

Dsb496 API Overview

Document Purpose

This document provides information for controlling the dsb496 Digital IO module from various programming languages using either the newer managed class library dll or the older standard dll for applications that do not support the managed class library. The sample source code directories contain samples for various languages that demonstrate use of each of the functions in a simple straight-forward manner and can be used as a starting point for user application. The samples applications show the exact calling sequence for each of the function described in this document including the full variable and function declarations. This text is meant to just give an overview of what is available. The user will need to have the programming language software installed on the pc used, and should be familiar with that language operation and use. This document only contains material related to the unique module dll software and is not intended as a programming language tutorial. All examples on the CD have been built and tested for a given version of programming software. Updated versions of programming software may require slight modifications to code or option settings.

Hardware Overview

The hardware is 96 bits of digital IO lines. Groups of 8 bits can be configured as either input or output. Each group of 8 bits is call a "Port" and are numbered from 1 to 12, Three (3) ports are grouped together to form what is call a bank numbered from 1 to 4. The lsb of a data value matches the lsb of a port or bank. In the case of a bank, unused bit (above bit 23) are ignored. Each 50 pin connector on the module contains a bank. Refer to the pin-out documents for exact locations.

Class Library API

The class library is a set of functions that can be called from any programming language supporting calls to a managed class library. It provides the necessary calls to the modules usb driver to do the requested action on the module, providing simple user interface. The class library name is "dsb496x32CL.dll" (32 bit version) and dsb496x64CL (64 bit version). Included with the class library is an .xml file that is used by Visual Studio to support IntelliSense help to make using the functions quick and easy. Both files can be found on the CD in the 'Class Library' directory. Local copies may also be in sample code directories to ease the transferring of the sample source to your working directories, as some Visual Studio version do not support relative paths. This Class Library is supported on Visual Studio 2005 and later.

Get the number of modules attached

`NumberOfModules()`

This function returns the number of modules current attached to the system.

Get the serial number of a module

`SerialNumber(module number, pointer to string, size of string buffer)`

This function returns the serial number of for the requested module number (range 1-256). The serial number is used to access multiple modules on the system. When a system is powered up with multiple modules attached the units may come up in a different order, and serial numbers are used to find them.

Get the DLL version

`DllVersion(pointer to string, size of string buffer)`

This function returns the dll version being used for the given serial number. `Ddsb496GetDllVersion`

Get the driver version

`DriverVersion(serial number, pointer to string, size of string buffer)`

This function returns the driver version being used for the given serial number.

Get the module firmware version

`FirmwareVersion(serial number, pointer to string, size of string buffer)`

This function returns the firmware version for the module.

Set port to input mode

`SetPortToInput(serial number, port number)`

This function sets the port to input mode.

Set port to output mode

`SetPortToOutput(serial number, port number)`

This function sets the port to output mode. The last values written to the output registers will be used.

Dsb496 API Overview

Check what state the port is in

`IsPortInput(serial number, port number)`

This function returns a true if port is an input, a false if it is an output.

Read a bank value

`ReadBank(serial number, bank number)`

This function returns the value of the given bank. If any ports are on outputs on that bank, the value will be the output value.

Read a port value

`ReadPort(serial number, port number)`

This function returns the value of the given port. If the port is an output, then that value is returned.

Write a bank value

`WriteBank(serial number, bank number , value)`

This function writes the value out to the given bank. The value is written to a set of output registers. If the port is set as an input that value will not change until port is changed to an input. Values are latched after all are written, so will change together.

Write a port value

`WritePort(serial number, port number, value)`

This function writes the value out to the given port. If the port is an input, that value will not be change until port is changed to an output.

Dsb496 API Overview

DLL API

The dll files dsb496x32.dll (32 bit version) or dsb496x64.dll (64 bit version) contain a set of callable functions that can be called from any programming language supporting calls to a standard windows dll. It provides a high level interface between a user application and the provided module usb drivers. The files are located under the "DLL" directory on the CD and may also appear in sample code directories to ease build. Also included is the file dsb49xDll.h which has function prototypes for languages requiring declarations.

Get the number of modules attached

`dsb496GetNumberOfModules()`

This function returns the number of modules current attached to the system.

Get the serial number of a module

`dsb496GetSerialNumber(module number, pointer to string, size of string buffer)`

This function returns the serial number of for the requested module number (range 1-256). The serial number is used to access multiple modules on the system. When a system is powered up with multiple modules attached the units may come up in a different order, and serial numbers are used to find them.

Get the DLL version

`dsb496GetDllVersion(pointer to string, size of string buffer)`

This function returns the dll version being used for the given serial number. `Ddsb496GetDllVersion`

Get the driver version

`dsb496GetDriverVersion(serial number, pointer to string, size of string buffer)`

This function returns the driver version being used for the given serial number.

Get the module firmware version

`dsb496GetFirmwareVersion(serial number, pointer to string, size of string buffer)`

This function returns the firmware version for the module.

Set port to input mode

`dsb496SetPortToInput(serial number, port number)`

This function sets the port to input mode.

Set port to output mode

`dsb496SetPortToOutput(serial number, port number)`

This function sets the port to output mode. The last values written to the output registers will be used.

Check what state the port is in

`dsb496IsPortInput(serial number, port number)`

This function returns a true if port is an input, a false if it is an output.

Read a bank value

`dsb496ReadBank(serial number, bank number)`

This function returns the value of the given bank. If any ports are on outputs on that bank, the value will be the output value.

Read a port value

`dsb496ReadPort(serial number, port number)`

This function returns the value of the given port. If the port is an output, then that value is returned.

Write a bank value

`dsb496WriteBank(serial number, bank number , value)`

This function writes the value out to the given bank. The value is written to a set of output registers. If the port is set as an input that value will not change until port is changed to an input. Values are latched after all are written, so will change together.

Dsb496 API Overview

Write a port value

`dsb496WritePort(serial number, port number, value)`

This function writes the value out to the given port. If the port is an input, that value will not be change until port is changed to an output.