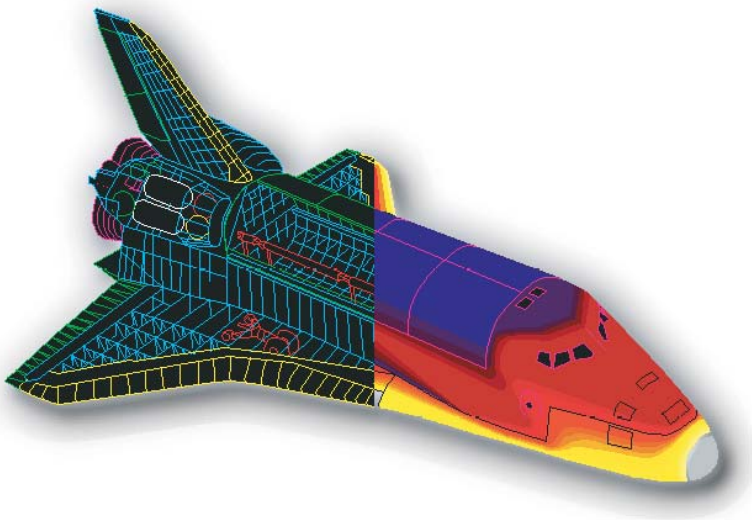
DAQmx for HTBasic

Professional National Instruments DAQ Driver

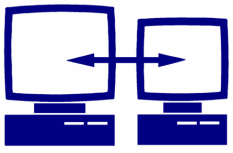
User's Guide



Tech Soft

DAQmx for HTBasic

User's Guide



Tech Soft

www.TechSoft.de

8th revision, January 2011

Manual part number: H-TN0010

Table of contents

File version: 21-Jan-2011 (SH)

DAQmx Library for HTBasic

System requirements:
TransEra HTBasic 9.X
Windows 2000, Windows XP, Windows Vista
Hardware: Any National Instruments DAQ board which is supported by the NI-DAQmx driver (and simulated devices)
National Instruments system software "NI-DAQmx" version 8.0 or higher (tested up to NI-DAQmx 9.1)
Strongly recommended: NI-DAQmx 8.6.1

Concept

HTBasic includes a National Instruments driver for older DAQ boards which uses the so called "Traditional NI-DAQ (Legacy)" driver model. This HTBasic driver is called "DAQNI".
However, many newer DAQ boards are no longer supported by this older driver model. Some time ago National Instruments introduced a new driver for all newer DAQ hardware called "NI-DAQmx. This driver simplifies the programming of the DAQ hardware significantly and provides a lot of new functionality.

The DAQmx Library for HTBasic allows you to use all the features of the new NI-DAQmx driver model under HTBasic including powerful functions like callback services which uses the built-in intelligence of the NI DAQ hardware allowing you to run multiple simultaneous measurements in the background (if the hardware supports it). The HTBasic library supports almost the entire function set of the NI-DAQmx driver (currently more than 200 functions) and provides a programming interface very similar to the ANSI C interface of the NI-DAQmx driver. In addition to the normal analog and digital input/output and counter functions the new driver also supports switches, special measurements like microphone measures, accelerator measures, special signal triggering, watchdog services, task management and much more (the hardware must support the function, e.g. not all National Instruments hardware includes a watchdog).

It is important that you understand the key concepts of the National Instruments NI-DAQmx driver before you start to use the DAQmx Library for HTBasic. Please read the National Instruments documentation "NI-DAQmx Help" and "NI-DAQmx C Reference Help" carefully.

Installation

Install hardware and software from National Instruments as described in your National Instruments product manual. Note that you do not have to use physical hardware in order to test this library, it works with simulated devices too. Please refer to your MAX (Measurement & Automation Explorer) documentation to find out how to set up simulated devices in MAX.
After you have installed the National Instruments software and (if desired) hardware first test your devices in MAX (Measurement & Automation Explorer) using the test panel.
If the test finished successfully you can install the "DAQmx Library for HTBasic" by running the file `DAQmx_htb_setup.exe`. Follow the instructions on the screen. After sucessful installation start HTBasic and load the desired example program from the "File" menu (see below). Make sure that you have enabled the option "Change MSI on Open".

Example programs

Many examples programs can be found in the subdirectory "HTB Examples" in the installation directory of your DAQmx for HTBasic Library. One of those examples is called "`combined_daqmx_example.prg`" which demonstrates the following basic functions:

- Analog output at port `ao0`
- Fast analog voltage input at port `ai0` and current input at Port `ai1`
- Digital output on bits 0..3 of digital port 0 (`port0`)
- Digital input on bits 0..3 of digital port 0 (`port1`)
- Counter I/O with counter 0 (`ctr0`)
- Optional: Usage of two stored tasks (`VoltageTask`, `DigitalInTask`)

You might connect the analog output port 0 (ao0) with analog input port 0 (ai0) in order to read back the analog output values in to your analog input port. Also, you might want to connect bits 0..3 of digital port 0 with bits 4..7 of digital port 1 for testing purposes.

Please note that the demo version only supports 2 simultaneous physical tasks (but it supports a virtually unlimited number of tasks using simulated devices).

If you have questions or problems please report them to:

Tech Soft GmbH
Test & Measurement Division
Karmeliterweg 114
13465 Berlin, Germany

Email: HTB@TechSoft.de http://www.TechSoft.de

Introduction

The DAQmx Library for HTBasic was developed to use new DAQ interfaces like the PCI-6259 from National instruments which are no longer supported by the current HTBasic DAQNI driver.

This library uses the Windows driver from National Instruments "NI-DAQmx" (`nicaiu.dll`). Due to the fact that the new driver model uses data types which are not available under HTBasic we developed a "wrapper DLL" which sits between the National Instruments driver and HTBasic.

The HTBasic driver consists of two files:

```
daqmx_htb.dll     Driver DLL ("Wrapper DLL")
daqmx_htb.lib     HTBasic library
shfolder.dll      Helper functions to determine special Windows directories
```

Files after installation

```
HTB Examples\*.*                Many HTBasic example programs
Driver\*.*                      Driver DLLs and HTBasic library
Doc\DAQmx_htb_manual.pdf        This document
Source\*.*                      Source files for wrapper DLL for Visual C++ 6.0 *)
```

*) Complete VC++ source code is only available in the special "Developer Version".

Library usage in customer programs

In order to simplify the usage of the the function in customer programs we have developed several support functions. At first you should load the function `FNHx_manage_subs` from the main HTBasic library (make sure that your MSI points to the directory where the file `"daqmx_htb.lib"` is located):

```
IF NOT INMEM("FNHx_manage_subs") THEN LOADSUB FNHx_manage_subs FROM "daqmx_htb.lib"
```

After that call the loaded function and check the return value:

```
IF NOT FNHx_manage_subs("LOAD",Verbose) THEN STOP  ! Load & initialize library
```

If the return value is 0 then the function failed for some reason. In this case the program will be stopped.
If the function returned 1 all went fine and you can start to use the function library. In the example program there is a label called `User_routines:` where several user routines are called (analog I/O, digital I/O etc.).

After all functions are finished you should use the following lines at the very end of your program:

```
Dummy=FNHx_manage_subs("UNLOAD",Verbose)     ! Unload and de-initialize library
DELSUB FNHx_manage_subs
```

Getting started

We recommend you to go through the source code of the example program in order to see how the library works and how to use it. It is important that you understand how the original functions of the National Instruments DAQmx library work, so always look into the documentation for this library since the HTBasic library is based on the original National Instruments documentation.
If you want to write your own applications a good starting point is the framework "`Program_framework.prg`". Make a copy of this file and use this as a starting point for your own programs. Go to label `User_routines:` and include your own code there.

Device info

Use the function `FNHx_get_dev_info` in order to print some device specific information on the screen including device names, serial number, driver versions etc. This function is used in the example program and the application framework program too. When you set the variable `Write_2_file` to 1 the function writes your current configuration into the file `daqmx_devinfo.txt` stored in the directory `[%APPDATA%]\HTB-DAQmx`. On English Windows systems this directory usually translates into `"c:\Documents and Settings\Username\Application Data"`.

Function Reference: Overview

Function names under HTBasic are usually shorter than the original NI-DAQmx functions due to the fact that HTBasic only supports function names with up to 15 characters. All functions use `_cdecl` calling convention.
Functions without a specific NI-DAQmx version number in the following table can be used under NI-DAQmx 8.0.
Some functions are only available under certain NI-DAQmx versions. Those functions require at least the NI-DAQmx version number installed on your system.

NI-DAQmx Function Overview

NI-DAQmx function name *)	HTBasic function name	HTBasic Ret.type	NI-DAQmx Version
1. Task Configuration and Control			
DAQmxLoadTask	FNHx_loadtask	LONG	
DAQmxCreateTask	FNHx_createtask	LONG	
DAQmxStartTask	FNHx_starttask	LONG	
DAQmxStopTask	FNHx_stoptask	LONG	
DAQmxClearTask	FNHx_cleartask	LONG	
DAQmxWaitUntilTaskDone	FNHx_waituntaskdn	LONG	
DAQmxIsTaskDone	FNHx_istaskdone	LONG	
DAQmxTaskControl	FNHx_taskcontrol	LONG	
DAQmxGetNthTaskChannel	FNHx_getnthtaskch	LONG	
2. Channel Configuration/Creation			
DAQmxCreateAIVoltageChan	FNHx_craivoltchan	LONG	
DAQmxCreateAIVoltageRMSChan	FNHx_craivlrmschn	LONG	v8.6
DAQmxCreateAICurrentChan	FNHx_craicurrchan	LONG	
DAQmxCreateAICurrentRMSChan	FNHx_craicurmschn	LONG	v8.6
DAQmxCreateAIThrmcplChan	FNHx_craithcpchan	LONG	
DAQmxCreateAIRTDChan	FNHx_crairtdchan	LONG	
DAQmxCreateAIThrmstrChanIex	FNHx_craithmchiex	LONG	
DAQmxCreateAIThrmstrChanVex	FNHx_craithmchvex	LONG	
DAQmxCreateAIFreqVoltageChan	FNHx_craifrvochan	LONG	
DAQmxCreateAIResistanceChan	FNHx_crairesichan	LONG	
DAQmxCreateAIStrainGageChan	FNHx_craistrgachn	LONG	
DAQmxCreateAIVoltageChanWithExcit	FNHx_craivochwexc	LONG	
DAQmxCreateAITempBuiltInSensorChan	FNHx_craitmpbschn	LONG	
DAQmxCreateAIAccelChan	FNHx_craiaccelchn	LONG	
DAQmxCreateAIMicrophoneChan	FNHx_craimicrchan	LONG	
DAQmxCreateAIPosLVDTChan	FNHx_craipslvdtch	LONG	
DAQmxCreateAIPosRVDTChan	FNHx_craipsrvdtch	LONG	
DAQmxCreateTEDSAIVoltageChan	FNHx_crtaivoltchn	LONG	
DAQmxCreateTEDSAICurrentChan	FNHx_crtaicurrchn	LONG	
DAQmxCreateTEDSAIThrmcplChan	FNHx_crtaithcpchn	LONG	
DAQmxCreateTEDSAIRTDChan	FNHx_crtairtdchn	LONG	
DAQmxCreateTEDSAIThrmstrChanIex	FNHx_crtaithchiex	LONG	
DAQmxCreateTEDSAIThrmstrChanVex	FNHx_crtaithchvex	LONG	
DAQmxCreateTEDSAIResistanceChan	FNHx_crtairesichn	LONG	
DAQmxCreateTEDSAIStrainGageChan	FNHx_crtaistrgchn	LONG	
DAQmxCreateTEDSAIVoltageChanWithExcit	FNHx_crtaivochwex	LONG	
DAQmxCreateTEDSAIAccelChan	FNHx_craitacclchn	LONG	
DAQmxCreateAIMicrophoneChan	FNHx_craimicrchan	LONG	
DAQmxCreateAIPosLVDTChan	FNHx_craipslvdtch	LONG	
DAQmxCreateAIPosRVDTChan	FNHx_craipsrvdtch	LONG	
DAQmxCreateAOVoltageChan	FNHx_craovoltchan	LONG	
DAQmxCreateAOCurrentChan	FNHx_craocurrchan	LONG	
DAQmxCreateAOFuncGenChan	FNHx_craofungnchn	LONG	v8.6
DAQmxCreateDIChan	FNHx_createdichan	LONG	
DAQmxCreateDOChan	FNHx_createdochan	LONG	
DAQmxCreateCIFreqChan	FNHx_crcifreqchan	LONG	
DAQmxCreateCIPeriodChan	FNHx_crciperichan	LONG	
DAQmxCreateCICountEdgesChan	FNHx_crcicntedgch	LONG	
DAQmxCreateCIPulseWidthChan	FNHx_crciplswchan	LONG	
DAQmxCreateCISemiPeriodChan	FNHx_crcisemperch	LONG	
DAQmxCreateCITwoEdgeSepChan	FNHx_crcitwedsech	LONG	
DAQmxCreateCILinEncoderChan	FNHx_crcilinencch	LONG	
DAQmxCreateCIAngEncoderChan	FNHx_crcianencchn	LONG	
DAQmxCreateCIGPSTimestampChan	FNHx_crcigpstschn	LONG	
DAQmxCreateCOPulseChanFreq	FNHx_crcopulchfrq	LONG	
DAQmxCreateCOPulseChanTime	FNHx_crcopulchtim	LONG	
DAQmxCreateCOPulseChanTicks	FNHx_crcopulchtck	LONG	

*) As described in the "NI-DAQmx C Reference Help"

