].DevName = strdup(HudaqGetBoardName(h));
    D[index].DevOrder=1;

    for(i=0;i<index;i++)
      if(!strcmp(D[index].DevName,D[i].DevName)) D[index].DevOrder++;
    HudaqCloseDevice(h);

    index++;
  } while(h!=0);

return index;
}


int main(int argc, char* argv[])
{
int i, count;
char ch;
HUDAQHANDLE h;
DeviceStruct ds[10];

  count = ListDevices(ds);
  if(count<=0) {printf("\nNo Hudaqdevice found");return(-1);}

  if(count>1)
    {
    for(i=0;i<count;i++)
      printf("\nPlease hit %c to choose %s card [%d]:",i+’A’,ds[i].DevName,ds[i].DevOrder);
    printf("\n");
    i=toupper(getchar());
    i-=’A’;
    if (i<0 || i>=10) return;
    }
  else
    i=0;

  h = HudaqOpenDevice(ds[i].DevName,ds[i].DevOrder, HudaqOpenNOINIT);
  if(h==0)
    return -3;                         /* Device cannot be openned. */


  printf("\nDevice [%s(%d)] has been succesfully opened.", ds[i].DevName, ds[i].DevOrder);

  HudaqCloseDevice(h);                 /* Close handle */

return 0;
}
```

9.18 ProbeDevices.c

```
/* ProbeDevices.c:
 *   This demo demonstrates extract information from all available devices.
 *   (c)2007-2020 Jaroslav Fojtik
 *
 *   This WILL NOT work with original Humusofts library!
 *    HudaqOpenDevice("",) with argument "" is not supported with original hudaqlib.
 */
#include <stdio.h>
#include "hudaqlib.h"


int main(void)
```

```
{
HUDAQHANDLE h;
const HudaqResourceInfo *HRI;
int i,j;
double value;
int NoAnalogIn,NoDigitalIn,NoEncoders,NoCounters;
int dev = 1;
#ifdef _MSC_VER
unsigned __int64 DMA_MemRaw;
void *Pointer;
size_t BlockSz;
#endif
        /* Open first device found of any name. */
  h = HudaqOpenDevice("",1,0);
  if(h==0)
    {printf("No HUDAQ device found\n"); return -1;}

  while(h!=0)
  {
    printf("\n================ DEVICE FOUND ======================");

    HRI = HudaqGetDeviceResources(h);
    printf("\nBus number %d, Slot number %d.",HRI->BusNumber, HRI->SlotNumber);
    printf("\nVendorID %Xh, DeviceID %Xh.",HRI->VendorID,HRI->DeviceID);

    for(i=0; i<HRI->NumMemResources; i++)
    {
      printf("\n Memory resource %d: Base:%Xh, Length:%Xh",
          i, HRI->MemResources[i].Base, HRI->MemResources[i].Length);
    }

    for(i=0; i<HRI->NumIOResources; i++)
    {
      printf("\n IO resource %d: Base:%Xh, Length:%Xh",
        i, HRI->IOResources[i].Base, HRI->IOResources[i].Length);
    }

    NoAnalogIn = HudaqGetParameter(h,0,HudaqAINUMCHANNELS);
    printf("\nAnalog channels AI:%d / AO:%d", NoAnalogIn, (int)HudaqGetParameter(h,0,
      HudaqAONUMCHANNELS));
    for (i=0; i<NoAnalogIn; i++)
    {
      value = HudaqAIRead(h,i);
      printf("\n  Analog channel %d, value read %fV.", i, value);
    }

    NoDigitalIn = HudaqGetParameter(h,0,HudaqDINUMCHANNELS);
    printf("\nDigital channels DI:%d / DO:%d", NoDigitalIn, (int)
      HudaqGetParameter(h,0,HudaqDONUMCHANNELS));
    for (i=0; i<NoDigitalIn; i++)
    {
      printf("\n  Digital input %d: %d",i,HudaqDIRead(h,i));
    }

