Nubotics Device Interface DLL

ver-1.1

Generated by Doxygen 1.5.5

Mon Mar 2 17:01:02 2009

Contents

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Module Documentation

4.1 Initialization Functions

4.1.1 Detailed Description

NdiCmdC.DLL Noetic Design, Inc. Command Interface Library.

This DLL can be used by Windows developers from C, C++, or Visual Basic. Use these functions to initiate communications.

Functions

- NDICMDC_API void NdiCmd_Init (void)

  Initialize the DLL.

- NDICMDC_API void NdiCmd_SetLogWindow (HWND wnd)

  Specify Output Control for Logging.

- NDICMDC_API void NdiCmd_SetLogFile (char *path, BOOL append)

  Specify Output File for Logging.

- NDICMDC_API void NdiCmd_LogIt (char *text)

  Send Text to the Log.

- NDICMDC_API void NdiCmd_LogQuiet (BOOL quiet)

  Set Logging Level.

- NDICMDC_API void NdiCmd_EnableDirect (void)

Enable Direct.

- NDICMDC_API BOOL NdiCmd_Search (void)

  Search for Ndi Devices.

- NDICMDC_API BOOL NdiCmd_CheckPort (int i)

  Check Port.

- NDICMDC_API int NdiCmd_GetFirstPort (void)

  Get First Port.

- NDICMDC_API int NdiCmd_GetNextPort (int prev)

  Get Next Port.

- NDICMDC_API void NdiCmd_SetPortEnabled (int i, BOOL enabled)

  Set Port Enabled.

- NDICMDC_API BOOL NdiCmd_GetPortEnabled (int i)

  Get Port Enabled.

- NDICMDC_API void NdiCmd_CloseAllPorts (void)

  Close All Ports.

- NDICMDC_API int NdiCmd_GetWheelCommander (void)

  Get Wheel Commander.

- NDICMDC_API int NdiCmd_GetNumWheelCommanders (void)

  Get Num Wheel Commanders.

- NDICMDC_API int NdiCmd_GetWheelCommanderNum (int num)

  Get Wheel Commander Num.

- NDICMDC_API int NdiCmd_GetUnicoder (void)

  Get Unicoder.

- NDICMDC_API int NdiCmd_GetNumUnicoders (void)

  Get Num Unicoders.

- NDICMDC_API int NdiCmd_GetUnicoderNum (int index)

  Get Unicoder Num.

- NDICMDC_API void NdiCmd_RunWizard (BOOL wiz_mode)

Run Wizard.

- NDICMDC_API BOOL NdiCmd_UpdateFirmware (char *filename, HWND statwindow)

  Update Firmware.

- NDICMDC_API BOOL NdiCmd_UpdateFirmwareEx (char *filename, HWND statwindow, BOOL special, BOOL direct)

  Update Firmware Ex.

- NDICMDC_API void NdiCmd_SetStatusWindow (HWND hwnd)

  Set Status Window.

4.1.2 Function Documentation

4.1.2.1 NDICMDC_API BOOL NdiCmd_CheckPort (int *i*)

Check Port.

Call this function to determine if the given port is accessible.

Parameters:

i the port number to check (values start at 1 under Windows)

Returns:

TRUE (1) if the specified port is accessible.

4.1.2.2 NDICMDC_API void NdiCmd_CloseAllPorts (void)

Close All Ports.

Call this function when you are about to exit your program.

Returns:

none.

4.1.2.3 NDICMDC_API void NdiCmd_EnableDirect (void)

Enable Direct.

Call this function to enable communication with bootloader inside Unicoder.

Returns:

TRUE (1) if a device has been found.

4.1.2.4 NDICMDC_API int NdiCmd_GetFirstPort (void)

Get First Port.

This function returns the port number of the first valid serial port.

Returns:

TRUE (1) if at least one valid serial port is present.

4.1.2.5 NDICMDC_API int NdiCmd_GetNextPort (int *prev*)

Get Next Port.

Given a previous serial port number, this returns the next valid.

Parameters:

prev an integer representing a port previously found.

Returns:

integer value of next available port, or -1 if none present.

4.1.2.6 NDICMDC_API int NdiCmd_GetNumUnicoders (void)

Get Num Unicoders.

After calling NdiCmd_Search(), this returns the number of Unicoders found.

Returns:

integer count.

4.1.2.7 NDICMDC_API int NdiCmd_GetNumWheelCommanders (void)

Get Num Wheel Commanders.

After calling NdiCmd_Search(), this returns the number of WheelCommanders found.

Returns:

integer count.

4.1.2.8 NDICMDC_API BOOL NdiCmd_GetPortEnabled (int *i*)

Get Port Enabled.

This function returns TRUE if the specified port is enabled for scanning.

Parameters:

i the port to check.

Returns:

TRUE (1) if this port is enabled for scanning.

4.1.2.9 NDICMDC_API int NdiCmd_GetUnicoder (void)

Get Unicoder.

After calling NdiCmd_Search(), this returns an integer handle representing the first Unicoder; use this value when accessing.

Returns:

integer handle.

4.1.2.10 NDICMDC_API int NdiCmd_GetUnicoderNum (int *index*)

Get Unicoder Num.

After calling NdiCmd_Search(), this returns an integer handle representing the specified Unicoder; use this value when accessing.

Parameters:

index the number of the Unicoder whose handle should be retrieved

Returns:

integer handle.

4.1.2.11 NDICMDC_API int NdiCmd_GetWheelCommander (void)

Get Wheel Commander.

After calling NdiCmd_Search(), this returns an integer handle representing the WheelCommander; use this value when accessing.

Returns:

integer handle.

4.1.2.12 NDICMDC_API int NdiCmd_GetWheelCommanderNum (int *num*)

Get Wheel Commander Num.

After calling NdiCmd_Search(), this returns an integer handle representing the specified WheelCommander; use this value when accessing.

Parameters:

index the number of the WheelCommadner whose handle should be retrieved

Returns:

integer handle.

4.1.2.13 NDICMDC_API void NdiCmd_Init (void)

Initialize the DLL.

Call this function first before using any other NdiCmdC functions.

4.1.2.14 NDICMDC_API void NdiCmd_LogIt (char * *text*)

Send Text to the Log.

Call this function to send a text string to the log.

Parameters:

text a pointer to a null-terminated string

Returns:

none

4.1.2.15 NDICMDC_API void NdiCmd_LogQuiet (BOOL *quiet*)

Set Logging Level.

Call this function to either quiet down logging or make it more verbose for debugging purposes.

Parameters:

quiet a boolean

Returns:

none

4.1.2.16 NDICMDC_API void NdiCmd_RunWizard (BOOL *wiz_mode*)

Run Wizard.

This function launches the WheelCommander configuration wizard or settings dialogs, depending on the wiz_mode parameter.

Parameters:

wiz_mode boolean value, set to 1 to launch a step-by-step wizard, 0 for a normal dialog.

Returns:

none.

4.1.2.17 NDICMDC_API BOOL NdiCmd_Search (void)

Search for Ndi Devices.

Call this function to search all enabled serial ports for devices at the current baud rate.

Returns:

TRUE (1) if a device has been found.

4.1.2.18 NDICMDC_API void NdiCmd_SetLogFile (char * *path*, BOOL *append*)

Specify Output File for Logging.

Call this function to cause debug information to be written to a file.

Parameters:

path to file

BOOL append true to append to existing log; false to overwrite

Returns:

none

4.1.2.19 NDICMDC_API void NdiCmd_SetLogWindow (HWND *wnd*)

Specify Output Control for Logging.