NI-DAQmx function name *)	HTBasic function name	HTBasic Ret.type	NI-DAQmx Version
3. Timing Functions			
DAQmxCfgSampClkTiming	FNHx_cfgsamclktim	LONG	
DAQmxCfgPipelinedSampClkTiming	FNHx_cfgplsclktim	LONG	v8.5
DAQmxCfgHandshakingTiming	FNHx_cfghshtiming	LONG	
DAQmxCfgBurstHandshakingTimingImportClock	FNHx_cfgbhtimiclk	LONG	
DAQmxCfgBurstHandshakingTimingExportClock	FNHx_cfgbhtimeclk	LONG	
DAQmxCfgChangeDetectionTiming	FNHx_cfgchgdettim	LONG	
DAQmxCfgImplicitTiming	FNHx_cfgimptiming	LONG	
4. Read Functions			
DAQmxReadAnalogF64	FNHx_readanalgf64	LONG	
DAQmxReadAnalogScalarF64	FNHx_readanalsf64	LONG	
DAQmxReadBinaryI16	FNHx_rdbinaryi16	LONG	
DAQmxReadBinaryI32	FNHx_rdbinaryi32	LONG	
DAQmxReadDigitalU16	FNHx_readdigitu16	LONG	
DAQmxReadDigitalU32	FNHx_readdigitu32	LONG	
DAQmxReadDigitalScalarU32	FNHx_readdigits32	LONG	
DAQmxReadDigitalLines	FNHx_readdiglines	LONG	
DAQmxReadCounterF64	FNHx_readcntrf64	LONG	
DAQmxReadCounterU32	FNHx_readcntru32	LONG	
DAQmxReadCounterScalarU32	FNHx_rdcntscalu32	LONG	
DAQmxReadCounterScalarF64	FNHx_rdcntscalf64	LONG	
5. Write Functions			
DAQmxWriteAnalogF64	FNHx_writeanalf64	LONG	
DAQmxWriteAnalogScalarF64	FNHx_writeanasf64	LONG	
DAQmxWriteBinaryI16	FNHx_writebini16	LONG	
DAQmxWriteBinaryI32	FNHx_writebini32	LONG	
DAQmxWriteDigitalU16	FNHx_writdigitu16	LONG	
DAQmxWriteDigitalU32	FNHx_writdigitu32	LONG	
DAQmxWriteDigitalScalarU32	FNHx_writdigscu32	LONG	
DAQmxWriteDigitalLines	FNHx_writdiglines	LONG	
DAQmxWriteCtrFreq	FNHx_wrctrfreq	LONG	
DAQmxWriteCtrFreqScalar	FNHx_wrctrfrqscal	LONG	
DAQmxWriteCtrTime	FNHx_wrctrtime	LONG	
DAQmxWriteCtrTimeScalar	FNHx_wrctrtimscal	LONG	
DAQmxWriteCtrTicks	FNHx_wrctrticks	LONG	
DAQmxWriteCtrTicksScalar	FNHx_wrctrtickssc	LONG	
6. Triggering			
DAQmxDisableStartTrig	FNHx_disstarttrig	LONG	
DAQmxCfgDigEdgeStartTrig	FNHx_cfdgedsttrig	LONG	
DAQmxCfgAnlgEdgeStartTrig	FNHx_cfganedstrig	LONG	
DAQmxCfgAnlgWindowStartTrig	FNHx_cfganwistrig	LONG	
DAQmxCfgDigPatternStartTrig	FNHx_cfgdiptstrig	LONG	
DAQmxDisableRefTrig	FNHx_disreftrig	LONG	
DAQmxCfgDigEdgeRefTrig	FNHx_cfgdiedrtrig	LONG	
DAQmxCfgAnlgEdgeRefTrig	FNHx_cfganedrtrig	LONG	
DAQmxCfgAnlgWindowRefTrig	FNHx_cfganwirtrig	LONG	
DAQmxCfgDigPatternRefTrig	FNHx_cfgdipartrig	LONG	
DAQmxDisableAdvTrig	FNHx_disadvtrig	LONG	
DAQmxCfgDigEdgeAdvTrig	FNHx_cfgdiedatrig	LONG	
DAQmxSendSoftwareTrigger	FNHx_sendsofttrig	LONG	
7. Calibration			
DAQmxSelfCal	FNHx_selfcal	LONG	
DAQmxPerformBridgeOffsetNullingCal	FNHx_perfbrofnlca	LONG	
DAQmxGetCalDevTemp	FNHx_getcadevtemp	LONG	
DAQmxGetSelfCalLastTemp	FNHx_getsfcaltemp	LONG	
DAQmxGetExtCalLastTemp	FNHx_getexcaltemp	LONG	
DAQmxESeriesCalAdjust	FNHx_esercaladjst	LONG	
DAQmxMSeriesCalAdjust	FNHx_msercaladjst	LONG	
DAQmxSSeriesCalAdjust	FNHx_ssercaladjst	LONG	
DAQmxSCBaseboardCalAdjust	FNHx_scbbcaladjst	LONG	
DAQmxAOSeriesCalAdjust	FNHx_aosecaladjst	LONG	
DAQmxDeviceSupportsCal	FNHx_devsuppcal	LONG	
DAQmxGetSelfCalSupported	FNHx_selfcalsupp	LONG	
DAQmxInitExtCal	FNHx_initextcal	LONG	
DAQmxCloseExtCal	FNHx_closeextcal	LONG	
DAQmxGetSelfCalLastDateAndTime	FNHx_getscaldtati	LONG	
DAQmxGetExtCalLastDateAndTime	FNHx_getecaldtati	LONG	
DAQmxRestoreLastExtCalConst	FNHx_reslexcalcon	LONG	
DAQmxChangeExtCalPassword	FNHx_chnextcalpwd	LONG	
DAQmxGetExtCalRecommendedInterval	FNHx_getecalrecin	LONG	

NI-DAQmx function name *)	HTBasic function name	HTBasic Ret.type	NI-DAQmx Version
7. Calibration (Continued)			
DAQmxAdjustDSAAICal	FNHx_adjdsaaical	LONG	
DAQmxAdjustDSAAOCal	FNHx_adjdsaaocal	LONG	
DAQmxAdjustDSATimebaseCal	FNHx_adjdsatbcal	LONG	
DAQmxAdjust4204Cal	FNHx_adj4204cal	LONG	
DAQmxAdjust4220Cal	FNHx_adj4220cal	LONG	
DAQmxAdjust4224Cal	FNHx_adj4224cal	LONG	
DAQmxAdjust4225Cal	FNHx_adj4225cal	LONG	
DAQmxSetup1126Cal	FNHx_set1126cal	LONG	v8.3
DAQmxAdjust1126Cal	FNHx_adj1126cal	LONG	v8.3
DAQmxSetup1141Cal	FNHx_set1141cal	LONG	v8.3
DAQmxAdjust1141Cal	FNHx_adj1141cal	LONG	v8.3
DAQmxSetup1142Cal	FNHx_set1142cal	LONG	v8.3
DAQmxAdjust1142Cal	FNHx_adj1142cal	LONG	v8.3
DAQmxSetup1143Cal	FNHx_set1143cal	LONG	v8.3
DAQmxAdjust1143Cal	FNHx_adj1143cal	LONG	v8.3
DAQmxSetup153xCal	FNHx_set153xcal	LONG	v8.3
DAQmxAdjust153xCal	FNHx_adj153xcal	LONG	v8.3
DAQmxSetup1540Cal	FNHx_set1540cal	LONG	v8.3
DAQmxAdjust1540Cal	FNHx_adj1540cal	LONG	v8.3
DAQmxSetup1102Cal	FNHx_set1102cal	LONG	v8.5
DAQmxAdjust1102Cal	FNHx_adj1102cal	LONG	v8.5
DAQmxSetup1104Cal	FNHx_set1104cal	LONG	v8.5
DAQmxAdjust1104Cal	FNHx_adj1104cal	LONG	v8.5
DAQmxSetup1112Cal	FNHx_set1112cal	LONG	v8.5
DAQmxAdjust1112Cal	FNHx_adj1112cal	LONG	v8.5
DAQmxSetup1122Cal	FNHx_set1122cal	LONG	v8.7
DAQmxAdjust1122Cal	FNHx_adj1122cal	LONG	v8.7
DAQmxSetup1124Cal	FNHx_set1124cal	LONG	v8.5
DAQmxAdjust1124Cal	FNHx_adj1124cal	LONG	v8.5
DAQmxSetup1125Cal	FNHx_set1125cal	LONG	v8.5
DAQmxAdjust1125Cal	FNHx_adj1125cal	LONG	v8.5
DAQmxSetup1502Cal	FNHx_set1502cal	LONG	v8.5
DAQmxAdjust1502Cal	FNHx_adj1502cal	LONG	v8.5
DAQmxSetup1503Cal	FNHx_set1503cal	LONG	v8.5
DAQmxAdjust1503Cal	FNHx_adj1503cal	LONG	v8.5
DAQmxAdjust1503CurrentCal	FNHx_adj1503ccl	LONG	v8.5
DAQmxSetup1520Cal	FNHx_set1520cal	LONG	v8.5
DAQmxAdjust1520Cal	FNHx_adj1520cal	LONG	v8.5
DAQmxSetup1521Cal	FNHx_set1521cal	LONG	v8.7
DAQmxAdjust1521Cal	FNHx_adj1521cal	LONG	v8.7
8. NI-DAQmx Specific Attribute Get/Set/Reset Function Declarations			
DAQmxGetPhysicalChanName	FNHx_getphyschnam	LONG	
DAQmxSetCIPrescaler	FNHx_setciprescal	LONG	
9. Device Control			
DAQmxResetDevice	FNHx_resetdevice	LONG	
10. Switch Functions			
DAQmxSwitchCreateScanList	FNHx_swcrscanlist	LONG	
DAQmxSwitchConnect	FNHx_swconnect	LONG	
DAQmxSwitchConnectMulti	FNHx_swconnectmul	LONG	
DAQmxSwitchDisconnect	FNHx_swdisconnect	LONG	
DAQmxSwitchDisconnectMulti	FNHx_swdisconnmul	LONG	
DAQmxSwitchDisconnectAll	FNHx_swdisconnall	LONG	
DAQmxSwitchSetTopologyAndReset	FNHx_swsettopares	LONG	
DAQmxSwitchFindPath	FNHx_swfindpath	LONG	
DAQmxSwitchOpenRelays	FNHx_swopenrelays	LONG	
DAQmxSwitchCloseRelays	FNHx_swclosrelays	LONG	
DAQmxSwitchWaitForSettling	FNHx_swwaitforset	LONG	
11. Signal Routing			
DAQmxConnectTerms	FNHx_connterms	LONG	
DAQmxDisconnectTerms	FNHx_disconnterms	LONG	
DAQmxTristateOutputTerm	FNHx_tristoutterm	LONG	
12. TEDS			
DAQmxConfigureTEDS	FNHx_configteds	LONG	
DAQmxClearTEDS	FNHx_clearteds	LONG	

*) As described in the "NI-DAQmx C Reference Help"

NI-DAQmx function name *)	HTBasic function name	HTBasic Ret.type	NI-DAQmx Version
13. Events & Signals			
DAQmxExportSignal	FNHx_exportsignal	LONG	
14. Buffer			
DAQmxGetBufInputBufSize	FNHx_getbufinpsiz	LONG	
DAQmxSetBufInputBufSize	FNHx_setbufinpsiz	LONG	
DAQmxGetBufOutputBufSize	FNHx_getbufoutsiz	LONG	
DAQmxSetBufOutputBufSize	FNHx_setbufoutsiz	LONG	
DAQmxGetBufOutputOnbrdBufSize	FNHx_getobufoutsz	LONG	
DAQmxSetBufOutputOnbrdBufSize	FNHx_setobufoutsz	LONG	
15. Other Get/Set specific functions			
DAQmxGetAIMax	FNHx_getaimax	LONG	
DAQmxGetAIMin	FNHx_getaimin	LONG	
DAQmxGetAILowpassCutoffFreq	FNHx_getailpcofrq	LONG	
DAQmxSetAILowpassCutoffFreq	FNHx_setailpcofrq	LONG	
DAQmxGetAILowpassEnable	FNHx_getailwpenab	LONG	
DAQmxSetAILowpassEnable	FNHx_setailwpenab	LONG	
DAQmxGetAnlgEdgeStartTrigHyst	FNHx_getanedsthys	LONG	
DAQmxSetAnlgEdgeStartTrigHyst	FNHx_setanedsthys	LONG	
DAQmxGetAIACExcitSyncEnable	FNHx_getaiacexsen	LONG	
DAQmxSetAIACExcitSyncEnable	FNHx_setaiacexsen	LONG	
DAQmxGetReadOverloadedChansExist	FNHx_getrdovchexi	LONG	
DAQmxGetReadOverloadedChans	FNHx_getrdovchans	LONG	
DAQmxGetDigLvlPauseTrigSrc	FNHx_getdiglvptsc	LONG	
DAQmxSetDigLvlPauseTrigSrc	FNHx_setdiglvptsc	LONG	
DAQmxGetPauseTrigType	FNHx_getpaustrtyp	LONG	
DAQmxSetPauseTrigType	FNHx_setpaustrtyp	LONG	
DAQmxGetDigLvlPauseTrigWhen	FNHx_getdlpatrwhn	LONG	
DAQmxSetDigLvlPauseTrigWhen	FNHx_setdlpatrwhn	LONG	
DAQmxGetCIGPSSyncSrc	FNHx_getcigpssysr	LONG	
DAQmxSetCIGPSSyncSrc	FNHx_setcigpssysr	LONG	
DAQmxGetHshkDelayAfterXfer	FNHx_gethsdelaxfr	LONG	
DAQmxSetHshkDelayAfterXfer	FNHx_sethsdelaxfr	LONG	
DAQmxGetExportedHshkEventInterlockedDeassertDelay	FNHx_getexhseiddl	LONG	
DAQmxSetExportedHshkEventInterlockedDeassertDelay	FNHx_setexhseiddl	LONG	
DAQmxGetDIDigFltrEnable	FNHx_getdidgflten	LONG	
DAQmxSetDIDigFltrEnable	FNHx_setdidgflten	LONG	
DAQmxGetDIDigFltrMinPulseWidth	FNHx_getdidgflmpw	LONG	
DAQmxSetDIDigFltrMinPulseWidth	FNHx_setdidgflmpw	LONG	
DAQmxGetAdvTrigType	FNHx_getadvtrgtyp	LONG	
DAQmxSetAdvTrigType	FNHx_setadvtrgtyp	LONG	
DAQmxGetSwitchScanRepeatMode	FNHx_getswscrpmod	LONG	
DAQmxSetSwitchScanRepeatMode	FNHx_setswscrpmod	LONG	
DAQmxGetMasterTimebaseSrc	FNHx_getmastbasrc	LONG	
DAQmxSetMasterTimebaseSrc	FNHx_setmastbasrc	LONG	
DAQmxGetMasterTimebaseRate	FNHx_getmastbrate	LONG	
DAQmxSetMasterTimebaseRate	FNHx_setmastbrate	LONG	
DAQmxGetRefClkSrc	FNHx_getrefclksrc	LONG	
DAQmxSetRefClkSrc	FNHx_setrefclksrc	LONG	
DAQmxGetRefClkRate	FNHx_getrefclkrat	LONG	
DAQmxSetRefClkRate	FNHx_setrefclkrat	LONG	
DAQmxGetSampClkTimebaseSrc	FNHx_getsclktbsrc	LONG	
DAQmxSetSampClkTimebaseSrc	FNHx_setsclktbsrc	LONG	
DAQmxGetSyncPulseSrc	FNHx_getsynpulsrc	LONG	
DAQmxSetSyncPulseSrc	FNHx_setsynpulsrc	LONG	
DAQmxGetAIThrmcplScaleType	FNHx_getaitcsctyp	LONG	v8.3
DAQmxSetAIThrmcplScaleType	FNHx_setaitcsctyp	LONG	v8.3
DAQmxResetAIThrmcplScaleType	FNHx_rstaitcsctyp	LONG	v8.3
DAQmxGetAIIsTEDS	FNHx_getaiisteds	LONG	v8.3
DAQmxGetDILogicFamily	FNHx_getdilogifam	LONG	v8.3
DAQmxSetDILogicFamily	FNHx_setdilogifam	LONG	v8.3
DAQmxResetDILogicFamily	FNHx_rstdilogifam	LONG	v8.3
DAQmxGetDIMemMapEnable	FNHx_getdimmapena	LONG	v8.3
DAQmxSetDIMemMapEnable	FNHx_setdimmapena	LONG	v8.3
DAQmxResetDIMemMapEnable	FNHx_rstdimmapena	LONG	v8.3
DAQmxGetDIAcquireOn	FNHx_getdiacquion	LONG	v8.3
DAQmxSetDIAcquireOn	FNHx_setdiacquion	LONG	v8.3
DAQmxResetDIAcquireOn	FNHx_rstdiacquion	LONG	v8.3
DAQmxGetDOLineStatesStartState	FNHx_getdolsstrts	LONG	v8.3
DAQmxSetDOLineStatesStartState	FNHx_setdolsstrts	LONG	v8.3
DAQmxResetDOLineStatesStartState	FNHx_rstdolsstrts	LONG	v8.3