    NoEncoders = HudaqGetParameter(h,0,HudaqEncNUMCHANNELS);
    printf("\nEncoder channels %d", NoEncoders);
    for (i=0; i<NoEncoders; i++)
    {
      printf("\n  Encoder value %d: %d",i,HudaqEncRead(h,i));
    }

    NoCounters = HudaqGetParameter(h,0,HudaqCtrNUMCHANNELS);
    printf("\nCounter channels %d", NoCounters);
    for (i=0; i<NoCounters; i++)
    {
      printf("\n  Counted value %d: %d",i,HudaqCtrRead(h,i));
    }

    printf("\nPWM channels %d", (int)HudaqGetParameter(h, 0, HudaqPwmNUMCHANNELS));

#ifdef _MSC_VER
    printf("\nIRQ counter: %g (%g)",
             HudaqGetParameter(h,0,HudaqIRQ), HudaqGetParameter(h, 0,
      HudaqIRQ+1));

    HudaqGetDMAinfo(h,&DMA_MemRaw,&Pointer,&BlockSz);
    printf("\nDMA mem raw %lX, pointer %p, size: %u.", DMA_MemRaw,Pointer,(unsigned)BlockSz);
#endif

    HudaqCloseDevice(h);
    h = HudaqOpenDevice("",++dev,0);
  }

  //  HudaqSetParameter(h, 0, HudaqIRQ, 1);
  //i = HudaqGetParameter(h, 0, HudaqIRQ);
  printf("\n");
return 0;
}
```

9.19 PWM3Write.c

```
/* Humusoft data acquisition library.
 *
 * Example that shows using of 3 phase PWM output channels.
 * This example works on MF625 only!
 */

/* Copyright 2002-2007 Humusoft s.r.o. */

#include <stdio.h>
#include <windows.h>

#include "hudaqlib.h"


int main(int argc, char* argv[])
{
  HUDAQHANDLE h;
  double value;

  /* open a handle to the first MF625 device in the system */
  h = HudaqOpenDevice("MF625", 1, 0);
  if (h==0)
  {
    printf("\nData acquisition device MF625 not found.\n");
    return(-1);
  }

  /* set the first PWM channel to frequency 1.5kHz with duty cycles 0.1 0.5 0.9 */
  HudaqPWM3Write(h, 0, 1500, 0.5, 0.5, 0.9);

value=0;
while(1)
{
  HudaqSetParameter(h, 0, HudaqPwmDEADBAND, 1e-5);
  HudaqPWM3Write(h, 0, 1500, value, 1-value, value);
  value+=5e-4;
  if(value>1) value=0;
  Sleep(1);
}


  /* close the device handle */
  HudaqCloseDevice(h);

  return(0);
}
```

9.20 PWMWrite.c

```
/* Humusoft data acquisition library.
 *
 * Example that shows using of PWM output channels.
 */
/* Copyright 2002-2007 Humusoft s.r.o.
   Copyright 2008-2020 Jaroslav FOjtik */

#include <stdio.h>

#include "hudaqlib.h"


int main(int argc, char* argv[])
{
HUDAQHANDLE h;
double value;
unsigned ret;

  /* open a handle to the first MF624 device in the system */
  h = HudaqOpenDevice("MF624", 1, 0);
  if(h==0)
  {
    printf("\nData acquisition device not found.\n");
    return(-1);
  }

  /* set first PWM channel to frequency 1.5kHz with duty cycle 0.5 */
  ret = HudaqPWMWrite(h,0,1500,0.5);
  if(ret!=HUDAQSUCCESS)
    printf("\nCannot write to PWM channel 0, error %d.\n",ret);

  /* set second PWM channel to frequency 2.5kHz with duty cycle 0.75 */
  ret = HudaqPWMWrite(h,1,2500,0.75);
  if(ret!=HUDAQSUCCESS)
    printf("\nCannot write to PWM channel 1, error %d.\n",ret);


  /* close the device handle */
  HudaqCloseDevice(h);

  return(0);
}
```