Call this function to cause debug information to be displayed on the specified edit control.

Parameters:

wnd a pointer to a rich edit control

Returns:

none

4.1.2.20 NDICMDC_API void NdiCmd_SetPortEnabled (int *i*, BOOL *enabled*)

Set Port Enabled.

Use this function to enable or disable the specified port number for future scanning with by NdiCmd_Search(). By default, all valid serial ports are enabled. This can result in long NdiCmd_Search() calls with certain BlueTooth serial port profile implementations; in that case, disable all but the expected port.

Parameters:

i the port to enable.

enabled a boolean value of 1 to enable or 0 to disable.

Returns:

none.

4.1.2.21 NDICMDC_API void NdiCmd_SetStatusWindow (HWND *hwnd*)

Set Status Window.

Progress messages created during the upgrade process will be displayed on either the console (stdout) or in a Window, or will be suppressed, depending on the value of hwnd:

- -1 send messages to stdout

- 0 suppress output
- HWND send messages to this windows handle, which will be sent WM_-SETTEXT messages with LPARAM containing a pointer to the message text

Parameters:

hwnd a Windows handle in which status messages from this library can be displayed.

Returns:

none.

4.1.2.22 NDICMDC_API BOOL NdiCmd_UpdateFirmware (char * *filename*, HWND *statwindow*)

Update Firmware.

This function will perform all the tasks needed to upgrade the firmware.

Parameters:

filename a string (of type char *) containing the name of the file on your system.

statwindow a Windows handle (of type HWND) to display progress,-1 to display on the console, or 0 for none.

Returns:

TRUE (1) if the firmware was correctly updated.

4.1.2.23 NDICMDC_API BOOL NdiCmd_UpdateFirmwareEx (char * *filename*, HWND *statwindow*, BOOL *special*, BOOL *direct*)

Update Firmware Ex.

This function is the same as NdiCmd_UpdateFirmware() except that it provides additional control over which type of firmware file is provided as well as allows for emergency recovery when only the bootloader is valid.

Parameters:

filename a string (of type char *) containing the name of the file on your system.

statwindow a Windows handle (of type HWND) to display progress,-1 to display on the console, or 0 for none.

special a boolean indicating an alternate hardware configuration is present

direct a boolean indicating emergency recovery is needed

Returns:

TRUE (1) if the firmware was correctly updated.

4.2 Communications Functions

4.2.1 Detailed Description

Use these functions to test communications and gather information about the device.

Functions

- NDICMDC_API BOOL NdiCmd_GetPortInfo (int handle, char *type_string, int *id)

  Get Port Info.

- NDICMDC_API BOOL NdiCmd_Sync (int handle)

  Synchronize.

- NDICMDC_API BOOL NdiCmd_CommTest (int handle, int first, int last, int step)

  Comm Test.

- NDICMDC_API BOOL NdiCmd_GetName (int handle, LPSTR name, int *version)

  Get Name.

- NDICMDC_API BOOL NdiCmd_Status (int handle, unsigned int *status)

  Status.

4.2.2 Function Documentation

4.2.2.1 NDICMDC_API BOOL NdiCmd_CommTest (int *handle*, int *first*, int *last*, int *step*)

Comm Test.

This function performs a rapid communications link test, by executing a series of WheelCommander Echo commands with values starting at first, ending at last, and increasing from first to last by step size increments.

Parameters:

handle an integer WheelCommander handle

first the first value to test with

last the last value to test with

step the increment amount

Returns:

4.2.2.2 NDICMDC_API BOOL NdiCmd_GetName (int *handle*, LPSTR *name*, int * *version*)

Get Name.

This returns the device name and version for the device at the specified handle.

Parameters:

handle an integer WheelCommander handle

name a character pointer to the device name

version a pointer to an integer containing the version

Returns:

4.2.2.3 NDICMDC_API BOOL NdiCmd_GetPortInfo (int *handle*, char * *type_string*, int * *id*)

Get Port Info.

This returns the current device type and id number this device is connected with.

Parameters:

type_string a pointer to a character buffer; length must be >= 64 characters.

id a pointer to an integer that receives the Id number

Returns:

TRUE if successful

4.2.2.4 NDICMDC_API BOOL NdiCmd_Status (int *handle*, unsigned int * *status*)

Status.

This returns the current status of the WheelCommander.

Parameters:

handle an integer WheelCommander handle

status a pointer to an unsigned int, returning the status bits

Returns:

4.2.2.5 NDICMDC_API BOOL NdiCmd_Sync (int *handle*)

Synchronize.

This command is used to confirm communications with the WheelCommander. It also clears out the WheelCommander's serial command buffer.

Parameters:

handle an integer WheelCommander handle

Returns:

TRUE(1) if synchronization was successful.

4.3 Settings Functions

4.3.1 Detailed Description

Use these functions to change communications and platform configuration settings.

Typedefs

- typedef enum _wc_baud **WC_BAUD**
- typedef enum _wc_pwm_frequency **WC_PWMFREQ**

Enumerations

- enum _wc_baud {
  BD_1200, BD_4800, BD_9600, BD_19200,
  BD_38400, BD_57600, BD_115200, BD_230400,
  BD_460800 }
  Baud Rate Values.

- enum _wc_pwm_frequency {
  PF_2KHZ, PF_4KHZ, PF_8KHZ, PF_16KHZ,
  PF_32KHZ, PF_64KHZ }
  PWM frequencies.

Functions

- NDICMDC_API BOOL NdiCmd_SetBaud (int handle, WC_BAUD baud)
  Set Baud.

- NDICMDC_API BOOL NdiCmd_GetBaud (int handle, int *baud)
  Get Baud.

- NDICMDC_API void NdiCmd_SetBlueRadio (BOOL enable, BOOL fast)
  Set Blue Radio.

- NDICMDC_API void NdiCmd_GetBlueRadio (BOOL *enable, BOOL *fast)
  Get Blue Radio.

- NDICMDC_API BOOL NdiCmd_SetI2CAddress (int handle, int addr)

Set I2C Address.

- NDICMDC_API BOOL NdiCmd_GetI2CAddress (int handle, BYTE *addr)

  Get I2C Address.

- NDICMDC_API BOOL NdiCmd_SetCommMode (int handle, BOOL slow_comms, BOOL notify, BOOL short_mode)

  Set Comm Mode.

- NDICMDC_API BOOL NdiCmd_SetPlatformSpecs (int handle, BOOL polarity, BOOL counter_axis, BYTE wheel_base, BYTE wheel_diameter)

  Set Platform Specs.

- NDICMDC_API BOOL NdiCmd_SetMotorMode (int handle, BOOL pwm, BOOL locked_antiphase, BOOL dual_dir_lines, WC_PWMFREQ pwm_frequency)

  Set Motor Mode.

- NDICMDC_API BOOL NdiCmd_GetEncoderProperties (int handle, BOOL *quadrature, WORD *ticks_per_rotation)

  Get Encoder Properties.

- NDICMDC_API BOOL NdiCmd_SetEncoderProperties (int handle, BOOL quadrature, WORD ticks_per_rotation)

  Set Encoder Properties.

4.3.2 Enumeration Type Documentation

4.3.2.1 enum _wc_baud

Baud Rate Values.

These values are passed to NdiCmd_SetBaud().