NI-DAQmx function name *)	HTBasic function name	HTBasic Ret.type	NI-DAQmx Version
15. Other Get/Set specific functions (Continued 2)			
DAQmxGetDOLineStatesPausedState	FNHx_getdolspauss	LONG	v8.3
DAQmxSetDOLineStatesPausedState	FNHx_setdolspauss	LONG	v8.3
DAQmxResetDOLineStatesPausedState	FNHx_rstdolspauss	LONG	v8.3
DAQmxGetDOLineStatesDoneState	FNHx_getdolsdones	LONG	v8.3
DAQmxSetDOLineStatesDoneState	FNHx_setdolsdones	LONG	v8.3
DAQmxResetDOLineStatesDoneState	FNHx_rstdolsdones	LONG	v8.3
DAQmxSetDigitalLogicFamilyPowerUpState	FNHx_setdlfampust	LONG	v8.3
DAQmxGetDOLogicFamily	FNHx_getdologifam	LONG	v8.3
DAQmxSetDOLogicFamily	FNHx_setdologifam	LONG	v8.3
DAQmxResetDOLogicFamily	FNHx_rstdologifam	LONG	v8.3
DAQmxGetDOMemMapEnable	FNHx_getdommapena	LONG	v8.3
DAQmxSetDOMemMapEnable	FNHx_setdommapena	LONG	v8.3
DAQmxResetDOMemMapEnable	FNHx_rstdommapena	LONG	v8.3
DAQmxGetDOGenerateOn	FNHx_getdogeneron	LONG	v8.3
DAQmxSetDOGenerateOn	FNHx_setdogeneron	LONG	v8.3
DAQmxResetDOGenerateOn	FNHx_rstdogeneron	LONG	v8.3
DAQmxSetPhysicalChanName	FNHx_setphyschnam	LONG	v8.3
DAQmxGetExportedPauseTrigOutputTerm	FNHx_getexptouttm	LONG	v8.3
DAQmxSetExportedPauseTrigOutputTerm	FNHx_setexptouttm	LONG	v8.3
DAQmxResetExportedPauseTrigOutputTerm	FNHx_rstexptouttm	LONG	v8.3
DAQmxGetExportedPauseTrigLvlActiveLvl	FNHx_getexptlaclv	LONG	v8.3
DAQmxSetExportedPauseTrigLvlActiveLvl	FNHx_setexptlaclv	LONG	v8.3
DAQmxResetExportedPauseTrigLvlActiveLvl	FNHx_rstexptlaclv	LONG	v8.3
DAQmxGetExportedRefTrigPulsePolarity	FNHx_getexrftrppo	LONG	v8.3
DAQmxSetExportedRefTrigPulsePolarity	FNHx_setexrftrppo	LONG	v8.3
DAQmxResetExportedRefTrigPulsePolarity	FNHx_rstexrftrppo	LONG	v8.3
DAQmxGetExportedStartTrigPulsePolarity	FNHx_getexsttrppo	LONG	v8.3
DAQmxSetExportedStartTrigPulsePolarity	FNHx_setexsttrppo	LONG	v8.3
DAQmxResetExportedStartTrigPulsePolarity	FNHx_rstexsttrppo	LONG	v8.3
DAQmxGetExportedRdyForXferEventDeassertCond	FNHx_getexrdfxedc	LONG	v8.3
DAQmxSetExportedRdyForXferEventDeassertCond	FNHx_setexrdfxedc	LONG	v8.3
DAQmxResetExportedRdyForXferEventDeassertCond	FNHx_rstexrdfxedc	LONG	v8.3
DAQmxGetExportedRdyForXferEventDeassertCondCustomThreshold	FNHx_getexrdyfxed	LONG	v8.3
DAQmxSetExportedRdyForXferEventDeassertCondCustomThreshold	FNHx_setexrdyfxed	LONG	v8.3
DAQmxResetExportedRdyForXferEventDeassertCondCustomThreshold	FNHx_rstexrdyfxed	LONG	v8.3
DAQmxGetExportedDataActiveEventOutputTerm	FNHx_getexdaceout	LONG	v8.3
DAQmxSetExportedDataActiveEventOutputTerm	FNHx_setexdaceout	LONG	v8.3
DAQmxResetExportedDataActiveEventOutputTerm	FNHx_rstexdaceout	LONG	v8.3
DAQmxGetExportedDataActiveEventLvlActiveLvl	FNHx_getexdaevlal	LONG	v8.3
DAQmxSetExportedDataActiveEventLvlActiveLvl	FNHx_setexdaevlal	LONG	v8.3
DAQmxResetExportedDataActiveEventLvlActiveLvl	FNHx_rstexdaevlal	LONG	v8.3
DAQmxGetExportedRdyForStartEventOutputTerm	FNHx_getexrfseotm	LONG	v8.3
DAQmxSetExportedRdyForStartEventOutputTerm	FNHx_setexrfseotm	LONG	v8.3
DAQmxResetExportedRdyForStartEventOutputTerm	FNHx_rstexrfseotm	LONG	v8.3
DAQmxGetExportedRdyForStartEventLvlActiveLvl	FNHx_getexrfselal	LONG	v8.3
DAQmxSetExportedRdyForStartEventLvlActiveLvl	FNHx_setexrfselal	LONG	v8.3
DAQmxResetExportedRdyForStartEventLvlActiveLvl	FNHx_rstexrfselal	LONG	v8.3
DAQmxGetDevProductCategory	FNHx_getdevprdcat	LONG	v8.3
DAQmxGetDevChassisModuleDevNames	FNHx_getdevchmdnm	LONG	v8.3
DAQmxGetDevAnlgTrigSupported	FNHx_getdevatrsup	LONG	v8.3
DAQmxGetDevDigTrigSupported	FNHx_getdevdtrsup	LONG	v8.3
DAQmxGetDevAIMaxSingleChanRate	FNHx_getdevaimscr	LONG	v8.3
DAQmxGetDevAIMaxMultiChanRate	FNHx_getdevaimmcr	LONG	v8.3
DAQmxGetDevAIMinRate	FNHx_getdevaimrat	LONG	v8.3
DAQmxGetDevAISimultaneousSamplingSupported	FNHx_getdevasssup	LONG	v8.3
DAQmxGetDevAITrigUsage	FNHx_getdevaitrus	LONG	v8.3
DAQmxGetDevAIVoltageRngs	FNHx_getdevavrang	LONG	v8.3
DAQmxGetDevAIVoltageIntExcitDiscreteVals	FNHx_getdevaviedv	LONG	v8.3
DAQmxGetDevAIVoltageIntExcitRangeVals	FNHx_getdevavierv	LONG	v8.3
DAQmxGetDevAICurrentRngs	FNHx_getdevacurra	LONG	v8.3
DAQmxGetDevAICurrentIntExcitDiscreteVals	FNHx_getdevaciedv	LONG	v8.3
DAQmxGetDevAIFreqRngs	FNHx_getdevafrqrn	LONG	v8.3
DAQmxGetDevAIGains	FNHx_getdevaigain	LONG	v8.3
DAQmxGetDevAICouplings	FNHx_getdevaicoup	LONG	v8.3
DAQmxGetDevAILowpassCutoffFreqDiscreteVals	FNHx_getdevalcfdv	LONG	v8.3
DAQmxGetDevAILowpassCutoffFreqRangeVals	FNHx_getdevalcfrv	LONG	v8.3
DAQmxGetDevAOPhysicalChans	FNHx_getdevaophch	LONG	v8.3
DAQmxGetDevAOSampClkSupported	FNHx_getdevaoscsp	LONG	v8.3
DAQmxGetDevAOMaxRate	FNHx_getdevaomart	LONG	v8.3

NI-DAQmx function name *)	HTBasic function name	HTBasic Ret.type	NI-DAQmx Version
15. Other Get/Set specific functions (Continued 3)			
DAQmxGetDevAOMinRate	FNHx_getdevaomirt	LONG	v8.3
DAQmxGetDevAOTrigUsage	FNHx_getdevaotrus	LONG	v8.3
DAQmxGetDevAOVoltageRngs	FNHx_getdevaovorg	LONG	v8.3
DAQmxGetDevAOCurrentRngs	FNHx_getdevaocurg	LONG	v8.3
DAQmxGetDevAOGains	FNHx_getdevaogain	LONG	v8.3
DAQmxGetDevDIMaxRate	FNHx_getdevdimart	LONG	v8.3
DAQmxGetDevDITrigUsage	FNHx_getdevditrus	LONG	v8.3
DAQmxGetDevDOMaxRate	FNHx_getdevdomart	LONG	v8.3
DAQmxGetDevDOTrigUsage	FNHx_getdevdotrus	LONG	v8.3
DAQmxGetDevCITrigUsage	FNHx_getdevcitrus	LONG	v8.3
DAQmxGetDevCISampClkSupported	FNHx_getdevciscsp	LONG	v8.3
DAQmxGetDevCIMaxTimebase	FNHx_getdevcimatb	LONG	v8.3
DAQmxGetDevCOTrigUsage	FNHx_getdevcotrus	LONG	v8.3
DAQmxGetDevCOMaxTimebase	FNHx_getdevcomatb	LONG	v8.3
DAQmxGetDevNumDMAChans	FNHx_getdevnmdchn	LONG	v8.3
DAQmxGetDevCompactDAQChassisDevName	FNHx_getdevcdcdnm	LONG	v8.3
DAQmxGetDevCompactDAQSlotNum	FNHx_getdevcdslnm	LONG	v8.3
DAQmxGetTaskNumDevices	FNHx_gettsknumdev	LONG	v8.3
DAQmxGetSampClkUnderflowBehavior	FNHx_getsmclufbeh	LONG	v8.3
DAQmxSetSampClkUnderflowBehavior	FNHx_setsmclufbeh	LONG	v8.3
DAQmxResetSampClkUnderflowBehavior	FNHx_rstsmclufbeh	LONG	v8.3
DAQmxGetAIConvRateEx	FNHx_getaicvratex	LONG	v8.3
DAQmxSetAIConvRateEx	FNHx_setaicvratex	LONG	v8.3
DAQmxResetAIConvRateEx	FNHx_rstaicvratex	LONG	v8.3
DAQmxGetAIConvMaxRateEx	FNHx_getaicvmrtex	LONG	v8.3
DAQmxGetAIConvSrcEx	FNHx_getaicvsrcex	LONG	v8.3
DAQmxSetAIConvSrcEx	FNHx_setaicvsrcex	LONG	v8.3
DAQmxResetAIConvSrcEx	FNHx_rstaicvsrcex	LONG	v8.3
DAQmxGetAIConvActiveEdgeEx	FNHx_getaicvaceex	LONG	v8.3
DAQmxSetAIConvActiveEdgeEx	FNHx_setaicvaceex	LONG	v8.3
DAQmxResetAIConvActiveEdgeEx	FNHx_rstaicvaceex	LONG	v8.3
DAQmxGetAIConvTimebaseDivEx	FNHx_getaicvtbdex	LONG	v8.3
DAQmxSetAIConvTimebaseDivEx	FNHx_setaicvtbdex	LONG	v8.3
DAQmxResetAIConvTimebaseDivEx	FNHx_rstaicvtbdex	LONG	v8.3
DAQmxGetAIConvTimebaseSrcEx	FNHx_getaicvtbsex	LONG	v8.3
DAQmxSetAIConvTimebaseSrcEx	FNHx_setaicvtbsex	LONG	v8.3
DAQmxResetAIConvTimebaseSrcEx	FNHx_rstaicvtbsex	LONG	v8.3
DAQmxGetDelayFromSampClkDelayUnits	FNHx_getdlfscdunt	LONG	v8.3
DAQmxSetDelayFromSampClkDelayUnits	FNHx_setdlfscdunt	LONG	v8.3
DAQmxResetDelayFromSampClkDelayUnits	FNHx_rstdlfscdunt	LONG	v8.3
DAQmxGetDelayFromSampClkDelayUnitsEx	FNHx_getdlfscduex	LONG	v8.3
DAQmxSetDelayFromSampClkDelayUnitsEx	FNHx_setdlfscduex	LONG	v8.3
DAQmxResetDelayFromSampClkDelayUnitsEx	FNHx_rstdlfscduex	LONG	v8.3
DAQmxGetDelayFromSampClkDelay	FNHx_getdlfscdlay	LONG	v8.3
DAQmxSetDelayFromSampClkDelay	FNHx_setdlfscdlay	LONG	v8.3
DAQmxResetDelayFromSampClkDelay	FNHx_rstdlfscdlay	LONG	v8.3
DAQmxGetDelayFromSampClkDelayEx	FNHx_getdlfscdlex	LONG	v8.3
DAQmxSetDelayFromSampClkDelayEx	FNHx_setdlfscdlex	LONG	v8.3
DAQmxResetDelayFromSampClkDelayEx	FNHx_rstdlfscdlex	LONG	v8.3
DAQmxGetDigPatternPauseTrigSrc	FNHx_getdigpptsrc	LONG	v8.3
DAQmxSetDigPatternPauseTrigSrc	FNHx_setdigpptsrc	LONG	v8.3
DAQmxResetDigPatternPauseTrigSrc	FNHx_rstdigpptsrc	LONG	v8.3
DAQmxGetDigPatternPauseTrigPattern	FNHx_getdigpptpat	LONG	v8.3
DAQmxSetDigPatternPauseTrigPattern	FNHx_setdigpptpat	LONG	v8.3
DAQmxResetDigPatternPauseTrigPattern	FNHx_rstdigpptpat	LONG	v8.3
DAQmxGetDigPatternPauseTrigWhen	FNHx_getdigpptwhn	LONG	v8.3
DAQmxSetDigPatternPauseTrigWhen	FNHx_setdigpptwhn	LONG	v8.3
DAQmxResetDigPatternPauseTrigWhen	FNHx_rstdigpptwhn	LONG	v8.3
DAQmxGetWriteNextWriteIsLast	FNHx_getwrnxwrila	LONG	v8.3
DAQmxSetWriteNextWriteIsLast	FNHx_setwrnxwrila	LONG	v8.3
DAQmxResetWriteNextWriteIsLast	FNHx_rstwrnxwrila	LONG	v8.3
DAQmxGetPhysicalChanAITermCfgs	FNHx_getphchaitcf	LONG	v8.3
DAQmxGetPhysicalChanAOTermCfgs	FNHx_getphchaotcf	LONG	v8.3
DAQmxGetPhysicalChanDIPortWidth	FNHx_getphchdipwd	LONG	v8.3
DAQmxGetPhysicalChanDISampClkSupported	FNHx_getphchdiscs	LONG	v8.3
DAQmxGetPhysicalChanDIChangeDetectSupported	FNHx_getphchdidsp	LONG	v8.3
DAQmxGetPhysicalChanDOPortWidth	FNHx_getphchdopwd	LONG	v8.3
DAQmxGetDevTerminals	FNHx_getdevtermin	LONG	v8.7

NI-DAQmx function name *)	HTBasic function name	HTBasic Ret.type	NI-DAQmx Version
DAQmxGetPhysicalChanDOSampClkSupported	FNHx_getphchdoscs	LONG	v8.3
DAQmxGetSampClkTimingResponseMode	FNHx_getsctresmod	LONG	v8.3
DAQmxSetSampClkTimingResponseMode	FNHx_setsctresmod	LONG	v8.3
DAQmxResetSampClkTimingResponseMode	FNHx_rstsctresmod	LONG	v8.3
DAQmxGetAIVoltagedBRef	FNHx_getaivolbref	LONG	v8.5
DAQmxSetAIVoltagedBRef	FNHx_setaivolbref	LONG	v8.5
DAQmxResetAIVoltagedBRef	FNHx_rstaivolbref	LONG	v8.5
DAQmxGetAISoundPressuredBRef	FNHx_getaispbyref	LONG	v8.5
DAQmxSetAISoundPressuredBRef	FNHx_setaispbyref	LONG	v8.5
DAQmxResettAISoundPressuredBRef	FNHx_rstaispbyref	LONG	v8.5
DAQmxGetAIAcceldBRef	FNHx_getaiacbyref	LONG	v8.5
DAQmxSetAIAcceldBRef	FNHx_setaiacbyref	LONG	v8.5
DAQmxResetAIAcceldBRef	FNHx_rstaiacbyref	LONG	v8.5
DAQmxGetAIADCTimingMode	FNHx_getaiadctimd	LONG	v8.5
DAQmxSetAIADCTimingMode	FNHx_setaiadctimd	LONG	v8.5
DAQmxResetAIADCTimingMode	FNHx_rstaiadctimd	LONG	v8.5
DAQmxGetCOConstrainedGenMode	FNHx_getcocongmod	LONG	v8.5
DAQmxSetCOConstrainedGenMode	FNHx_setcocongmod	LONG	v8.5
DAQmxResetCOConstrainedGenMode	FNHx_rstcocongmod	LONG	v8.5
DAQmxGetExportedSampClkDelayOffset	FNHx_getexscdloff	LONG	v8.5
DAQmxSetExportedSampClkDelayOffset	FNHx_setexscdloff	LONG	v8.5
DAQmxResetExportedSampClkDelayOffset	FNHx_rstexscdloff	LONG	v8.5
DAQmxGetDevCIMaxSize	FNHx_getdevcimasz	LONG	v8.5
DAQmxGetDevCOMaxSize	FNHx_getdevcomasz	LONG	v8.5
DAQmxGetReadOvercurrentChansExist	FNHx_getrdocuchex	LONG	v8.5
DAQmxGetReadOvercurrentChans	FNHx_getrdovcuchn	LONG	v8.5
DAQmxGetReadOpenCurrentLoopChansExist	FNHx_getrdoculcex	LONG	v8.5
DAQmxGetReadOpenCurrentLoopChans	FNHx_getrdopculch	LONG	v8.5
DAQmxGetWriteOvercurrentChansExist	FNHx_getwrovcchex	LONG	v8.5
DAQmxGetWriteOvercurrentChans	FNHx_getwrovcchns	LONG	v8.5
DAQmxGetWriteOpenCurrentLoopChansExist	FNHx_getwropcchex	LONG	v8.5
DAQmxGetWriteOpenCurrentLoopChans	FNHx_getwropcchns	LONG	v8.5
DAQmxGetWritePowerSupplyFaultChansExist	FNHx_getwrpsfchex	LONG	v8.5
DAQmxGetWritePowerSupplyFaultChans	FNHx_getwrpsfchns	LONG	v8.5
DAQmxGetAIVoltageACRMSUnits	FNHx_getaivoacrmu	LONG	v8.6
DAQmxSetAIVoltageACRMSUnits	FNHx_setaivoacrmu	LONG	v8.6
DAQmxResetAIVoltageACRMSUnits	FNHx_rstaivoacrmu	LONG	v8.6
DAQmxGetAICurrentACRMSUnits	FNHx_getaicuacrmu	LONG	v8.6
DAQmxSetAICurrentACRMSUnits	FNHx_setaicuacrmu	LONG	v8.6
DAQmxResetAICurrentACRMSUnits	FNHx_rstaicuacrmu	LONG	v8.6
DAQmxGetAOVoltageCurrentLimit	FNHx_getaoculimit	LONG	v8.6
DAQmxSetAOVoltageCurrentLimit	FNHx_setaoculimit	LONG	v8.6
DAQmxResetAOVoltageCurrentLimit	FNHx_rstaoculimit	LONG	v8.6
DAQmxGetAOFuncGenType	FNHx_getaofungtyp	LONG	v8.6
DAQmxSetAOFuncGenType	FNHx_setaofungtyp	LONG	v8.6
DAQmxResetAOFuncGenType	FNHx_rstaofungtyp	LONG	v8.6
DAQmxGetAOFuncGenFreq	FNHx_getaofungfrq	LONG	v8.6
DAQmxSetAOFuncGenFreq	FNHx_setaofungfrq	LONG	v8.6
DAQmxResetAOFuncGenFreq	FNHx_rstaofungfrq	LONG	v8.6
DAQmxGetAOFuncGenAmplitude	FNHx_getaofungamp	LONG	v8.6
DAQmxSetAOFuncGenAmplitude	FNHx_setaofungamp	LONG	v8.6
DAQmxResetAOFuncGenAmplitude	FNHx_rstaofungamp	LONG	v8.6
DAQmxGetAOFuncGenOffset	FNHx_getaofungoff	LONG	v8.6
DAQmxSetAOFuncGenOffset	FNHx_setaofungoff	LONG	v8.6
DAQmxResetAOFuncGenOffset	FNHx_rstaofungoff	LONG	v8.6
DAQmxGetAOFuncGenSquareDutyCycle	FNHx_getaofungsdc	LONG	v8.6
DAQmxSetAOFuncGenSquareDutyCycle	FNHx_setaofungsdc	LONG	v8.6
DAQmxResetAOFuncGenSquareDutyCycle	FNHx_rstaofungsdc	LONG	v8.6
DAQmxGetAOFuncGenModulationType	FNHx_getaofungmtp	LONG	v8.6
DAQmxSetAOFuncGenModulationType	FNHx_setaofungmtp	LONG	v8.6
DAQmxResetAOFuncGenModulationType	FNHx_rstaofungmtp	LONG	v8.6
DAQmxGetAOFuncGenFMDeviation	FNHx_getaofungfmd	LONG	v8.6
DAQmxSetAOFuncGenFMDeviation	FNHx_setaofungfmd	LONG	v8.6
DAQmxResetAOFuncGenFMDeviation	FNHx_rstaofungfmd	LONG	v8.6
DAQmxGetSampTimingEngine	FNHx_getsamtimeng	LONG	v8.6
DAQmxSetSampTimingEngine	FNHx_setsamtimeng	LONG	v8.6
DAQmxResetSampTimingEngine	FNHx_rstsamtimeng	LONG	v8.6
DAQmxGetPhysicalChanAOManualControlEnable	FNHx_getphchaomce	LONG	v8.6
DAQmxSetPhysicalChanAOManualControlEnable	FNHx_setphchaomce	LONG	v8.6

NI-DAQmx function name *)	HTBasic function name	HTBasic Ret.type	NI-DAQmx Version
DAQmxResetPhysicalChanAOManualControlEnable	FNHx_rstphchaomce	LONG	v8.6
DAQmxGetPhysicalChanAOManualControlAmplitude	FNHx_getpcaomcamp	LONG	v8.6
DAQmxGetPhysicalChanAOManualControlFreq	FNHx_getpcaomcfrq	LONG	v8.6
DAQmxPerformBridgeOffsetNullingCalEx	FNHx_perfbrofncex	LONG	v8.7
DAQmxPerformStrainShuntCal	FNHx_perfstrshcal	LONG	v8.7
DAQmxPerformBridgeShuntCal	FNHx_perfbrishcal	LONG	v8.7
DAQmxSetCOPulseFreq	FNHx_setcopulfreq	LONG	
DAQmxSetCOPulseDutyCyc	FNHx_setcopdutcyc	LONG	
DAQmxSetBufferAttribute	FNHx_setbufattrib	LONG	
DAQmxSetTrigAttribute	FNHx_settriattrib	LONG	
16. Callback services			
DAQmxRegisterEveryNSamplesEvent	FNHx_regevnsmevnt	LONG	
DAQmxRegisterSignalEvent	FNHx_regsignevent	LONG	
DAQmxRegisterDoneEvent	FNHx_regdoneevent	LONG	
17. Watchdog			
DAQmxCreateWatchdogTimerTask	FNHx_crwdogtimtsk	LONG	
DAQmxControlWatchdogTask	FNHx_ctrlwdogtsk	LONG	
18. Storage			
DAQmxSaveTask	FNHx_savetask	LONG	
DAQmxDeleteSavedTask	FNHx_delsavedtask	LONG	
19. Special calibration functions			
DAQmxGetAIChanCalPolyForwardCoeff	FNHx_getaichcpfco	LONG	
DAQmxSetAIChanCalPolyForwardCoeff	FNHx_setaichcpfco	LONG	
DAQmxResetAIChanCalPolyForwardCoeff	FNHx_rstaichcpfco	LONG	
DAQmxGetAIDevScalingCoeff	FNHx_getaidevscco	LONG	

Determining HTBasic function names for NI-DAQmx functions

In case you need to know the corresponding HTBasic function name for a given NI-DAQmx function you might want to use the HTBasic support function `FNHx_load_libs`:

```
INTEGER Debug, Dummy
Dummy=FNHx_load_libs(Debug,"DAQmxCreateTEDSAIPosRVDTChan")
```

HTBasic Output:

```
NI-DAQmx function name: DAQmxCreateTEDSAIPosRVDTChan
HTBasic function name:  FNHx_crtaiprvdtch
```

```
! this lists all availabe NI-DAQmx functions and their HTBasic function names
Dummy=FNHx_load_libs(Debug,"*")
```

Info: `FNHx_load_libs` is part of the HTBasic library file "`daqmx_htb.lib`".

Special HTBasic Support Functions

HTBasic function name	Description	HTBasic Return type
SUB Hx_manage_subs	Load / Init or Unload/De-init HTBasic lib	VOID
FNHx_getdaqmxvers$	Retrieve DAQmx version numbers	STRING
FNHx_getdllvers$	Retrieve DAQmx wrapper DLL version number	STRING
FNHx_getdevname$	Extract device name (strips leading "/")	STRING
FNHx_gettnamwpref	Special function	LONG
SUB Hx_show_errstat	Show error status information	VOID
FNHx_get_constant	Return constant from nidaqmx.h	INT / LONG / REAL
FNHx_get_constant$	Return constant from nidaqmx.h	STRING
SUB Hx_enter_kbd	Wait for key stroke	VOID
FNHx_get_dev_info	Get device and system information	LONG
FNHx_lib_version$	Get HTBasic library version information	STRING
FNHx_numofdevices	Get numbers of NI devices defined in MAX	LONG
FNHx_getsfpath$	Get path to special windows directories	STRING

Global variables for HTBasic

The following global variables are used for callback services.

HTBasic variable name	Description	HTBasic Return type
g_Evnt_taskhandle	Taskhandle for which the callback occured	LONG
g_Evnt_type	Event type for callback	LONG

```
g_Evnt_status          Special status information for event         LONG
g_Callback_locked      Callback Locking indicator                   LONG
g_Callback_pending     Indicates that another callback is pending   LONG
g_nSamples             Number of read samples for callback          LONG
```

Please note: You must declare and initialize all variables in HTBasic before you call DLL functions, otherwise the system may become instable.

Variable mapping

NI-DAQmx variable	HTBasic variable
short	INTEGER
long	LONG
uInt8	INTEGER
uInt16	INTEGER
uInt32	LONG
uInt64	N.A.
bool32	LONG
char	STRING
float64	REAL

Use the National Instruments NI-DAQmx Documentation

It is important that you understand how the original functions of the National Instruments DAQmx library work, so always look into the documentation "NI-DAQmx C Reference Help" for a more detailed description of the parameters used and their meaning. You can find many example programs which can be easily ported to HTBasic in the following directory:

```
%Program Files%\National Instruments\NI-DAQ\Examples\DAQmx ANSI C
```

The function reference always shows the HTBasic funtion name and the original NI-DAQmx function name, see the example below:

FNHx_loadtask NI-DAQmx: **DAQmxLoadTask**

Purpose: Loads an existing task

Return variables of DLL functions

Due to the nature of DLL handling it is very important that you always use a return variable for the function calls, otherwise HTBasic may crash. Example:

Correct:

```
LONG Result
LONG Taskid2
Result=FNHx_stoptask((Taskid2))
!
IF Result=0 THEN
   !... subsequent commands...
END IF
```

Incorrect:

```
LONG Taskid2
!
IF FNHx_stoptask((Taskid2))=0 THEN  ! this may crash HTBasic !!!
   !... subsequent commands...
END IF
```

The (incorrect) construction shown above should never be used, you always must use a return value for DLL functions, otherwise HTBasic may crash or become unstable.

Function declarations for all implemented functions

Task Configuration and Control

FNHx_loadtask NI-DAQmx: **DAQmxLoadTask**

Purpose: Loads an existing task

HTBasic Definition:
```
LONG Result=FNHx_loadtask((Taskname$),LONG Taskid)
```

```
Input:
 Taskname$          Name of task to load
Return value:    0 = Function was successful
                     Taskid now contains Task ID of loaded task
                <>0 = Error occured
```

Example:
```
 LONG Result
 LONG Taskid
 Result=FNHx_loadtask("VoltageTask",Taskid)
 !
 IF Result=0 THEN
    !... subsequent commands...
 END IF
```

FNHx_createtask NI-DAQmx: **DAQmxLoadTask**

Purpose: Creates a new task

HTBasic Definition:
```
LONG Result=FNHx_createtask(LONG Taskid,(Taskname$))
```

```
Input:
 Taskname$          Name of task to be created (or NULL string "")
Return value:    0 = Function was successful
                     Taskid now contains Task ID of newly created task
                <>0 = Error occured
```

Example:
```
 INTEGER Verbose
 LONG Result
 LONG Analogin_task
 Result=FNHx_createtask(Analogin_task,"AnalogInTask")
 !
 IF Result=0 THEN
    !... subsequent commands...
 END IF
```

FNHx_starttask NI-DAQmx: **DAQmxStartTask**

Purpose: Starts a task

HTBasic Definition:
```
LONG Result=FNHx_starttask((LONG Taskid))
```

```
Input:
 Taskid             Task ID of task to start
Return value:    0 = Function was successful
                <>0 = Error occured
```

Example:
```
 LONG Result
 LONG Taskid
 Result=FNHx_starttask((Taskid))
 !
 IF Result=0 THEN
    !... subsequent commands...
 END IF
```

FNHx_stoptask — NI-DAQmx: DAQmxStopTask

Purpose: Stops a task

HTBasic Definition:
```
LONG Result=FNHx_stoptask((LONG Taskid))
```

```
Input:
 Taskid             Task ID of task to stop
Return value:    0 = Function was successful
                <>0 = Error occured
```

Example:
```
 LONG Result
 LONG Taskhandle
 Result=FNHx_stoptask((Taskhandle))
 !
 IF Result=0 THEN
    !... subsequent commands...
 END IF
```

FNHx_cleartask — NI-DAQmx: DAQmxClearTask

Purpose: Clears a task

HTBasic Definition:
```
LONG Result=FNHx_cleartask((LONG Taskid))
```

```
Input:
 Taskid             Task ID of task to be deleted
Return value:    0 = Function was successful
                <>0 = Error occured
```

Example:
```
 LONG Result
 LONG Taskid
 Result=FNHx_cleartask((Taskid))
 !
 IF Result=0 THEN
    !... subsequent commands...
 END IF
```

FNHx_waituntaskdn — NI-DAQmx: DAQmxWaitUntilTaskIsDone

Purpose: Waits until a task is done

HTBasic Definition:
```
LONG Result=FNHx_waituntaskdn((LONG Taskid), REAL Timetowait)
```

```
Input:
 Taskid             Task ID of task to be checked
 Timetowait         Wait time, until the task should be finished until a timeout occurs
Return value:    0 = Function was successful
                <>0 = Error occured
```

Example:
```
 LONG Result
 LONG Taskid
 REAL Timetowait
 Timetowait=5                ! 5 wait time
 Result=FNHx_waituntaskdn((Taskid),Timetowait)
 !
 IF Result=0 THEN
    !... subsequent commands...
 END IF
```

FNHx_istaskdone NI-DAQmx: **DAQmxIsTaskDone**

Purpose: Checks if a task is done

HTBasic Definition:
```
LONG Result=FNHx_istaskdone((LONG Taskid), LONG Istaskdone)
```

```
Input:
 Taskid            Task ID of task to be checked
Return value:   0 = Function was successful
                    Istaskdone now contains 0 (Task done) or 1 (Task still running)
              <>0 = Error occured
```

Example:
```
 LONG Istaskdone
 LONG Taskid
 Result=FNHx_istaskdone((Taskid),Istaskdone)
 !
 IF Result=0 THEN
    PRINT "Task Status of";Taskid;":";Istaskdone
    !... subsequent commands...
 END IF
```