Enumerator:

BD_1200 1200 baud

BD_4800 4800 baud

BD_9600 9600 baud

BD_19200 19200 baud

BD_38400 38400 baud

BD_57600 57600 baud

BD_115200 115200 baud
BD_230400 230400 baud
BD_460800 460800 baud

4.3.2.2 enum _wc_pwm_frequency

PWM frequencies.

These values are passed to NdiCmd_SetMotorMode().

Enumerator:

PF_2KHZ 2KHz
PF_4KHZ 4KHz
PF_8KHZ 8KHz
PF_16KHZ 16KHz
PF_32KHZ 32KHz
PF_64KHZ 64KHz

4.3.3 Function Documentation

4.3.3.1 NDICMDC_API BOOL NdiCmd_GetBaud (int *handle*, int * *baud*)

Get Baud.

This function gets the current PC serial port baud rate, which will also be the value it expects the WheelCommander and optional BlueRadios module to be set to.

Parameters:

handle an integer WheelCommander handle
baud integer pointer to the returned baud rate value

Returns:

4.3.3.2 NDICMDC_API void NdiCmd_GetBlueRadio (BOOL * *enable*, BOOL * *fast*)

Get Blue Radio.

This function lets you query the state of BlueRadio handling.

Parameters:

enable a pointer to a BOOL value returning the enable state

fast a pointer to a BOOL value returning the fast data mode

Returns:

4.3.3.3 NDICMDC_API BOOL NdiCmd_GetEncoderProperties (int *handle*, BOOL * *quadrature*, WORD * *ticks_per_rotation*)

Get Encoder Properties.

This allows the user to retrieve the current encoder properties.

Parameters:

handle an integer WheelCommander handle

quadrature a pointer to a boolean having a value of TRUE if the encoder has quadrature outputs, or FALSE if it has sign/magnitude outputs (the default)

ticks_per_rotation a pointer to a 16 bit integer indicating the number of encoder ticks per wheel rotation (128 by default)

Returns:

BOOL

4.3.3.4 NDICMDC_API BOOL NdiCmd_GetI2CAddress (int *handle*, BYTE * *addr*)

Get I2C Address.

This function returns the slave I2C address of the Unicoder encoder.

Parameters:

handle an integer Unicoder handle

addr a pointer to a BYTE, containing the returned slave address

Returns:

4.3.3.5 NDICMDC_API BOOL NdiCmd_SetBaud (int *handle*, WC_BAUD *baud*)

Set Baud.

This function sets the WheelCommander baud rate, the BlueRadio Bluetooth module's baud rate (if enabled), and finally the PC's serial port baud rate, to the specified rate. This change takes place immediately, and is stored in EEPROM on the WheelCommander as well as in the Windows registry on the PC. If BlueRadio support is turned on via NdiCmd_SetBlueRadio(), the baud rate will also be stored in the BlueRadio module's EEPROM.

Parameters:

handle an integer WheelCommander handle

baud integer baud rate value

Returns:

4.3.3.6 NDICMDC_API void NdiCmd_SetBlueRadio (BOOL *enable*, BOOL *fast*)

Set Blue Radio.

This function allows you to change the WheelCommander's BlueRadio settings. You can enable or disable BlueRadio functionality, and optionally turn on fast data mode. NOTE: fast data mode can only be turned off by power cycling the BlueRadio module.

Parameters:

enable a BOOL value to turn on BlueRadio handling

fast a BOOL value to enable fast data mode

Returns:

4.3.3.7 NDICMDC_API BOOL NdiCmd_SetCommMode (int *handle*, BOOL *slow_comms*, BOOL *notify*, BOOL *short_mode*)

Set Comm Mode.

This routine provides a simplified interface to the bit-encoded MODE parameter byte in the WheelCommander. With it, one can change the slow_comms bit (for BasicStamp

compatibility), the notify bit (for asynchronous motion completion notification), and the short_mode bit, for enabling shorter return strings (especially useful in I2C mode on an OOPIC). Changes are stored in the WheelCommander's EEPROM, and take place immediately.

Parameters:

handle an integer WheelCommander handle

slow_comms a BOOL set to 1 to insert a 40 millisecond delay between each transmitted character

notify a BOOL set to 1 to delay response to GO until the WheelCommander is at speed or has reached the goal position; 0 by default

short_mode a BOOL, set to 1 to prevent echoing of the command character at the start of the return packet

Returns:

4.3.3.8 NDICMDC_API BOOL NdiCmd_SetEncoderProperties (int *handle*, BOOL *quadrature*, WORD *ticks_per_rotation*)

Set Encoder Properties.

This allows the user to set up the WheelCommander for encoders with specifications other than those of the Nubotics WheelWatcher product line.

Parameters:

handle an integer WheelCommander handle

quadrature a boolean having a value of TRUE if the encoder has quadrature outputs, or FALSE if it has sign/magnitude outputs (the default)

ticks_per_rotation a 16 bit integer indicating the number of encoder ticks per wheel rotation (128 by default)

Returns:

BOOL

4.3.3.9 NDICMDC_API BOOL NdiCmd_SetI2CAddress (int *handle*, int *addr*)

Set I2C Address.

This function allows you to change the WheelCommander's I2C 7 bit address, whether connected via I2C with the PC or not. The value specified is stored in the WheelCommander's EEPROM; changes take place immediately. Recommended values are:

- 0x10
- 0x11 ...
- 0x17 where bits 3-6 contain the recommended device ID of 0010b (2), and the least significant 3 bits contains the slave address (0-7).

Parameters:

handle an integer WheelCommander handle
addr an integer containing the new device ID

Returns:

4.3.3.10 NDICMDC_API BOOL NdiCmd_SetMotorMode (int *handle*, BOOL *pwm*, BOOL *locked_antiphase*, BOOL *dual_dir_lines*, WC_PWMFREQ *pwm_frequency*)

Set Motor Mode.

This function allows you to enable either servo mode (by setting pwm FALSE), or DC motor mode. In DC motor mode, the locked_antiphase, dual_dir_lines, and pwm_-frequency parameters are used to specify how the H bridges should be controlled.

Parameters:

handle an integer WheelCommander handle
pwm a BOOL value, where 1 = TRUE enables DC motor mode
locked_antiphase a BOOL value enables duty-cycle based speed and direction control
dual_dir_lines a BOOL value enables use of D2 and D3 as the inverted values of the DIR lines
pwm_frequency a member of the WC_PWMFREQ enumeration

Returns:

4.3.3.11 NDICMDC_API BOOL NdiCmd_SetPlatformSpecs (int *handle*, BOOL *polarity*, BOOL *counter_axis*, BYTE *wheel_base*, BYTE *wheel_diameter*)

Set Platform Specs.

This simplifies access to other mode bits, as well as other EEPROM values, that deal with configuration of the robot platform.

Parameters:

handle an integer WheelCommander handle

polarity a BOOL, used to control the forward direction of the platform

counter_axis a BOOL, used to enable inverted motion of the two wheels relative to each other

wheel_base a BYTE containing the distance between the centerlines of the wheels in the current distance units

wheel_diameter a BYTE containing the wheel diameter, in the same distance units (multiples of .1" or mm work well)

Returns:

4.4 Analog and Digital I/O Functions

4.4.1 Detailed Description

Use these functions to read or write to digital I/O pins or read from analog input pins.

Data Structures

- union _int_union_

  INT FLAGS.

- union _io_flags_union

  IO FLAGS.

Typedefs

- typedef union _int_union_ **INT_FLAGS**
- typedef union _io_flags_union **IO_FLAGS**

Functions

- NDICMDC_API BOOL NdiCmd_ReadDIO (int handle, int pin, BOOL *value)

  Read Digital Input.