FNHx_taskcontrol NI-DAQmx: **DAQmxTaskControl**

Purpose: Control a task

HTBasic Definition:
```
LONG Result=FNHx_taskcontrol((LONG Taskid), (LONG Action))
```

```
Input:
 Taskid            Task ID of task to be controlled
 Action            Control code (e.g. "DAQmx_Val_Task_Start" or "DAQmx_Val_Task_Abort")
Return value:   0 = Function was successful
              <>0 = Error occured
```

Example:
```
 LONG Taskid, Action
 INTEGER Iresult, Debug
 Action=FNHx_get_constant("DAQmx_Val_Task_Abort",Iresult,Debug)    ! get LONG constant
 Result=FNHx_taskcontrol((Taskid),(Action))
 !
 IF Result=0 THEN
    !... subsequent commands...
 END IF
```

FNHx_getnthtaskch NI-DAQmx: **DAQmxGetNthTaskChannel**

Purpose: Returns the *nth* channel of a task

HTBasic Definition:
```
LONG Result=FNHx_getnthtaskch((LONG Taskid), (LONG Index), Buf$, (LONG Bufsize))
```

```
Input:
 Taskid            The task used in this function
 Index             The Nth channel you want to return. The index starts at 1
 Bufsize           The size, in bytes, of buffer. If you pass 0, this function returns
                   the number of bytes needed to allocate.
Output:
 Buf$         The Nth channel in the index. If you pass NULL, this function returns
              the number of bytes needed to allocate.
Return value:   0 = Function was successful
              <>0 = Error occured
```

Example:
```
 LONG Taskid, Index, Bufsize
 DIM Buf$[1024]
 Asize=SIZE(Buf$)          ! Size of array
 Result=FNHx_getnthtaskch((Taskid),(Index),Buf$,(Bufsize))
```

Channel Configuration / Creation

FNHx_craivoltchan NI-DAQmx: **DAQmxCreateAIVoltageChan**

Purpose: Defines an analog Input Voltage channel

HTBasic Definition:

```
LONG Result=FNHx_craivoltchan((LONG Taskid), Phys_chan$, My_chn$, (LONG Termcfg),
                              (REAL Minval), (REAL Maxval), (LONG Units), Custscalname$)

Input:
 Taskid            The task to which to add the channels that this function creates
 Phys_chan$        The names of the physical channels to use to create virtual channels
 My_chn$           The name(s) to assign to the created virtual channel(s)
 Termcfg           The input terminal configuration for the channel
 Minval            The minimum value, in Units, that you expect to measure
 Maxval            The maximum value, in Units, that you expect to measure
 Units             The units to use to return the voltage measurements
 Custscalname$     The name of a custom scale to apply to the channel

Return value:   0 = Function was successful
              <>0 = Error occured
```

Example:

```
 INTEGER Iresult1, Iresult2
 LONG Result, Taskid, Termcfg, Units
 REAL V_min, V_max
 DIM Tmp$[255], Devchn$[255]
 !
 Termcfg=FNHx_get_constant("DAQmx_Val_Cfg_Default",Iresult1) ! constant 1
 Units=FNHx_get_constant("DAQmx_Val_Volts",Iresult2)         ! constant 2
 V_min=-10.0
 V_max=10.0
 Tmp$=""
 Devchn$="Dev1/ai0"
 Result=FNHx_craivoltchan((Taskid),Devchn$,Tmp$,(Termcfg),(V_min),(V_max),(Units),Tmp$)
```

FNHx_craicurrchan NI-DAQmx: **DAQmxCreateAICurrentChan**

Purpose: Defines an analog Current Input channel

HTBasic Definition:

```
LONG Result=FNHx_craicurrchan((LONG Taskid), Phys_chan$, My_chn$, (LONG Termcfg),
                              (REAL Minval), (REAL Maxval), (LONG Units),
                              (REAL Shuntresloc), (REAL Extshuntresval), Custscalename$)

Input:
 Taskid            The task to which to add the channels that this function creates
 Phys_chan$        The names of the physical channels to use to create virtual channels
 My_chn$           The name(s) to assign to the created virtual channel(s)
 Termcfg           The input terminal configuration for the channel
 Minval            The minimum value, in Units, that you expect to measure
 Maxval            The maximum value, in Units, that you expect to measure
 Units             The units to use to return the voltage measurements
 Shuntresloc       The location of the shunt resistor
 Extshuntresval    The value, in ohms, of an external shunt resistor
 Custscalename$    The name of a custom scale to apply to the channel

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_craithcpchan NI-DAQmx: **DAQmxCreateAIThrmcplChan**

Purpose: Defines analog temperature input channel

HTBasic Definition:

```
LONG Result=FNHx_craithcpchan((LONG Taskid), Phys_chan$, My_chn$, (LONG Termcfg),
                              (REAL Minval), (REAL Maxval), (LONG Units),
                              (REAL Tcouptype), (REAL Cjcsrc), (REAL Cjcval), Cjcchan$)
```

Input:

Taskid	The task to which to add the channels that this function creates
Phys_chan$	The names of the physical channels to use to create virtual channels
My_chn$	The name(s) to assign to the created virtual channel(s)
Termcfg	The input terminal configuration for the channel
Minval	The minimum value, in **Units**, that you expect to measure
Maxval	The maximum value, in **Units**, that you expect to measure
Units	The units to use to return the voltage measurements
Tcouptype	The type of thermocouple connected to the channel
Cjcsrc	The source of cold junction compensation
Cjcval	The temperature of the cold junction of the thermocouple if you set **Cjcsrc** to DAQmx_Val_ConstVal
Cjcchan$	The channel that acquires the temperature of the thermocouple cold-junction if you set **Cjcsrc** to DAQmx_Val_Chan

Return value: 0 = Function was successful
<>0 = Error occured

FNHx_crairtdchan NI-DAQmx: **DAQmxCreateAIRTDChan**

Purpose: Creates a channel that uses a RTD to measure temperature

HTBasic Definition:

```
LONG Result=FNHx_crairtdchan((LONG Taskid), Phys_chan$, My_chn$, (LONG Termcfg),
                             (REAL Minval), (REAL Maxval), (LONG Units),
                             (REAL Rtdtype), (LONG Resiscfg), (LONG Curexcsrc),
                             (REAL Curexcval), (REAL R0))
```

Input:

Taskid	The task to which to add the channels that this function creates
Phys_chan$	The names of the physical channels to use to create virtual channels
My_chn$	The name(s) to assign to the created virtual channel(s)
Termcfg	The input terminal configuration for the channel
Minval	The minimum value, in **Units**, that you expect to measure
Maxval	The maximum value, in **Units**, that you expect to measure
Units	The units to use to return the measurements
Rtdtype	The type of RTD connected to the channel
Resiscfg	The configuration for resistance measurements
Curexcsrc	The source of excitation
Curexcval	The amount of excitation, in amperes, that the sensor requires
R0	The sensor resistance in ohms at 0 deg C for the Callendar-Van Dusen equation

Return value: 0 = Function was successful
<>0 = Error occured

FNHx_craithmchiex NI-DAQmx: **DAQmxCreateAIThrmstrChanIex**

Purpose: Creates channel(s) that use a thermistor to measure temperature (thermistor requires voltage excitation)

HTBasic Definition:

```
LONG Result=FNHx_craithmchiex((LONG Taskid), Phys_chan$, My_chn$, (REAL Minval),
                              (REAL Maxval), (LONG Units), (LONG Resconfig),
                              (LONG Curexcsrc), (REAL Curexcval), (REAL A), (REAL B),
                              (REAL C))
```

Input:

Taskid	The task to which to add the channels that this function creates
Phys_chan$	The names of the physical channels to use to create virtual channels
My_chn$	The name(s) to assign to the created virtual channel(s)
Minval	The minimum value, in **Units**, that you expect to measure
Maxval	The maximum value, in **Units**, that you expect to measure
Units	The units to use to return the measurements
Resconfig	The configuration for resistance measurements (2-, 3- or 4-wire)
Curexcsrc	The source of excitation
Curexcval	The amount of excitation, in amperes, that the sensor requires
A,B,C	The A, B and C constants from the Steinhart-Hart thermistor equation

Return value: 0 = Function was successful
<>0 = Error occured

FNHx_craithmchvex NI-DAQmx: **DAQmxCreateAIThrmstrChanVex**

Purpose: Creates channel(s) that use a thermistor to measure temperature (thermistor requires voltage excitation)

HTBasic Definition:

```
LONG Result=FNHx_craithmchvex((LONG Taskid), Phys_chan$, My_chn$, (REAL Minval),
                              (REAL Maxval), (LONG Units), (LONG Resconfig),
                              (LONG Volexcsrc), (REAL Volexcval), (REAL A), (REAL B),
                              (REAL C), (REAL R1))
```

Input:

Taskid	The task to which to add the channels that this function creates
Phys_chan$	The names of the physical channels to use to create virtual channels
My_chn$	The name(s) to assign to the created virtual channel(s)
Minval	The minimum value, in **Units**, that you expect to measure
Maxval	The maximum value, in **Units**, that you expect to measure
Units	The units to use to return the measurements
Resconfig	The configuration for resistance measurements (2-, 3- or 4-wire)
Volexcsrc	The source of excitation
Volexcval	The amount of excitation, in volts, that the sensor requires
A,B,C	The A, B and C constants from the Steinhart-Hart thermistor equation
R1	The value, in ohms, of the reference resistor

Return value: 0 = Function was successful
<>0 = Error occured

FNHx_craifrvochan NI-DAQmx: **DAQmxCreateAIFreqVoltageChan**

Purpose: Creates channel(s) that use a frequency-to-voltage converter to measure frequency

HTBasic Definition:

```
LONG Result=FNHx_craifrvochan((LONG Taskid), Phys_chan$, My_chn$, (LONG Termcfg),
                              (REAL Minval), (REAL Maxval), (LONG Units),
                              (REAL Tholdlev), (REAL Hysteresis), Custscalename$)
```

Input:

Taskid	The task to which to add the channels that this function creates
Phys_chan$	The names of the physical channels to use to create virtual channels
My_chn$	The name(s) to assign to the created virtual channel(s)
Termcfg	The input terminal configuration for the channel
Minval	The minimum value, in **Units**, that you expect to measure
Maxval	The maximum value, in **Units**, that you expect to measure
Units	The units to use to return the measurements
Tholdlev	The voltage level at which to recognize waveform repetitions
Hysteresis	Specifies in volts a window below thresholdLevel
Custscalename$	The name of a custom scale to apply to the channel

Return value: 0 = Function was successful
<>0 = Error occured

FNHx_crairesichan NI-DAQmx: **DAQmxCreateAIResistanceChan**

Purpose: Create channel(s) to measure resistance

HTBasic Definition:

```
LONG Result=FNHx_crairesichan((LONG Taskid), Phys_chan$, My_chn$, (LONG Termcfg),
                              (REAL Minval), (REAL Maxval), (LONG Units),
                              (REAL Resconfig), (REAL Curexcsrc), (REAL Curexcval),
                              Custscalename$)
```

Input:

Taskid	The task to which to add the channels that this function creates
Phys_chan$	The names of the physical channels to use to create virtual channels
My_chn$	The name(s) to assign to the created virtual channel(s)
Termcfg	The input terminal configuration for the channel
Minval	The minimum value, in units, that you expect to measure
Maxval	The maximum value, in units, that you expect to measure
Units	The units to use to return the measurements
Resconfig	The configuration for resistance measurements
Curexcsrc	The source of excitation
Curexcval	The amount of excitation, in amperes, that the sensor requires
Custscalename$	The name of a custom scale to apply to the channel

Return value: 0 = Function was successful
<>0 = Error occured

FNHx_craistrgachn NI-DAQmx: **DAQmxCreateAIStrainGageChan**

Purpose: Creates channel(s) to measure strain

HTBasic Definition:

```
LONG Result=FNHx_craistrgachn((LONG Taskid), Phys_chan$, My_chn$, (REAL Minval),
                              (REAL Maxval), (LONG Units), (LONG Strainconfig),
                              (LONG Volexcsrc), (REAL Volexcval), (REAL Gagefactor),
                              (REAL Inbrvol), (REAL Nomgageres), (REAL Poisrat),
                              (REAL Leadwireres), Custscalename$)
```

Input:

Taskid	The task to which to add the channels that this function creates
Phys_chan$	The names of the physical channels to use to create virtual channels
My_chn$	The name(s) to assign to the created virtual channel(s)
Minval	The minimum value, in units, that you expect to measure
Maxval	The maximum value, in units, that you expect to measure
Units	The units to use to return the measurements
Strainconfig	The strain gage bridge configuration
Volexcsrc	The source of excitation
Volexcval	The amount of excitation, in volts, that the sensor requires
Gagefactor	The sensitivity of the strain gages (see DAQmx documentation)
Inbrvol	The bridge output voltage in the unloaded condition
Nomgageres	The resistance, in ohms, of the gages in an unstrained position
Poisrat	The ratio of lateral strain to axial strain in the material in which you measure strain
Leadwireres	The amount, in ohms, of resistance in the lead wires
Custscalename$	The name of a custom scale to apply to the channel

Return value: 0 = Function was successful
<>0 = Error occured

FNHx_craivochwexc NI-DAQmx: **DAQmxCreateAIVoltageChanWithExcit**

Purpose: Creates channel(s) to measure voltage for custom sensors with excitation

HTBasic Definition:

```
LONG Result=FNHx_craivochwexc((LONG Taskid), Phys_chan$, My_chn$, (LONG Termconfig),
                              (REAL Minval), (REAL Maxval), (LONG Units), (LONG Bricfg),
                              (LONG Volexcsrc), (REAL Volexcval), (LONG Useexcforscal),
                              Custscalename$)
```

Input:

Taskid	The task to which to add the channels that this function creates
Phys_chan$	The names of the physical channels to use to create virtual channels
My_chn$	The name(s) to assign to the created virtual channel(s)
Termconfig	The input terminal configuration for the channel
Minval	The minimum value, in **Units**, that you expect to measure
Maxval	The maximum value, in **Units**, that you expect to measure
Units	The units to use to return the measurements

```
 Bricfg            The type of Wheatstone bridge the sensor is
 Volexcsrc         The source of excitation
 Volexcval         The amount of excitation, in volts, that the sensor requires
 Useexcforscal     Specifies whether NI-DAQmx divides the measurement by the excitation
 Custscalename$    The name of a custom scale to apply to the channel

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_craitmpbschn NI-DAQmx: **DAQmxCreateAITempBuiltInSensorChan**

Purpose: Creates channel(s) to measure temperature with a built-in sensor of a terminal block or device

HTBasic Definition:

```
LONG Result=FNHx_craitmpbschn((LONG Taskid), Phys_chan$, My_chn$, (LONG Units))

Input:
 Taskid            The task to which to add the channels that this function creates
 Phys_chan$        The names of the physical channels to use to create virtual channels
 My_chn$           The name(s) to assign to the created virtual channel(s)
 Units             The units to use to return the measurements

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_craiaccelchn NI-DAQmx: **DAQmxCreateAIAccelChan**

Purpose: Creates channel(s) that use an accelerometer to measure acceleration

HTBasic Definition:

```
LONG Result=FNHx_craiaccelchn((LONG Taskid), Phys_chan$, My_chn$, (LONG Termconfig),
                              (REAL Minval), (REAL Maxval), (LONG Units), (LONG Sensit),
                              (LONG Curexcsrc), (REAL Curexcval), Custscalename$)

Input:
 Taskid            The task to which to add the channels that this function creates
 Phys_chan$        The names of the physical channels to use to create virtual channels
 My_chn$           The name(s) to assign to the created virtual channel(s)
 Termconfig        The input terminal configuration for the channel
 Minval            The minimum value, in Units, that you expect to measure
 Maxval            The maximum value, in Units, that you expect to measure
 Units             The units to use to return the measurements
 Sensit            The sensitivity of the sensor
 Curexcsrc         The source of excitation
 Curexcval         The amount of excitation, in ampere, that the sensor requires
 Custscalename$    The name of a custom scale to apply to the channel

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_craimicrchan NI-DAQmx: **DAQmxCreateAIMicrophoneChan**

Purpose: Creates channel(s) that use a microphone to measure sound pressure

HTBasic Definition:

```
LONG Result=FNHx_craimicrchan((LONG Taskid), Phys_chan$, My_chn$, (LONG Termconfig),
                              (LONG Units), (REAL Micsens), (REAL Maxsoundplevel),
                              (LONG Curexcsrc), (REAL Curexcval), Custscalename$)

Input:
 Taskid            The task to which to add the channels that this function creates
 Phys_chan$        The names of the physical channels to use to create virtual channels
 My_chn$           The name(s) to assign to the created virtual channel(s)
 Termconfig        The input terminal configuration for the channel
 Units             The units to use to return the measurements
 Micsens           The sensitivity of the microphone (in millivolts per pascal)
 Maxsoundplevel    The maximum instantaneous sound pressure level you expect to measure.
                    This value is in decibels, referenced to 20 micropascals
 Curexcsrc         The source of excitation
 Curexcval         The amount of excitation, in ampere, that the sensor requires
 Custscalename$    The name of a custom scale to apply to the channel

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_craipslvdtch NI-DAQmx: **DAQmxCreateAIPosLVDTChan**

Purpose: Creates channel(s) that use an LVDT to measure linear position

HTBasic Definition:

```
LONG Result=FNHx_craipslvdtch((LONG Taskid), Phys_chan$, My_chn$, (REAL Minval),
                              (REAL Maxval),(LONG Units), (REAL Sens), (LONG Sensunits),
                              (LONG Volexcsrc), (REAL Volexcval), (REAL Volexcfrq),
                              (LONG Acexcwmode), Custscalename$)
```

```
Input:
 Taskid            The task to which to add the channels that this function creates
 Phys_chan$        The names of the physical channels to use to create virtual channels
 My_chn$           The name(s) to assign to the created virtual channel(s)
 Minval            The minimum value, in Units, that you expect to measure
 Maxval            The maximum value, in Units, that you expect to measure
 Units             The units to use to return the measurements
 Sens              The sensitivity of the sensor
 Sensunits         The units of Sens
 Volexcsrc         The source of excitation
 Volexcval         The amount of excitation, in volts, that the sensor requires
 Volexcfrq         The excitation frequency, in hertz, that the sensor requires
 Acexcwmode        The number of leads on the sensor
 Custscalename$    The name of a custom scale to apply to the channel

Return value:   0 = Function was successful
               <>0 = Error occured
```

FNHx_craipsrvdtch NI-DAQmx: **DAQmxCreateAIPosRVDTChan**

Purpose: Creates channel(s) that use an RVDT to measure angular position

HTBasic Definition:

```
LONG Result=FNHx_craipsrvdtch((LONG Taskid), Phys_chan$, My_chn$, (REAL Minval),
                              (REAL Maxval),(LONG Units), (REAL Sens), (LONG Sensunits),
                              (LONG Volexcsrc), (REAL Volexcval), (REAL Volexcfrq),
                              (LONG Acexcwmode), Custscalename$)
```

```
Input:
 Taskid            The task to which to add the channels that this function creates
 Phys_chan$        The names of the physical channels to use to create virtual channels
 My_chn$           The name(s) to assign to the created virtual channel(s)
 Minval            The minimum value, in Units, that you expect to measure
 Maxval            The maximum value, in Units, that you expect to measure
 Units             The units to use to return the measurements
 Sens              The sensitivity of the sensor
 Sensunits         The units of Sens
 Volexcsrc         The source of excitation
 Volexcval         The amount of excitation, in volts, that the sensor requires
 Volexcfrq         The excitation frequency, in hertz, that the sensor requires
 Acexcwmode        The number of leads on the sensor
 Custscalename$    The name of a custom scale to apply to the channel

Return value:   0 = Function was successful
               <>0 = Error occured
```

FNHx_crtaivoltchn NI-DAQmx: **DAQmxCreateTEDSAIVoltageChan**

Purpose: Creates channel(s) to measure voltage

HTBasic Definition:

```
LONG Result=FNHx_crtaivoltchn((LONG Taskid), Phys_chan$, My_chn$, (LONG Termcfg),
                              (REAL Minval), (REAL Maxval), (LONG Units),
                              Custscalename$)
```

```
Input:
 Taskid            The task to which to add the channels that this function creates
 Phys_chan$        The names of the physical channels to use to create virtual channels
 My_chn$           The name(s) to assign to the created virtual channel(s)
 Termcfg           The input terminal configuration for the channel
 Minval            The minimum value, in units, that you expect to measure
 Maxval            The maximum value, in units, that you expect to measure
 Units             The units to use to return the measurements
 Custscalename$    The name of a custom scale to apply to the channel

Return value:   0 = Function was successful
               <>0 = Error occured
```

FNHx_crtaicurrchn NI-DAQmx: **DAQmxCreateTEDSAICurrentChan**

Purpose: Creates channel(s) to measure current

HTBasic Definition:

```
LONG Result=FNHx_crtaicurrchn((LONG Taskid), Phys_chan$, My_chn$, (LONG Termcfg),
                              (REAL Minval), (REAL Maxval), (LONG Units),
                              (LONG Shuntresloc), (LONG Extshuntresval), Custscalename$)

Input:
 Taskid            The task to which to add the channels that this function creates
 Phys_chan$        The names of the physical channels to use to create virtual channels
 My_chn$           The name(s) to assign to the created virtual channel(s)
 Termcfg           The input terminal configuration for the channel
 Minval            The minimum value, in units, that you expect to measure
 Maxval            The maximum value, in units, that you expect to measure
 Units             The units to use to return the measurements
 Shuntresloc       The location of the shunt resistor
 Extshuntresval    The value, in ohms, of an external shunt resistor
 Custscalename$    The name of a custom scale to apply to the channel

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_crtaithcpchn NI-DAQmx: **DAQmxCreateTEDSAIThrmcplChan**

Purpose: Creates channel(s) that use a thermocouple to measure temperature

HTBasic Definition:

```
LONG Result=FNHx_crtaithcpchn((LONG Taskid), Phys_chan$, My_chn$, (LONG Termcfg),
                              (REAL Minval), (REAL Maxval), (LONG Units),
                              (LONG Cjcsource), (REAL Cjcval), Cjcchannel$)

Input:
 Taskid            The task to which to add the channels that this function creates
 Phys_chan$        The names of the physical channels to use to create virtual channels
 My_chn$           The name(s) to assign to the created virtual channel(s)
 Termcfg           The input terminal configuration for the channel
 Minval            The minimum value, in units, that you expect to measure
 Maxval            The maximum value, in units, that you expect to measure
 Units             The units to use to return the measurements
 Cjcsource         The source of cold junction compensation
 Cjcval            The temperature of the cold junction of the thermocouple if you set
                   Cjcsource to "DAQmx_Val_ConstVal"
 Cjcchannel$       The channel that acquires the temperature of the thermocouple
                   cold-junction if you set Cjcsource to "DAQmx_Val_Chan"

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_crtairtdchn NI-DAQmx: **DAQmxCreateTEDSAIRTDChan**

Purpose: Creates channel(s) that use an RTD to measure temperature

HTBasic Definition:

```
LONG Result=FNHx_crtairtdchn((LONG Taskid), Phys_chan$, My_chn$, (REAL Minval),
                             (REAL Maxval), (LONG Units), (LONG Resconfig),
                             (LONG Curexcsrc), (REAL Curexcval))

Input:
 Taskid            The task to which to add the channels that this function creates
 Phys_chan$        The names of the physical channels to use to create virtual channels
 My_chn$           The name(s) to assign to the created virtual channel(s)
 Minval            The minimum value, in Units, that you expect to measure
 Maxval            The maximum value, in Units, that you expect to measure
 Units             The units to use to return the measurements
 Resconfig         The configuration for resistance measurements
 Curexcsrc         The source of excitation
 Curexcval         The amount of excitation, in amperes, that the sensor requires

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_crtaithchiex NI-DAQmx: **DAQmxCreateTEDSAIThrmstrChanIex**

Purpose: Creates channel(s) that use a thermistor to measure temperature

HTBasic Definition:

```
LONG Result=FNHx_crtaithchiex((LONG Taskid), Phys_chan$, My_chn$, (REAL Minval),
                              (REAL Maxval), (LONG Units), (LONG Resconfig),
                              (LONG Curexcsrc), (REAL Curexcval))

Input:
 Taskid            The task to which to add the channels that this function creates
 Phys_chan$        The names of the physical channels to use to create virtual channels
 My_chn$           The name(s) to assign to the created virtual channel(s)
 Minval            The minimum value, in Units, that you expect to measure
 Maxval            The maximum value, in Units, that you expect to measure
 Units             The units to use to return the measurements
 Resconfig         The configuration for resistance measurements (2-, 3- or 4-wire)
 Curexcsrc         The source of excitation
 Curexcval         The amount of excitation, in amperes, that the sensor requires

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_crtaithchvex NI-DAQmx: **DAQmxCreateTEDSAIThrmstrChanVex**

Purpose: Creates channel(s) that use a thermistor to measure temperature

HTBasic Definition:

```
LONG Result=FNHx_crtaithchvex((LONG Taskid), Phys_chan$, My_chn$, (REAL Minval),
                              (REAL Maxval), (LONG Units), (LONG Resconfig),
                              (LONG Volexcsrc), (REAL Volexcval), (REAL R1))

Input:
 Taskid            The task to which to add the channels that this function creates
 Phys_chan$        The names of the physical channels to use to create virtual channels
 My_chn$           The name(s) to assign to the created virtual channel(s)
 Minval            The minimum value, in Units, that you expect to measure
 Maxval            The maximum value, in Units, that you expect to measure
 Units             The units to use to return the measurements
 Resconfig         The configuration for resistance measurements (2-, 3- or 4-wire)
 Volexcsrc         The source of excitation
 Volexcval         The amount of excitation, in volts, that the sensor requires
 R1                The value, in ohms, of the reference resistor

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_crtairesichn NI-DAQmx: **DAQmxCreateTEDSAIResistanceChan**

Purpose: Creates channel(s) to measure resistance

HTBasic Definition:

```
LONG Result=FNHx_crtairesichn((LONG Taskid), Phys_chan$, My_chn$, (REAL Minval),
                              (REAL Maxval), (LONG Units), (REAL Resconfig),
                              (REAL Curexcsrc), (REAL Curexcval), Custscalename$)

Input:
 Taskid            The task to which to add the channels that this function creates
 Phys_chan$        The names of the physical channels to use to create virtual channels
 My_chn$           The name(s) to assign to the created virtual channel(s)
 Minval            The minimum value, in units, that you expect to measure
 Maxval            The maximum value, in units, that you expect to measure
 Units             The units to use to return the measurements
 Resconfig         The configuration for resistance measurements
 Curexcsrc         The source of excitation
 Curexcval         The amount of excitation, in amperes, that the sensor requires
 Custscalename$    The name of a custom scale to apply to the channel

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_crtaistrgchn NI-DAQmx: **DAQmxCreateTEDSAIStrainGageChan**

Purpose: Creates channel(s) to measure strain

HTBasic Definition:

```
LONG Result=FNHx_crtaistrgchn((LONG Taskid), Physchn$, Mychn$, (REAL Minv), (REAL Maxv),
                              (LONG Units), (LONG Volexcsrc), (REAL Volexcval),
                              (REAL Inibrivol), (REAL Leadwireres), Custscalename$)
```

Input:

Taskid	The task to which to add the channels that this function creates
Physchn$	The names of the physical channels to use to create virtual channels
Mychn$	The name(s) to assign to the created virtual channel(s)
Minv	The minimum value, in units, that you expect to measure
Maxv	The maximum value, in units, that you expect to measure
Units	The units to use to return the measurements
Volexcsrc	The source of excitation
Volexcval	The amount of excitation, in volts, that the sensor requires
Inibrivol	The bridge output voltage in the unloaded condition
Leadwireres	The amount, in ohms, of resistance in the lead wires
Custscalename$	The name of a custom scale to apply to the channel

Return value: 0 = Function was successful
<>0 = Error occured

FNHx_crtaivochwex NI-DAQmx: **DAQmxCreateTEDSAIVoltageChanWithExcit**

Purpose: Creates channel(s) to measure voltage (for custom sensors that require excitation)

HTBasic Definition:

```
LONG Result=FNHx_crtaivochwex((LONG Taskid), Phys_chan$, My_chn$, (LONG Termconfig),
                              (REAL Minval), (REAL Maxval), (LONG Units),
                              (LONG Volexcsrc), (REAL Volexcval), Custscalename$)
```

Input:

Taskid	The task to which to add the channels that this function creates
Phys_chan$	The names of the physical channels to use to create virtual channels
My_chn$	The name(s) to assign to the created virtual channel(s)
Termconfig	The input terminal configuration for the channel
Minval	The minimum value, in **Units**, that you expect to measure
Maxval	The maximum value, in **Units**, that you expect to measure
Units	The units to use to return the measurements
Volexcsrc	The source of excitation
Volexcval	The amount of excitation, in volts, that the sensor requires
Custscalename$	The name of a custom scale to apply to the channel

Return value: 0 = Function was successful
<>0 = Error occured

FNHx_craitacclchn NI-DAQmx: **DAQmxCreateTEDSAIAccelChan**

Purpose: Creates channel(s) that use an accelerometer to measure acceleration

HTBasic Definition:

```
LONG Result=FNHx_craitacclchn((LONG Taskid), Phys_chan$, My_chn$, (LONG Termconfig),
                              (REAL Minval), (REAL Maxval), (LONG Units),
                              (LONG Curexcsrc), (REAL Curexcval), Custscalename$)
```

Input:

Taskid	The task to which to add the channels that this function creates
Phys_chan$	The names of the physical channels to use to create virtual channels
My_chn$	The name(s) to assign to the created virtual channel(s)
Termconfig	The input terminal configuration for the channel
Minval	The minimum value, in **Units**, that you expect to measure
Maxval	The maximum value, in **Units**, that you expect to measure
Units	The units to use to return the measurements
Curexcsrc	The source of excitation
Curexcval	The amount of excitation, in ampere, that the sensor requires
Custscalename$	The name of a custom scale to apply to the channel

Return value: 0 = Function was successful
<>0 = Error occured

FNHx_crtaimicchan NI-DAQmx: **DAQmxCreateTEDSAIMicrophoneChan**

Purpose: Creates channel(s) that use a microphone to measure sound pressure

HTBasic Definition:

```
LONG Result=FNHx_crtaimicchan((LONG Taskid), Phys_chan$, My_chn$, (LONG Termconfig),
                              (LONG Units), (REAL Maxsoundplevel), (LONG Curexcsrc),
                              (REAL Curexcval), Custscalename$)

Input:
 Taskid            The task to which to add the channels that this function creates
 Phys_chan$        The names of the physical channels to use to create virtual channels
 My_chn$           The name(s) to assign to the created virtual channel(s)
 Termconfig        The input terminal configuration for the channel
 Units             The units to use to return the measurements
 Maxsoundplevel    The maximum instantaneous sound pressure level you expect to measure.
                    This value is in decibels, referenced to 20 micropascals
 Curexcsrc         The source of excitation
 Curexcval         The amount of excitation, in ampere, that the sensor requires
 Custscalename$    The name of a custom scale to apply to the channel