- NDICMDC_API BOOL NdiCmd_WriteDIO (int handle, int pin, BOOL value)

  Write Digital Output.

- NDICMDC_API BOOL NdiCmd_ReadAIO (int handle, int pin, double *value)

  Read Analog Input.

- NDICMDC_API BOOL NdiCmd_SetInterrupt (int handle, INT_FLAGS flags)

  Set Interrupt.

- NDICMDC_API BOOL NdiCmd_GetInterruptStatus (int handle, INT_FLAGS *flags, WORD *io_lines)

  Get Interrupt Status.

- NDICMDC_API BOOL NdiCmd_SetDIOMonitorInfo (int handle, int pin, IO_-FLAGS flags)

  Set DIO Monitor Info.

- NDICMDC_API BOOL NdiCmd_SetAIOMonitorInfo (int handle, int pin, IO_-FLAGS flags, WORD lo, WORD hi)

  Set AIO Monitor Info.

4.4.2 Function Documentation

4.4.2.1 NDICMDC_API BOOL NdiCmd_GetInterruptStatus (int *handle*, INT_FLAGS * *flags*, WORD * *io_lines*)

Get Interrupt Status.

When the host determines that the interrupt line has been asserted, it should call this function to find out what caused the interrupt. It must then clear the interrupt by querying the state of the active source.

Parameters:

handle an integer WheelCommander handle

flags a pointer to an INT_FLAGS bitfield structure

io_lines a pointer to a WORD containing one bit per digital input in bits 0-7, and a bit for each analog input in bits 8-15.

Returns:

BOOL

4.4.2.2 NDICMDC_API BOOL NdiCmd_ReadAIO (int *handle*, int *pin*, double * *value*)

Read Analog Input.

Given the specified pin, this reads the analog voltage level by reading the 10 bit A/D converter value, then scaling that by 5.0 volts per 1024 values.

Parameters:

handle an integer WheelCommander handle

pin an integer specifying which analog input to read

value a pointer to a double returning the voltage

Returns:

BOOL

4.4.2.3 NDICMDC_API BOOL NdiCmd_ReadDIO (int *handle*, int *pin*, BOOL * *value*)

Read Digital Input.

Given the I/O pin number to read (0-7), this sets the specified pin to be a high-impedance input, then returns the current logic level of that pin at the BOOL pointed to by value.

Parameters:

handle an integer WheelCommander handle

pin an integer specifying which DIO pin to read

value a pointer to a BOOL, where the level is returned

Returns:

BOOL

4.4.2.4 NDICMDC_API BOOL NdiCmd_SetAIOMonitorInfo (int *handle*, int *pin*, IO_FLAGS *flags*, WORD *lo*, WORD *hi*)

Set AIO Monitor Info.

For the analog input 'pin', set up monitoring (or turn it off) based on the IO_FLAGS parameter flags and the lo and/or hi reference values. If watch_for_hi and watch_for_-lo are both set in flags, then lo and hi can either specify a range outside of which an interrupt occurs (if lo $<$ hi) or a range inside of which an interrupt occurs (if hi $<$ lo). If only one of watch_for_lo or watch_for_hi is set, then this allows for a simple threshold test.

Parameters:

handle an integer WheelCommander handle

pin an integer indicating the digital input pin number (0-7)

flags an IO_FLAGS bitfield structure

lo a 10 bit analog input reference point (stored in a 16 bit WORD)

hi a 10 bit analog input reference point (stored in a 16 bit WORD)

Returns:

BOOL

4.4.2.5 NDICMDC_API BOOL NdiCmd_SetDIOMonitorInfo (int *handle*, int *pin*, IO_FLAGS *flags*)

Set DIO Monitor Info.

For the digital input 'pin', set up monitoring (or turn it off) based on the IO_FLAGS parameter flags. If watch_for_lo is set in flags, then an interrupt will occur when the input goes low; if watch_for_hi, when it goes high. If both are set, then any change causes an interrupt. None of this occurs if monitor is not set in flags, or if 'input' in flags is clear, or if 'digital' in flags is clear. If the flag called 'set' is high in the IO_FLAGS parameter passed, in, then the user interrupt will be immediately asserted.

Parameters:

handle an integer WheelCommander handle

pin an integer indicating the digital input pin number (0-7)

flags an IO_FLAGS bitfield structure

Returns:

BOOL

4.4.2.6 NDICMDC_API BOOL NdiCmd_SetInterrupt (int *handle*, INT_FLAGS *flags*)

Set Interrupt.

This provides a means to enable assertion of a specified digital output line, either as an active high or low signal, upon the occurance of certain conditions (motion operation complete; digital input at a certain state; analog input in a certain range). Digital and analog input monitoring is set using NdiCmd_SetDIOMonitorInfo() and NdiCmd_SetAIOMonitorInfo().

Parameters:

handle an integer WheelCommander handle

flags an INT_FLAGS bitfield structure

Returns:

BOOL

4.4.2.7 NDICMDC_API BOOL NdiCmd_WriteDIO (int *handle*, int *pin*, BOOL *value*)

Write Digital Output.

This sets the specified pin to be a digital output, then sets the logic level of the specified pin to value.

Parameters:

handle an integer WheelCommander handle

pin an integer specifying which DIO pin to write

value a BOOL setting the level of the pin

Returns:

BOOL

4.5 Motion Control Functions

4.5.1 Detailed Description

Use these functions to set motion parameters (velocity, acceleration, rotation rate, etc.), then execute them. Also provided are functions to stop motion.

Functions

- NDICMDC_API BOOL NdiCmd_Coast (int handle)
  Coast.
- NDICMDC_API BOOL NdiCmd_Brake (int handle)
  Brake.
- NDICMDC_API BOOL NdiCmd_Reset (int handle)
  Reset.
- NDICMDC_API BOOL NdiCmd_SetAcceleration (int handle, int accel)
  Set Acceleration.
- NDICMDC_API BOOL NdiCmd_GetAcceleration (int handle, int *accel)
  Get Acceleration.
- NDICMDC_API BOOL NdiCmd_SetVelocity (int handle, int velocity)
  Set Velocity.
- NDICMDC_API BOOL NdiCmd_GetVelocity (int handle, int *velocity)
  Get Velocity.
- NDICMDC_API BOOL NdiCmd_SetRotation (int handle, int rotation)
  Set Rotation Rate.
- NDICMDC_API BOOL NdiCmd_GetRotation (int handle, int *rotation)
  Get Rotation Rate.
- NDICMDC_API BOOL NdiCmd_SetDistance (int handle, long distance)
  Set Distance.
- NDICMDC_API BOOL NdiCmd_GetDistance (int handle, long *distance)
  Get Distance.

- NDICMDC_API BOOL NdiCmd_SetAngle (int handle, int angle)
  Set Angle.
- NDICMDC_API BOOL NdiCmd_GetAngle (int handle, int *angle)
  Get Angle.
- NDICMDC_API BOOL NdiCmd_SetXDistance (int handle, long xdistance)
  Set Delta Distance.
- NDICMDC_API BOOL NdiCmd_GetXDistance (int handle, long *xdistance)
  Get Delta Distance.
- NDICMDC_API BOOL NdiCmd_SetXAngle (int handle, int xangle)
  Set Delta Angle.
- NDICMDC_API BOOL NdiCmd_GetXAngle (int handle, int *xangle)
  Get Delta Angle.
- NDICMDC_API BOOL NdiCmd_Go (int handle)
  Go.

4.5.2 Function Documentation

4.5.2.1 NDICMDC_API BOOL NdiCmd_Brake (int *handle*)

Brake.

This function actively brakes the servo or DC motor. Current will continue to be consumed.

Parameters:

handle an integer WheelCommander handle

Returns:

4.5.2.2 NDICMDC_API BOOL NdiCmd_Coast (int *handle*)

Coast.

This function stops any current motion, and sets the servo or DC motors into low power mode.

Parameters:

handle an integer WheelCommander handle

Returns:

4.5.2.3 NDICMDC_API BOOL NdiCmd_GetAcceleration (int *handle*, int * *accel*)

Get Acceleration.

This returns the current acceleration value. The value returned will be a scaled version of the value passed to NdiCmd_SetAcceleration().

Parameters:

handle an integer WheelCommander handle

accel a pointer to an integer

Returns:

4.5.2.4 NDICMDC_API BOOL NdiCmd_GetAngle (int *handle*, int * *angle*)

Get Angle.

This returns the current rotation relative to the origin.

Parameters:

handle an integer WheelCommander handle

angle a pointer to an integer returning the angle rotated in degrees

Returns:

4.5.2.5 NDICMDC_API BOOL NdiCmd_GetDistance (int *handle*, long * *distance*)

Get Distance.

This returns the current position.

Parameters:

handle an integer WheelCommander handle

distance a pointer to an integer returning current position

Returns:

4.5.2.6 NDICMDC_API BOOL NdiCmd_GetRotation (int *handle*, int * *rotation*)

Get Rotation Rate.

This returns the measured rate of rotation.

Parameters:

handle an integer WheelCommander handle

rotation a pointer to an integer returning rate of rotation in degrees per second

Returns:

4.5.2.7 NDICMDC_API BOOL NdiCmd_GetVelocity (int *handle*, int * *velocity*)

Get Velocity.

This returns the current measured velocity.

Parameters:

handle an integer WheelCommander handle

velocity a pointer to an integer in units of distance per second

Returns:

4.5.2.8 NDICMDC_API BOOL NdiCmd_GetXAngle (int *handle*, int * *xangle*)

Get Delta Angle.

This returns the current angle relative to the origin, not a relative angle.

Parameters:

handle an integer WheelCommander handle

xangle a pointer to an integer returning the current angle in degrees

Returns:

4.5.2.9 NDICMDC_API BOOL NdiCmd_GetXDistance (int *handle*, long * *xdistance*)

Get Delta Distance.

This returns the current position in the robot's frame of reference, not the relative distance travelled.

Parameters:

handle an integer WheelCommander handle

xdistance a pointer to a long integer, returning the position

Returns:

4.5.2.10 NDICMDC_API BOOL NdiCmd_Go (int *handle*)

Go.

This starts execution of either the velocity control loops, if only NdiCmd_SetVelocity() and/or NdiCmd_SetRotation() were called, or starts the position control loops, if any of the position, angle, or distance commands were called prior to this.

Parameters:

handle an integer WheelCommander handle

Returns:

4.5.2.11 NDICMDC_API BOOL NdiCmd_Reset (int *handle*)

Reset.

This clears all motion parameters (velocity, rotation rate, position, and angle), and stops the servos or DC motors.

Parameters:

handle an integer WheelCommander handle

Returns:

4.5.2.12 NDICMDC_API BOOL NdiCmd_SetAcceleration (int *handle*, int *accel*)

Set Acceleration.

This sets the acceleration value used in NdiCmd_SetPosition(), NdiCmd_-SetDistance(), NdiCmd_SetAngle(), and NdiCmd_SetXAngle().

Parameters:

handle an integer WheelCommander handle

accel an integer in units of distance per second squared

Returns:

4.5.2.13 NDICMDC_API BOOL NdiCmd_SetAngle (int *handle*, int *angle*)

Set Angle.

This sets the goal angle relative to the origin (which is reset to 0 degrees at position 0 on power on or execution of NdiCmd_Reset()).

Parameters:

handle an integer WheelCommander handle

angle an integer in degrees

Returns:

4.5.2.14 NDICMDC_API BOOL NdiCmd_SetDistance (int *handle*, long *distance*)

Set Distance.

This sets the goal position relative to the origin, which is zero at power on or execution of NdiCmd_Reset().

Parameters:

handle an integer WheelCommander handle

distance a long integer setting position in current units of distance

Returns:

4.5.2.15 NDICMDC_API BOOL NdiCmd_SetRotation (int *handle*, int *rotation*)

Set Rotation Rate.

This sets the rotation rate for use during velocity control mode.

Parameters:

handle an integer WheelCommander handle

rotation an integer in units of degrees per second

Returns:

4.5.2.16 NDICMDC_API BOOL NdiCmd_SetVelocity (int *handle*, int *velocity*)

Set Velocity.

This sets the velocity value for either velocity control mode or the maximum velocity for use during a positioning command.

Parameters:

handle an integer WheelCommander handle

velocity an integer in units of distance per second

Returns:

4.5.2.17 NDICMDC_API BOOL NdiCmd_SetXAngle (int *handle*, int *xangle*)

Set Delta Angle.

This sets a goal change in angle relative to the current pose.

Parameters:

handle an integer WheelCommander handle

xangle an integer angle to rotate by, in degrees

Returns:

4.5.2.18 NDICMDC_API BOOL NdiCmd_SetXDistance (int *handle*, long *xdistance*)

Set Delta Distance.

This sets a distance relative to the current position to travel.

Parameters:

handle an integer WheelCommander handle

xdistance a long integer in units of distance

Returns:

4.6 Unicoder Functions

4.6.1 Detailed Description

Use these functions apply to the Unicoder universal encoder only.

Typedefs

- typedef enum _uc_mode_type **UNICODER_MODE**
- typedef enum _uc_encoder_type **ENCODER_TYPE**

Enumerations

- enum _uc_mode_type {

  MD_I2C = 0x01, MD_RS232 = 0x02, MD_ASCII_HEX = 0x04, MD_-SHORT_RET = 0x08,

  MD_INVERT_SERVO = 0x10, MD_FILTER_PERIOD = 0x20 }

  Unicoder Mode Values.

- enum _uc_encoder_type { ET_DEFAULT, ET_QUADRATURE, ET_-STEPDIR, ET_COMMUTATION }

  Encoder Types.

Functions

- NDICMDC_API BOOL NdiCmd_UCSetConstant (int handle, int addr, BYTE const_value)

  Set Constant.

- NDICMDC_API BOOL NdiCmd_UCGetConstant (int handle, int addr, BYTE *const_value)

  Get.

- NDICMDC_API BOOL NdiCmd_SetMode (int handle, BYTE mode)

  Set Mode.

- NDICMDC_API BOOL NdiCmd_GetMode (int handle, BYTE *mode)

  Get Mode.

- NDICMDC_API BOOL NdiCmd_GetEncoderMode (int handle, ENCODER_-TYPE *enc_type, int *num_bits, BOOL *wide_index, BOOL *ccw)

  Get Encoder Mode.

- NDICMDC_API BOOL NdiCmd_SetEncoderMode (int handle, ENCODER_-TYPE enc_type, int num_bits, BOOL wide_index, BOOL ccw)

  Set Encoder Mode.

- NDICMDC_API BOOL NdiCmd_GetZeroPosition (int handle, int *angle)

  Get Zero Position.