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_crtaiplvdtch NI-DAQmx: **DAQmxCreateTEDSAIPosLVDTChan**

Purpose: Creates channel(s) that use an LVDT to measure linear position

HTBasic Definition:

```
LONG Result=FNHx_crtaiplvdtch((LONG Taskid), Physchn$, Mychn$, (REAL Minv), (REAL Maxv),
                              (LONG Units), (LONG Volexcsrc), (REAL Volexcval),
                              (REAL Volexcfrq), (LONG Acexcwmode), Custscalename$)

Input:
 Taskid            The task to which to add the channels that this function creates
 Physchn$          The names of the physical channels to use to create virtual channels
 Mychn$            The name(s) to assign to the created virtual channel(s)
 Minv              The minimum value, in Units, that you expect to measure
 Maxv              The maximum value, in Units, that you expect to measure
 Units             The units to use to return the measurements
 Volexcsrc         The source of excitation
 Volexcval         The amount of excitation, in volts, that the sensor requires
 Volexcfrq         The excitation frequency, in hertz, that the sensor requires
 Acexcwmode        The number of leads on the sensor
 Custscalename$    The name of a custom scale to apply to the channel

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_crtaiprvdtch NI-DAQmx: **DAQmxCreateTEDSAIPosRVDTChan**

Purpose: Creates channel(s) that use an RVDT to measure angular position

HTBasic Definition:

```
LONG Result=FNHx_crtaiprvdtch((LONG Taskid), Physchn$, Mychn$, (REAL Minv), (REAL Maxv),
                              (LONG Units), (LONG Volexcsrc), (REAL Volexcval),
                              (REAL Volexcfrq), (LONG Acexcwmode), Custscalename$)

Input:
 Taskid            The task to which to add the channels that this function creates
 Physchn$          The names of the physical channels to use to create virtual channels
 Mychn$            The name(s) to assign to the created virtual channel(s)
 Minv              The minimum value, in Units, that you expect to measure
 Maxv              The maximum value, in Units, that you expect to measure
 Units             The units to use to return the measurements
 Volexcsrc         The source of excitation
 Volexcval         The amount of excitation, in volts, that the sensor requires
 Volexcfrq         The excitation frequency, in hertz, that the sensor requires
 Acexcwmode        The number of leads on the sensor
 Custscalename$    The name of a custom scale to apply to the channel

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_craovoltchan NI-DAQmx: **DAQmxCreateAOVoltageChan**

Purpose: Creates channel(s) to generate voltage

HTBasic-Definition:

```
LONG Result=FNHx_craovoltchan((LONG Taskid), Phys_chan$, My_chn$, (REAL Minval),
                              (REAL Maxval), (LONG Units), Custscalename$)

Input:
 Taskid            The task to which to add the channels that this function creates
 Phys_chan$        The names of the physical channels to use to create virtual channels
 My_chn$           The name(s) to assign to the created virtual channel(s)
 Minval            The minimum value, in units, that you expect to measure
 Maxval            The maximum value, in units, that you expect to measure
 Units             The units to use to return the measurements
 Custscalename$    The name of a custom scale to apply to the channel

Return value:   0 = Function was successful
              <>0 = Error occured
```

Example:

```
 INTEGER Iresult1
 LONG Result, Taskid, Units
 REAL V_min, V_max
 DIM Tmp$[255], Devchn$[255]
 !
 Units=FNHx_get_constant("DAQmx_Val_Volts",Iresult1)         ! constant 1
 V_min=-10.0
 V_max=10.0
 Tmp$=""
 Devchn$="Dev1/ao0"
 Result=FNHx_craovoltchan((Taskid),Devchn$,Tmp$,(V_min),(V_max),(Units),Tmp$)
```

FNHx_craocurrchan NI-DAQmx: **DAQmxCreateAOCurrentChan**

Purpose: Creates channel(s) to generate current

HTBasic-Definition:

```
LONG Result=FNHx_craocurrchan((LONG Taskid), Phys_chan$, My_chn$, (REAL Minval),
                              (REAL Maxval), (LONG Units), Custscalename$)

Input:
 Taskid            The task to which to add the channels that this function creates
 Phys_chan$        The names of the physical channels to use to create virtual channels
 My_chn$           The name(s) to assign to the created virtual channel(s)
 Minval            The minimum value, in units, that you expect to measure
 Maxval            The maximum value, in units, that you expect to measure
 Units             The units to use to return the measurements
 Custscalename$    The name of a custom scale to apply to the channel

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_createdichan NI-DAQmx: **DAQmxCreateDIChan**

Purpose: Creates channel(s) to measure digital signals

HTBasic Definition:

```
LONG Result=FNHx_createdichan((LONG Taskid), Lines$, My_chn$, (LONG Linegrouping))
```

```
Input:
 Taskid            The task to which to add the channels that this function creates
 Lines$            The names of the digital lines used to create a virtual channel
 My_chn$           The name(s) to assign to the created virtual channel(s)
 Linegrouping      Specifies whether to group digital lines into one or more virtual
                   channels

Return value:   0 = Function was successful
              <>0 = Error occured
```

Example:

```
 INTEGER Iresult1
 LONG Result, Taskid, Linegrouping
 DIM Lines$[255], Tmp$[255]
 !
 Linegrouping=FNHx_get_constant("DAQmx_Val_ChanForAllLines",Iresult1)
 Lines$="Dev1/port0/line0:3"
 Tmp$=""
 Result=FNHx_createdichan((Taskid),Lines$,Tmp$,(Linegrouping))
```

FNHx_createdochan NI-DAQmx: **DAQmxCreateDOChan**

Purpose: Creates channel(s) to generate digital signals

HTBasic Definition:

```
LONG Result=FNHx_createdochan((LONG Taskid), Lines$, My_chn$, (LONG Linegrouping))
```

```
Input:
 Taskid            The task to which to add the channels that this function creates
 Lines$            The names of the digital lines used to create a virtual channel
 My_chn$           The name(s) to assign to the created virtual channel(s)
 Linegrouping      Specifies whether to group digital lines into one or more virtual
                   channels

Return value:   0 = Function was successful
              <>0 = Error occured
```

Example:

```
 INTEGER Iresult1
 LONG Result, Taskid, Linegrouping
 DIM Lines$[255], Tmp$[255]
 !
 Linegrouping=FNHx_get_constant("DAQmx_Val_ChanForAllLines",Iresult1)
 Lines$="Dev1/port0/line4:7"
 Tmp$=""
Result=FNHx_createdochan((Taskid),Lines$,Tmp$,(Linegrouping))
```

FNHx_crcifreqchan — NI-DAQmx: DAQmxCreateCIFreqChan

Purpose: Creates a channel to measure the frequency of a digital signal and adds the channel to the task

HTBasic Definition:

```
LONG Result=FNHx_crcifreqchan((LONG Taskid), Counter$, My_chn$, (REAL Minval),
                              (REAL Maxval), (LONG Units), (LONG Edge),
                              (LONG Measmeth), (REAL Meastime), (LONG Divisor), Csn$)
```

Input:

Taskid	The task to which to add the channels that this function creates
Counter$	The name of the counter to use to create virtual channels
My_chn$	The name(s) to assign to the created virtual channel(s)
Minval	The minimum value, in units, that you expect to measure
Maxval	The maximum value, in units, that you expect to measure
Units	The units to use to return the measurements
Edge	Specifies between which edges to measure the frequency or period of the signal
Measmeth	The method used to calculate the period or frequency of the signal
Meastime	The length of time to measure the frequency or period of a digital signal, when measMethod is "DAQmx_Val_HighFreq2Ctr"
Divisor	The value by which to divide the input signal, when measMethod is "DAQmx_Val_LargeRng2Ctr"
Csn$	The name of a custom scale to apply to the channel

Return value: 0 = Function was successful
<>0 = Error occured

FNHx_crciperichan — NI-DAQmx: DAQmxCreateCIPeriodChan

Purpose: Creates a channel to measure the period of a digital signal and adds the channel to the task

HTBasic Definition:

```
LONG Result=FNHx_crciperichan((LONG Taskid), Counter$, My_chn$, (REAL Minval),
                              (REAL Maxval), (LONG Units), (LONG Edge),
                              (LONG Measmeth), (REAL Meastime), (LONG Divisor), Csn$)
```

Input:

Taskid	The task to which to add the channels that this function creates
Counter$	The name of the counter to use to create virtual channels
My_chn$	The name(s) to assign to the created virtual channel(s)
Minval	The minimum value, in units, that you expect to measure
Maxval	The maximum value, in units, that you expect to measure
Units	The units to use to return the measurements
Edge	Specifies between which edges to measure the frequency or period of the signal
Measmeth	The method used to calculate the period or frequency of the signal
Meastime	The length of time to measure the frequency or period of a digital signal, when measMethod is "DAQmx_Val_HighFreq2Ctr"
Divisor	The value by which to divide the input signal, when measMethod is "DAQmx_Val_LargeRng2Ctr"
Csn$	The name of a custom scale to apply to the channel

Return value: 0 = Function was successful
<>0 = Error occured

FNHx_crcicntedgch NI-DAQmx: **DAQmxCreateCICountEdgesChan**

Purpose: Creates a channel to count the number of rising or falling edges of a digital signal

HTBasic Definition:

```
LONG Result=FNHx_crcicntedgch((LONG Taskid), Counter$, My_chn$, (LONG Edge),
                              (LONG Initcount), (LONG Countdirection))
```

```
Input:
 Taskid            The task to which to add the channels that this function creates
 Lines$            The names of the digital lines used to create a virtual channel
 My_chn$           The name(s) to assign to the created virtual channel(s)
 Edge              Specifies on which edges of the input signal to increment or
                   decrement the count
 Initcount         The value from which to start counting
 Countdirection    Specifies whether to increment or decrement the counter on each edge

Return value:   0 = Function was successful
              <>0 = Error occured
```

Example:

```
 INTEGER Iresult1, Iresult2
 LONG Result, Taskid, Edge, Initcount, Countdir
 DIM Counter$[255], Tmp$[255]
 !
 Counter$="Dev1/ctr0"                                  ! Counter 0
 Tmp$=""
 Edge=FNHx_get_constant("DAQmx_Val_Rising",Iresult1)
 Initcount=0
 Countdir=FNHx_get_constant("DAQmx_Val_CountUp",Iresult2)
 Result=FNHx_crcicntedgch((Taskid),Counter$,Tmp$,(Edge),(Initcount),(Countdir))
```

FNHx_crciplswchan NI-DAQmx: **DAQmxCreateCIPulseWidthChan**

Purpose: Creates a channel to measure the width of a digital pulse and adds the channel to the task

HTBasic Definition:

```
LONG Result=FNHx_crciplswchan((LONG Taskid), Counter$, My_chn$, (REAL Minval),
                              (REAL Maxval), (LONG Units), (LONG Startingedge), Csn$)
```

```
Input:
 Taskid            The task to which to add the channels that this function creates
 Counter$          The name of the counter to use to create virtual channels
 My_chn$           The name(s) to assign to the created virtual channel(s)
 Minval            The minimum value, in units, that you expect to measure
 Maxval            The maximum value, in units, that you expect to measure
 Units             The units to use to return the measurements
 Startingedge      Specifies on which edge to begin measuring pulse width
 Csn$              The name of a custom scale to apply to the channel

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_crcisemperch NI-DAQmx: **DAQmxCreateCISemiPeriodChan**

Purpose: Creates a channel to measure the width of a digital pulse and adds the channel to the task

HTBasic Definition:

```
LONG Result=FNHx_crcisemperch((LONG Taskid), Counter$, My_chn$, (REAL Minval),
                              (REAL Maxval), (LONG Units), Csn$)
```

```
Input:
 Taskid            The task to which to add the channels that this function creates
 Counter$          The name of the counter to use to create virtual channels
 My_chn$           The name(s) to assign to the created virtual channel(s)
 Minval            The minimum value, in units, that you expect to measure
 Maxval            The maximum value, in units, that you expect to measure
 Units             The units to use to return the measurements
 Csn$              The name of a custom scale to apply to the channel

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_crcitwedsech NI-DAQmx: **DAQmxCreateCITwoEdgeSepChan**

Purpose: Creates a channel that measures the amount of time between the rising or falling edge of one digital signal and the rising or falling edge of another digital signal

HTBasic Definition:

```
LONG Result=FNHx_crcitwedsech((LONG Taskid), Counter$, My_chn$, (REAL Minval),
                              (REAL Maxval), (LONG Units), (LONG Firstedge),
                              (LONG Secondedge), Csn$)
```

Input:

Taskid	The task to which to add the channels that this function creates
Counter$	The name of the counter to use to create virtual channels
My_chn$	The name(s) to assign to the created virtual channel(s)
Minval	The minimum value, in units, that you expect to measure
Maxval	The maximum value, in units, that you expect to measure
Units	The units to use to return the measurements
Firstedge	Specifies on which edge of the first signal to start each measurement
Secondedge	Specifies on which edge of the first signal to stop each measurement
Csn$	The name of a custom scale to apply to the channel

Return value: 0 = Function was successful
<>0 = Error occured

FNHx_crcilinencch NI-DAQmx: **DAQmxCreateCILinEncoderChan**

Purpose: Creates a channel that uses a linear encoder to measure linear position

HTBasic Definition:

```
LONG Result=FNHx_crcilinencch((LONG Taskid), Counter$, My_chn$, (LONG Decodingtype),
                              (LONG Zidxenable), (REAL Zidxval), (LONG Zidxphase),
                              (LONG Units), (REAL Distppulse), (REAL Initpos), Csn$)
```

Input:

Taskid	The task to which to add the channels that this function creates
Counter$	The name of the counter to use to create virtual channels
My_chn$	The name(s) to assign to the created virtual channel(s)
Decodingtype	Specifies how to count and interpret the pulses that the encoder generates on signal A and signal B
Zidxenable	Specifies whether to enable z indexing for the measurement
Zidxval	The value, in units, to which to reset the measurement when signal Z is high and signal A and signal B are at the states you specify with **Zidxphase**
Zidxphase	The states at which signal A and signal B must be while signal Z is high for NI-DAQmx to reset the measurement
Units	The units to use to return the measurements
Distppulse	The distance measured for each pulse the encoder generates (in **Units**)
Initpos	The position of the encoder when the measurement begins (in **Units**)
Csn$	The name of a custom scale to apply to the channel

Return value: 0 = Function was successful
<>0 = Error occured

FNHx_crcianencchn NI-DAQmx: **DAQmxCreateCIAngEncoderChan**

Purpose: Creates a channel that uses an angular encoder to measure angular position

HTBasic Definition:

```
LONG Result=FNHx_crcianencchn((LONG Taskid), Counter$, My_chn$, (LONG Decodingtype),
                              (LONG Zidxenable), (REAL Zidxval), (LONG Zidxphase),
                              (LONG Units), (REAL Pulsesperref), (REAL Initang), Csn$)
```

Input:

Taskid	The task to which to add the channels that this function creates
Counter$	The name of the counter to use to create virtual channels
My_chn$	The name(s) to assign to the created virtual channel(s)
Decodingtype	Specifies how to count and interpret the pulses that the encoder generates on signal A and signal B
Zidxenable	Specifies whether to enable z indexing for the measurement
Zidxval	The value, in units, to which to reset the measurement when signal Z is high and signal A and signal B are at the states you specify with **Zidxphase**
Zidxphase	The states at which signal A and signal B must be while signal Z is high for NI-DAQmx to reset the measurement
Units	The units to use to return the measurements

```
 Pulsesperref      The number of pulses the encoder generates per revolution
 Initang           The starting angle of the encoder when measurement begins (in Units)
 Csn$              The name of a custom scale to apply to the channel