- NDICMDC_API BOOL NdiCmd_SetZeroPosition (int handle, int angle)

  Set Zero Position.

- NDICMDC_API BOOL NdiCmd_GetPeriod (int handle, int *period)

  Get Period.

- NDICMDC_API BOOL NdiCmd_GetTime (int handle, int *the_time)

  Get Time.

- NDICMDC_API BOOL NdiCmd_SetTime (int handle, int the_time)

  Set Time.

- NDICMDC_API BOOL NdiCmd_GetServo (int handle, int *speed)

  Get Servo.

- NDICMDC_API BOOL NdiCmd_SetServo (int handle, int speed)

  Set Servo.

- NDICMDC_API BOOL NdiCmd_UCCoast (int handle)

  Unicoder Coast.

- NDICMDC_API BOOL NdiCmd_SetCenterPosition (int handle, int center_-pos)

  Set Servo Center Position.

- NDICMDC_API BOOL NdiCmd_GetCenterPosition (int handle, int *center_-pos)

  Get Servo Center Position.

- NDICMDC_API BOOL NdiCmd_UCStoreParameters (int handle)

  Unicoder Store Parameters.

4.6.2 Enumeration Type Documentation

4.6.2.1 enum _uc_encoder_type

Encoder Types.

These values are passed to NdiCmd_SetEncoderMode().

Enumerator:

ET_DEFAULT Default (quadrature, 1024 transitions per rotation)

ET_QUADRATURE Quadrature (ChA, ChB 90 degrees out of phase)

ET_STEPDIR Step and Direction (sign / magnitude)

ET_COMMUTATION Commutation for brushless DC motors

4.6.2.2 enum _uc_mode_type

Unicoder Mode Values.

These values can be bitwise ORed together to form a mode BYTE.

Enumerator:

MD_I2C readonly; indicates that I2C communications are enabled

MD_RS232 readonly; indicates that RS232 communications are enabled

MD_ASCII_HEX readonly; indicates that ASCII hex parameters and values are used; otherwise, binary values are used

MD_SHORT_RET set for OOPIC compatibility (short return values)

MD_INVERT_SERVO if 1, the servo control pulses will be high for most of the 20ms, and low only for the 0.5 to 2.5 ms portion

MD_FILTER_PERIOD if 1, turns on a digital filter of the period and velocity measurement; averages edge-to-edge timing every 20ms

4.6.3 Function Documentation

4.6.3.1 NDICMDC_API BOOL NdiCmd_GetCenterPosition (int *handle*, int * *center_pos*)

Get Servo Center Position.

This function retrieves the pulsewidth of the servo output pulse's center or zero position. It is 3067 by default, which corresponds to 1.5ms.

Parameters:

handle an integer WheelCommander handle

center_pos a pointer to an integer specifying pulsewidth in multiples of 489ns

Returns:

4.6.3.2 NDICMDC_API BOOL NdiCmd_GetEncoderMode (int *handle*, ENCODER_TYPE * *enc_type*, int * *num_bits*, BOOL * *wide_index*, BOOL * *ccw*)

Get Encoder Mode.

This function queries the Unicoder for its current operating mode. This includes the nature of its incremental output signals, the number of bits of resolution, the size of the index pulse, and the direction of rotation.

Parameters:

handle an integer Unicoder handle

enc_type a member of the ENCODER_TYPE enumeration

num_bits an integer representing the resolution

wide_index a BOOL indicating that the index pulse is 3 wide rather than 1

ccw a BOOL indicating that the angle increases in a counter-clockwise direction when true

Returns:

4.6.3.3 NDICMDC_API BOOL NdiCmd_GetMode (int *handle*, BYTE * *mode*)

Get Mode.

This function retrieves the current operating mode of the Unicoder encoder.

Parameters:

handle an integer Unicoder handle

mode a BYTE containing an ORing of UNICODER_MODE values, containing the mode returned

Returns:

4.6.3.4 NDICMDC_API BOOL NdiCmd_GetPeriod (int *handle*, int * *period*)

Get Period.

This function returns the elapsed time between the previous two transitions of the encoder's incremental signals. The unit of time is the Unicoder's own internal timebase, nominally in multiples of 489ns (+/- 2%).

Parameters:

handle an integer Unicoder handle

period an integer indicating the elapsed time between encoder transitions

Returns:

4.6.3.5 NDICMDC_API BOOL NdiCmd_GetServo (int *handle*, int * *speed*)

Get Servo.

This function returns the current pulsewidth deviation, or 0x8000 if RC servo pulses are currently turned off.

Parameters:

handle an integer WheelCommander handle

speed an integer specifying pulsewidth delta from 1.5ms

Returns:

4.6.3.6 NDICMDC_API BOOL NdiCmd_GetTime (int *handle*, int * *the_time*)

Get Time.

This function returns the current system time, in multiples of 489ns. This is useful to calibrate velocity and other time-dependent measurements, using the main robot controller's more accurate crystal-controlled clock. The clock inside the Unicoder encoder is accurate to +/- 2%, so velocity and period measurements are likewise limited in accuracy.

Parameters:

handle an integer Unicoder handle

the_time an integer indicating the system time in multiples of 489ns.

Returns:

4.6.3.7 NDICMDC_API BOOL NdiCmd_GetZeroPosition (int *handle*, int * *angle*)

Get Zero Position.

This function queries the Unicoder for the angle relative to the factory default at which the index pulse and user angle origin begin.

Parameters:

handle an integer Unicoder handle

angle an integer representing the number of encoder positions from 0 to 1023 (0 is the default)

Returns:

4.6.3.8 NDICMDC_API BOOL NdiCmd_SetCenterPosition (int *handle*, int *center_pos*)

Set Servo Center Position.

This function sets the pulsewidth of the servo output pulse's center or zero position. It is 3067 by default, which corresponds to 1.5ms.

Parameters:

handle an integer WheelCommander handle

center_pos an integer specifying pulsewidth in multiples of 489ns

Returns:

4.6.3.9 NDICMDC_API BOOL NdiCmd_SetEncoderMode (int *handle*, ENCODER_TYPE *enc_type*, int *num_bits*, BOOL *wide_index*, BOOL *ccw*)

Set Encoder Mode.

This function allows you to change the Unicoder's current operating mode. This includes the nature of its incremental output signals, the number of bits of resolution, the size of the index pulse, and the direction of rotation.

Parameters:

handle an integer Unicoder handle

enc_type a member of the ENCODER_TYPE enumeration

num_bits an integer representing the resolution

wide_index a BOOL indicating that the index pulse is 3 wide rather than 1

ccw a BOOL indicating that the angle increases in a counter-clockwise direction when true

Returns:

4.6.3.10 NDICMDC_API BOOL NdiCmd_SetMode (int *handle*, BYTE *mode*)

Set Mode.

This function allows you to change the current operating mode.

Parameters:

handle an integer Unicoder handle

mode a BYTE containing an ORing of UNICODER_MODE values, containing the mode to be set

Returns:

4.6.3.11 NDICMDC_API BOOL NdiCmd_SetServo (int *handle*, int *speed*)

Set Servo.

This function enables production of an RC servo control pulse on TP2, with the specified deviation in multiples of 489ns from the center pulsewidth of 1.5ms. In this mode,

the encoder should be set to sign/magnitude mode, as the ChB/DIR line is shared with TP2 through a 1K resistor, and is sampled only on edges of the servo control pulse. Useful values are in the range of +/- 2048, corresponding to +/- 1ms.

Parameters:

handle an integer WheelCommander handle

speed an integer specifying pulsewidth delta from 1.5ms

Returns:

4.6.3.12 NDICMDC_API BOOL NdiCmd_SetTime (int *handle*, int *the_time*)

Set Time.

This function allows you to change the current system time, in multiples of 489ns.

Parameters:

handle an integer Unicoder handle

the_time an integer indicating the system time in multiples of 489ns.

Returns:

4.6.3.13 NDICMDC_API BOOL NdiCmd_SetZeroPosition (int *handle*, int *angle*)

Set Zero Position.

This function allows you to change the angle of the Unicoder's index pulse and the amount to shift the user-accessible angle by.

Parameters:

handle an integer Unicoder handle

angle an integer representing the number of encoder positions from 0 to 1023 (0 is the default)

Returns:

4.6.3.14 NDICMDC_API BOOL NdiCmd_UCCoast (int *handle*)

Unicoder Coast.

This function stops any current motion, and sets the servo into low power mode. It is now safe to use modes other than sign/magnitude.

Parameters:

handle an integer WheelCommander handle

Returns:

4.6.3.15 NDICMDC_API BOOL NdiCmd_UCGetConstant (int *handle*, int *addr*, BYTE * *const_value*)

Get.

This function retrieves a byte from the flash constants area.

Parameters:

handle an integer Unicoder handle

addr an integer address in the flash

const_value a pointer to a BYTE, containing the value read

Returns:

4.6.3.16 NDICMDC_API BOOL NdiCmd_UCSetConstant (int *handle*, int *addr*, BYTE *const_value*)

Set Constant.

This function lets you directly access the constants stored in Flash in the Unicoder. This is for advanced users only. The functions NdiCmd_GetMode(), NdiCmd_GetI2CAddr(), and so forth, are easier to use.

Parameters:

handle an integer Unicoder handle

addr an integer indicating the address in the flash constants area

const_value a BYTE indicating the value to write to addr

Returns:

4.6.3.17 NDICMDC_API BOOL NdiCmd_UCStoreParameters (int *handle*)

Unicoder Store Parameters.

This function writes any user-modifiable constants to flash.

Parameters:

handle an integer WheelCommander handle

Returns:

Chapter 5

Data Structure Documentation

5.1 _int_union_ Union Reference

`#include <NdiCmdC.h>`

5.1.1 Detailed Description

INT FLAGS.

These are used to control or report the state of the user interrupt system.

Data Fields

- struct _int_union_::_int_flags flags
- BYTE data

Data Structures

- struct **_int_flags**

5.1.2 Field Documentation

5.1.2.1 struct _int_union_::_int_flags _int_union_::flags

this is a bitfield structure

5.1.2.2 BYTE _int_union_::data

this is a BYTE for accessing all flags at once

5.2 _io_flags_union Union Reference

`#include <NdiCmdC.h>`

5.2.1 Detailed Description

IO FLAGS.

These are used to control or report the state of the analog or digital input monitoring system.

Data Fields

- struct _io_flags_union::_io_flags flags
- BYTE data

Data Structures

- struct **_io_flags**

5.2.2 Field Documentation

5.2.2.1 struct _io_flags_union::_io_flags _io_flags_union::flags

this is a bitfield structure

5.2.2.2 BYTE _io_flags_union::data

this is a BYTE for accessing all flags at once

Chapter 6

File Documentation

6.1 NdiCmdC.h File Reference

6.1.1 Detailed Description

Noetic DLL

Data Structures

- union _int_union_
  INT FLAGS.
- struct **_int_union_::_int_flags**
- union _io_flags_union
  IO FLAGS.
- struct **_io_flags_union::_io_flags**

Defines

- #define NDICMDC_API __declspec(dllimport)
  Declare functions as importable DLL functions.

Typedefs

- typedef enum _wc_baud **WC_BAUD**

- typedef enum _wc_pwm_frequency **WC_PWMFREQ**
- typedef union _int_union_ **INT_FLAGS**
- typedef union _io_flags_union **IO_FLAGS**
- typedef enum _uc_mode_type **UNICODER_MODE**
- typedef enum _uc_encoder_type **ENCODER_TYPE**

Enumerations

- enum _wc_baud {

  BD_1200, BD_4800, BD_9600, BD_19200,

  BD_38400, BD_57600, BD_115200, BD_230400,

  BD_460800 }

  Baud Rate Values.

- enum _wc_pwm_frequency {

  PF_2KHZ, PF_4KHZ, PF_8KHZ, PF_16KHZ,

  PF_32KHZ, PF_64KHZ }

  PWM frequencies.

- enum _uc_mode_type {

  MD_I2C = 0x01, MD_RS232 = 0x02, MD_ASCII_HEX = 0x04, MD_SHORT_RET = 0x08,

  MD_INVERT_SERVO = 0x10, MD_FILTER_PERIOD = 0x20 }

  Unicoder Mode Values.

- enum _uc_encoder_type { ET_DEFAULT, ET_QUADRATURE, ET_STEPDIR, ET_COMMUTATION }

  Encoder Types.

Functions

- NDICMDC_API void NdiCmd_Init (void)

  Initialize the DLL.

- NDICMDC_API void NdiCmd_SetLogWindow (HWND wnd)

  Specify Output Control for Logging.

- NDICMDC_API void NdiCmd_SetLogFile (char *path, BOOL append)

  Specify Output File for Logging.

- NDICMDC_API void NdiCmd_LogIt (char *text)
  Send Text to the Log.
- NDICMDC_API void NdiCmd_LogQuiet (BOOL quiet)
  Set Logging Level.
- NDICMDC_API void NdiCmd_EnableDirect (void)
  Enable Direct.
- NDICMDC_API BOOL NdiCmd_Search (void)
  Search for Ndi Devices.
- NDICMDC_API BOOL NdiCmd_CheckPort (int i)
  Check Port.
- NDICMDC_API int NdiCmd_GetFirstPort (void)
  Get First Port.
- NDICMDC_API int NdiCmd_GetNextPort (int prev)
  Get Next Port.
- NDICMDC_API void NdiCmd_SetPortEnabled (int i, BOOL enabled)
  Set Port Enabled.
- NDICMDC_API BOOL NdiCmd_GetPortEnabled (int i)
  Get Port Enabled.
- NDICMDC_API void NdiCmd_CloseAllPorts (void)
  Close All Ports.
- NDICMDC_API int NdiCmd_GetWheelCommander (void)
  Get Wheel Commander.
- NDICMDC_API int NdiCmd_GetNumWheelCommanders (void)
  Get Num Wheel Commanders.
- NDICMDC_API int NdiCmd_GetWheelCommanderNum (int num)
  Get Wheel Commander Num.
- NDICMDC_API int NdiCmd_GetUnicoder (void)
  Get Unicoder.

- NDICMDC_API int NdiCmd_GetNumUnicoders (void)

  Get Num Unicoders.

- NDICMDC_API int NdiCmd_GetUnicoderNum (int index)

  Get Unicoder Num.

- NDICMDC_API void NdiCmd_RunWizard (BOOL wiz_mode)

  Run Wizard.

- NDICMDC_API BOOL NdiCmd_UpdateFirmware (char *filename, HWND statwindow)

  Update Firmware.

- NDICMDC_API BOOL NdiCmd_UpdateFirmwareEx (char *filename, HWND statwindow, BOOL special, BOOL direct)

  Update Firmware Ex.

- NDICMDC_API void NdiCmd_SetStatusWindow (HWND hwnd)

  Set Status Window.

- NDICMDC_API BOOL NdiCmd_GetPortInfo (int handle, char *type_string, int *id)

  Get Port Info.

- NDICMDC_API BOOL NdiCmd_Sync (int handle)

  Synchronize.

- NDICMDC_API BOOL NdiCmd_CommTest (int handle, int first, int last, int step)

  Comm Test.

- NDICMDC_API BOOL NdiCmd_GetName (int handle, LPSTR name, int *version)

  Get Name.

- NDICMDC_API BOOL NdiCmd_Status (int handle, unsigned int *status)

  Status.

- NDICMDC_API BOOL NdiCmd_SetBaud (int handle, WC_BAUD baud)

  Set Baud.

- NDICMDC_API BOOL NdiCmd_GetBaud (int handle, int *baud)

Get Baud.

- NDICMDC_API void NdiCmd_SetBlueRadio (BOOL enable, BOOL fast)

  Set Blue Radio.

- NDICMDC_API void NdiCmd_GetBlueRadio (BOOL *enable, BOOL *fast)

  Get Blue Radio.

- NDICMDC_API BOOL NdiCmd_SetI2CAddress (int handle, int addr)

  Set I2C Address.

- NDICMDC_API BOOL NdiCmd_GetI2CAddress (int handle, BYTE *addr)

  Get I2C Address.

- NDICMDC_API BOOL NdiCmd_SetCommMode (int handle, BOOL slow_comms, BOOL notify, BOOL short_mode)

  Set Comm Mode.

- NDICMDC_API BOOL NdiCmd_SetPlatformSpecs (int handle, BOOL polarity, BOOL counter_axis, BYTE wheel_base, BYTE wheel_diameter)

  Set Platform Specs.

- NDICMDC_API BOOL NdiCmd_SetMotorMode (int handle, BOOL pwm, BOOL locked_antiphase, BOOL dual_dir_lines, WC_PWMFREQ pwm_frequency)

  Set Motor Mode.

- NDICMDC_API BOOL NdiCmd_GetEncoderProperties (int handle, BOOL *quadrature, WORD *ticks_per_rotation)

  Get Encoder Properties.

- NDICMDC_API BOOL NdiCmd_SetEncoderProperties (int handle, BOOL quadrature, WORD ticks_per_rotation)

  Set Encoder Properties.

- NDICMDC_API BOOL NdiCmd_ReadDIO (int handle, int pin, BOOL *value)

  Read Digital Input.

- NDICMDC_API BOOL NdiCmd_WriteDIO (int handle, int pin, BOOL value)

  Write Digital Output.

- NDICMDC_API BOOL NdiCmd_ReadAIO (int handle, int pin, double *value)

  Read Analog Input.

- NDICMDC_API BOOL NdiCmd_SetInterrupt (int handle, INT_FLAGS flags)

  Set Interrupt.

- NDICMDC_API BOOL NdiCmd_GetInterruptStatus (int handle, INT_FLAGS *flags, WORD *io_lines)

  Get Interrupt Status.

- NDICMDC_API BOOL NdiCmd_SetDIOMonitorInfo (int handle, int pin, IO_-FLAGS flags)

  Set DIO Monitor Info.

- NDICMDC_API BOOL NdiCmd_SetAIOMonitorInfo (int handle, int pin, IO_-FLAGS flags, WORD lo, WORD hi)

  Set AIO Monitor Info.

- NDICMDC_API BOOL NdiCmd_Coast (int handle)

  Coast.

- NDICMDC_API BOOL NdiCmd_Brake (int handle)

  Brake.

- NDICMDC_API BOOL NdiCmd_Reset (int handle)

  Reset.

- NDICMDC_API BOOL NdiCmd_SetAcceleration (int handle, int accel)

  Set Acceleration.

- NDICMDC_API BOOL NdiCmd_GetAcceleration (int handle, int *accel)

  Get Acceleration.

- NDICMDC_API BOOL NdiCmd_SetVelocity (int handle, int velocity)

  Set Velocity.

- NDICMDC_API BOOL NdiCmd_GetVelocity (int handle, int *velocity)

  Get Velocity.

- NDICMDC_API BOOL NdiCmd_SetRotation (int handle, int rotation)

  Set Rotation Rate.

- NDICMDC_API BOOL NdiCmd_GetRotation (int handle, int *rotation)
  Get Rotation Rate.
- NDICMDC_API BOOL NdiCmd_SetDistance (int handle, long distance)
  Set Distance.
- NDICMDC_API BOOL NdiCmd_GetDistance (int handle, long *distance)
  Get Distance.
- NDICMDC_API BOOL NdiCmd_SetAngle (int handle, int angle)
  Set Angle.
- NDICMDC_API BOOL NdiCmd_GetAngle (int handle, int *angle)
  Get Angle.
- NDICMDC_API BOOL NdiCmd_SetXDistance (int handle, long xdistance)
  Set Delta Distance.
- NDICMDC_API BOOL NdiCmd_GetXDistance (int handle, long *xdistance)
  Get Delta Distance.
- NDICMDC_API BOOL NdiCmd_SetXAngle (int handle, int xangle)
  Set Delta Angle.
- NDICMDC_API BOOL NdiCmd_GetXAngle (int handle, int *xangle)
  Get Delta Angle.
- NDICMDC_API BOOL NdiCmd_Go (int handle)
  Go.
- NDICMDC_API BOOL NdiCmd_UCSetConstant (int handle, int addr, BYTE const_value)
  Set Constant.
- NDICMDC_API BOOL NdiCmd_UCGetConstant (int handle, int addr, BYTE *const_value)
  Get.
- NDICMDC_API BOOL NdiCmd_SetMode (int handle, BYTE mode)
  Set Mode.
- NDICMDC_API BOOL NdiCmd_GetMode (int handle, BYTE *mode)

Get Mode.

- NDICMDC_API BOOL NdiCmd_GetEncoderMode (int handle, ENCODER_-TYPE *enc_type, int *num_bits, BOOL *wide_index, BOOL *ccw)

  Get Encoder Mode.

- NDICMDC_API BOOL NdiCmd_SetEncoderMode (int handle, ENCODER_-TYPE enc_type, int num_bits, BOOL wide_index, BOOL ccw)

  Set Encoder Mode.

- NDICMDC_API BOOL NdiCmd_GetZeroPosition (int handle, int *angle)

  Get Zero Position.

- NDICMDC_API BOOL NdiCmd_SetZeroPosition (int handle, int angle)

  Set Zero Position.

- NDICMDC_API BOOL NdiCmd_GetPeriod (int handle, int *period)

  Get Period.

- NDICMDC_API BOOL NdiCmd_GetTime (int handle, int *the_time)

  Get Time.

- NDICMDC_API BOOL NdiCmd_SetTime (int handle, int the_time)

  Set Time.

- NDICMDC_API BOOL NdiCmd_GetServo (int handle, int *speed)

  Get Servo.

- NDICMDC_API BOOL NdiCmd_SetServo (int handle, int speed)

  Set Servo.

- NDICMDC_API BOOL NdiCmd_UCCoast (int handle)

  Unicoder Coast.

- NDICMDC_API BOOL NdiCmd_SetCenterPosition (int handle, int center_-pos)

  Set Servo Center Position.

- NDICMDC_API BOOL NdiCmd_GetCenterPosition (int handle, int *center_-pos)

  Get Servo Center Position.

- NDICMDC_API BOOL NdiCmd_UCStoreParameters (int handle)

Unicoder Store Parameters.