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_crcigpstschn — NI-DAQmx: **DAQmxCreateCIGPSTimestampChan**

Purpose: Creates a channel that uses a special-purpose counter to take a timestamp and synchronizes that counter to a GPS receiver

HTBasic Definition:

```
LONG Result=FNHx_crcigpstschn((LONG Taskid), Counter$, My_chn$, (LONG Units),
                              (LONG Syncmethod), Csn$)

Input:
 Taskid            The task to which to add the channels that this function creates
 Counter$          The name of the counter to use to create virtual channels
 My_chn$           The name(s) to assign to the created virtual channel(s)
 Units             The units to use to return the measurements
 Syncmethod        The method to use to synchronize the counter to a GPS receiver
 Csn$              The name of a custom scale to apply to the channel

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_crcopulchfrq — NI-DAQmx: **DAQmxCreateCOPulseChanFreq**

Purpose: Creates channel(s) to generate digital pulses that **Freq** and **Dutycycle** define

HTBasic Definition:

```
LONG Result=FNHx_crcopulchfrq((LONG Taskid), Counter$, My_chn$, (LONG Units),
                              (LONG Idlestate), (REAL Initdelay), (REAL Freq),
                              (REAL Dutycycle))

Input:
 Taskid            The task to which to add the channels that this function creates
 Counter$          The name of the counter to use to create virtual channels
 My_chn$           The name(s) to assign to the created virtual channel(s)
 Units             The units to use to return the measurements
 Idlestate         The resting state of the output terminal
 Initdelay         The amount of time in seconds to wait before generating the 1st pulse
 Freq              The frequency at which to generate pulses
 Dutycycle         The width of the pulse divided by the pulse period.
                   NI-DAQmx uses this ratio, combined with frequency, to determine pulse
                   width and the interval between pulses

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_crcopulchtim — NI-DAQmx: **DAQmxCreateCOPulseChanTime**

Purpose: Creates channel(s) to generate digital pulses defined by the amount of time the pulse is at a high state and the amount of time the pulse is at a low state

HTBasic Definition:

```
LONG Result=FNHx_crcopulchtim((LONG Taskid), Counter$, My_chn$, (LONG Units),
                              (LONG Idlestate), (REAL Initdelay), (REAL Lowtime),
                              (REAL Hightime))

Input:
 Taskid            The task to which to add the channels that this function creates
 Counter$          The name of the counter to use to create virtual channels
 My_chn$           The name(s) to assign to the created virtual channel(s)
 Units             The units to use to return the measurements
 Idlestate         The resting state of the output terminal
 Initdelay         The amount of time in seconds to wait before generating the 1st pulse
 Lowtime           The amount of time the pulse is low, in seconds
 Hightime          The amount of time the pulse is high, in seconds

Return value:   0 = Function was successful
              <>0 = Error occured
```

Purpose: Creates channel(s) to generate digital pulses defined by the number of timebase ticks that the pulse is at a high state and the number of timebase ticks that the pulse is at a low state

HTBasic Definition:

```
LONG Result=FNHx_crcopulchtck((LONG Taskid), Counter$, My_chn$, Srcterminal$,
                              (LONG Idlestate), (REAL Initdelay), (REAL Lowticks),
                              (REAL Highticks))
```

Input:

Taskid	The task to which to add the channels that this function creates
Counter$	The name of the counter to use to create virtual channels
Srcterminal$	The terminal to which you connect an external timebase
Idlestate	The resting state of the output terminal
Initdelay	The amount of time in seconds to wait before generating the 1st pulse
Lowticks	The number of timebase ticks that the pulse is low
Highticks	The number of timebase ticks that the pulse is high

Return value: 0 = Function was successful
<>0 = Error occured

FNHx_craivlrmschn NI-DAQmx 8.6: **DAQmxCreateAIVoltageRMSChan**

Purpose: Creates channel(s) to measure RMS voltage and adds the channel(s) to the task you specify with **Taskid**

HTBasic Definition:

```
LONG Result=FNHx_craivlrmschn((LONG Taskid), Phys_chan$, My_chn$, (LONG Termcfg),
                              (REAL Minval), (REAL Maxval), (LONG Units), Custscalname$)

Input:
 Taskid            The task to which to add the channels that this function creates
 Phys_chan$        The names of the physical channels to use to create virtual channels
 My_chn$           The name(s) to assign to the created virtual channel(s)
 Termcfg           The input terminal configuration for the channel
 Minval            The minimum value, in Units, that you expect to measure
 Maxval            The maximum value, in Units, that you expect to measure
 Units             The units to use to return the voltage measurements
 Custscalname$     The name of a custom scale to apply to the channel

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_craicurmschn NI-DAQmx 8.6: **DAQmxCreateAICurrentRMSChan**

Purpose: Creates channel(s) for RMS current measurement and adds the channel(s) to the task you specify with **Taskid**

HTBasic Definition:

```
LONG Result=FNHx_craicurmschn((LONG Taskid), Phys_chan$, My_chn$, (LONG Termcfg),
                              (REAL Minval), (REAL Maxval), (LONG Units),
                              (REAL Shuntresloc), (REAL Extshuntresval), Custscalename$)

Input:
 Taskid            The task to which to add the channels that this function creates
 Phys_chan$        The names of the physical channels to use to create virtual channels
 My_chn$           The name(s) to assign to the created virtual channel(s)
 Termcfg           The input terminal configuration for the channel
 Minval            The minimum value, in Units, that you expect to measure
 Maxval            The maximum value, in Units, that you expect to measure
 Units             The units to use to return the voltage measurements
 Shuntresloc       The location of the shunt resistor
 Extshuntresval    The value, in ohms, of an external shunt resistor
 Custscalename$    The name of a custom scale to apply to the channel

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

Purpose: Creates a channel for continually generating a waveform on the selected physical channel

HTBasic Definition:

```
LONG Result=FNHx_craofungnchn((LONG Taskid), Phys_chan$, My_chn$, (LONG Type),
                              (REAL Freq), (REAL Amplitude), (LONG Offset))

Input:
 Taskid            The task to which to add the channels that this function creates
 Phys_chan$        The names of the physical channels to use to create virtual channels
 My_chn$           The name(s) to assign to the created virtual channel(s)
 Type              Specifies the kind of waveform to generate
 Freq              The frequency of the waveform to generate in Hertz
 Amplitude         The zero-to-peak amplitude of the waveform to generate in volts
 Offset            The voltage offset of the waveform to generate

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

Timing Functions

FNHx_cfgsamclktim NI-DAQmx: **DAQmxCfgSampClkTiming**

Purpose: Sets the source of the Sample Clock, the rate of the Sample Clock, and the number of samples to acquire or generate

HTBasic Definition:

```
LONG Result=FNHx_cfgsamclktim((LONG Taskid), Source$, (REAL Rate), (LONG Activeedge),
                              (LONG Samplemode), (LONG Sampsperchan))

Input:
 Taskid            The task to which to add the channels that this function creates
 Source$           The source terminal of the Sample Clock
 Rate              The sampling rate in samples per second
 Activeedge        Specifies on which edge of the clock to acquire or generate samples
 Samplemode        Specifies whether the task acquires or generates samples continuously
                   or if it acquires or generates a finite number of samples
 Sampsperchan      The number of samples to acquire or generate for each channel in the
                   task if Samplemode is "DAQmx_Val_FiniteSamps". If Samplemode is
                   "DAQmx_Val_ContSamps", NI-DAQmx uses this value to determine the
                   buffer size.

Return value:   0 = Function was successful
              <>0 = Error occured
```

Example:

```
 INTEGER Iresult1, Iresult2
 LONG Result, Taskid, Sampsperchn, Actedge, Smode
 REAL Rate
 DIM Tmp$[255], Devchn$[255]
 !
 Sampsperchn=100                                    ! Samples per channel (max. 32767)
 Actedge=FNHx_get_constant("DAQmx_Val_Rising",Iresult1)    ! get constant 1
 Smode=FNHx_get_constant("DAQmx_Val_ContSamps",Iresult2)   ! get constant 2
 Rate=20000.0                                              ! 20.000 Samples / s
 Result=FNHx_cfgsamclktim((Taskid),Tmp1$,(Rate),(Actedge),(Smode),(Sampsperchn))
```

FNHx_cfgplsclktim NI-DAQmx 8.5: **DAQmxCfgPipelinedSampClkTiming**

Purpose: Sets the source of the Sample Clock, the rate of the Sample Clock, and the number of samples to acquire or generate

HTBasic Definition:

```
LONG Result=FNHx_cfgplsclktim((LONG Taskid), Source$, (REAL Rate), (LONG Activeedge),
                              (LONG Samplemode), (LONG Sampsperchan))

Input:
 Taskid            The task to which to add the channels that this function creates
 Source$           The source terminal of the Sample Clock
 Rate              The sampling rate in samples per second
 Activeedge        Specifies on which edge of the clock to acquire or generate samples
 Samplemode        Specifies whether the task acquires or generates samples continuously
                   or if it acquires or generates a finite number of samples
 Sampsperchan      The number of samples to acquire or generate for each channel in the
                   task if Samplemode is "DAQmx_Val_FiniteSamps". If Samplemode is
                   "DAQmx_Val_ContSamps", NI-DAQmx uses this value to determine the
                   buffer size.

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This timing type allows handshaking using the Pause trigger, the Ready for Transfer event, or the Data Active event. Refer to the device documentation for more information.

This timing type is supported only by the NI 6536 and NI 6537 and is available with NI-DAQmx 8.5 and higher only.

Purpose: Determines the number of digital samples to acquire or generate using digital handshaking between the device and a peripheral device

HTBasic Definition:

```
LONG Result=FNHx_cfghshtiming((LONG Taskid), (LONG Samplemode), (LONG Sampsperchn_msb),
                              (LONG Sampsperchn_lsb))

Input:
 Taskid             The task to which to add the channels that this function creates
 Samplemode         Specifies whether the task acquires or generates samples continuously
                    or if it acquires or generates a finite number of samples
 Sampsperchn_msb    Upper 32 Bit of Samples per channel variable (64 Bit)
 Sampsperchn_lsb    Lower 32 Bit of Samples per channel variable (64 Bit)
                    The number of samples to acquire or generate for each channel in the
                    task if Samplemode is "DAQmx_Val_FiniteSamps". If Samplemode is
                    "DAQmx_Val_ContSamps", NI-DAQmx uses this value to determine the
                    buffer size

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_cfgbhtimiclk NI-DAQmx: **DAQmxCfgBurstHandshakingTimingImportClock**

Purpose: Configures when the DAQ device transfers data to a peripheral device, using an imported sample clock to control burst handshaking timing

HTBasic Definition:

```
LONG Result=FNHx_cfgbhtimiclk((LONG Taskid), (LONG Samplemode), (LONG Sampsperchn_msb),
                              (LONG Sampsperchn_lsb), (REAL Sampclockrate), Sampclocksrc$,
                              (LONG Sampclkactedge), (LONG Pausewhen), (LONG Rdyevntactlev))
```

Input:

Taskid	The task to which to add the channels that this function creates
Samplemode	Specifies whether the task acquires or generates samples continuously or if it acquires or generates a finite number of samples
Sampsperchn_msb	Upper 32 Bit of Samples per channel variable (64 Bit)
Sampsperchn_lsb	Lower 32 Bit of Samples per channel variable (64 Bit) The number of samples to acquire or generate for each channel in the task if **Samplemode** is "DAQmx_Val_FiniteSamps". If **Samplemode** is "DAQmx_Val_ContSamps", NI-DAQmx uses this value to determine the buffer size
Sampclockrate	Specifies the sampling rate in samples per channel per second
Sampclocksrc$	Specifies the terminal of the signal to use as the Sample Clock
Sampclkactedge	Specifies on which edge of a clock pulse sampling takes place
Pausewhen	Specifies whether the task pauses while the signal is high or low
Rdyevntactlev	Specifies the polarity for the Ready for Transfer event

Return value: 0 = Function was successful
<>0 = Error occured

FNHx_cfgbhtimeclk NI-DAQmx: **DAQmxCfgBurstHandshakingTimingExportClock**

Purpose: Configures when the DAQ device transfers data to a peripheral device, using the DAQ device's onboard sample clock to control burst handshaking timing

HTBasic Definition:

```
LONG Result=FNHx_cfgbhtimeclk((LONG Taskid), (LONG Samplemode), (LONG Sampsperchn_msb),
                              (LONG Sampsperchn_lsb), (REAL Sampclockrate), Sampclkoutterm$,
                              (LONG Sampclkpulspol), (LONG Pausewhen), (LONG Rdyevntactlev))
```

Input:

Taskid	The task to which to add the channels that this function creates
Samplemode	Specifies whether the task acquires or generates samples continuously or if it acquires or generates a finite number of samples
Sampsperchn_msb	Upper 32 Bit of Samples per channel variable (64 Bit)
Sampsperchn_lsb	Lower 32 Bit of Samples per channel variable (64 Bit) The number of samples to acquire or generate for each channel in the task if **Samplemode** is "DAQmx_Val_FiniteSamps". If **Samplemode** is "DAQmx_Val_ContSamps", NI-DAQmx uses this value to determine the buffer size
Sampclockrate	Specifies the sampling rate in samples per channel per second
Sampclkoutterm$	Specifies the terminal to which to route the Sample Clock
Sampclkpulspol	Specifies if the polarity for the exported sample clock is active high or active low
Pausewhen	Specifies whether the task pauses while the signal is high or low
Rdyevntactlev	Specifies the polarity for the Ready for Transfer event

Return value: 0 = Function was successful
<>0 = Error occured

FNHx_cfgchgdettim — NI-DAQmx: DAQmxCfgChangeDetectionTiming

Purpose: Configures the task to acquire samples on the rising and/or falling edges of the lines or ports you specify

HTBasic Definition:

```
LONG Result=FNHx_cfgchgdettim((LONG Taskid), Risedgechan$, Faledgechan$, (LONG Smode),
                              (LONG Sampsperchn_msb), (LONG Sampsperchn_lsb))
```

Input:

Taskid	The task to which to add the channels that this function creates
Risedgechan$	The names of the digital lines or ports on which to detect rising edges
Faledgechan$	The names of the digital lines or ports on which to detect falling edges
Smode	Specifies whether the task acquires or generates samples continuously or if it acquires or generates a finite number of samples
Sampsperchn_msb	Upper 32 Bit of Samples per channel variable (64 Bit)
Sampsperchn_lsb	Lower 32 Bit of Samples per channel variable (64 Bit) The number of samples to acquire or generate for each channel in the task if **Samplemode** is "DAQmx_Val_FiniteSamps". If **Samplemode** is "DAQmx_Val_ContSamps", NI-DAQmx uses this value to determine the buffer size

Return value: 0 = Function was successful
<>0 = Error occured

FNHx_cfgimptiming — NI-DAQmx: DAQmxCfgImplicitTiming

Purpose: Sets only the number of samples to acquire or generate without specifying timing

HTBasic Definition:

```
LONG Result=FNHx_cfgimptiming((LONG Taskid), (LONG Samplemode), (LONG Sampchtoaquire))
```

Input:

Taskid	The task to which to add the channels that this function creates
Samplemode	Specifies whether the task acquires or generates samples continuously or if it acquires or generates a finite number of samples
Sampchtoaquire	The number of samples to acquire or generate for each channel in the task if **Samplemode** is "DAQmx_Val_FiniteSamps". If **Samplemode** is "DAQmx_Val_ContSamps", NI-DAQmx uses this value to determine the buffer size

Return value: 0 = Function was successful
<>0 = Error occured

Example:

```
INTEGER Iresult1
LONG Result, Taskid, Samplemode, Sampchtoaquire
!
Samplemode=FNHx_get_constant("DAQmx_Val_FiniteSamps",Iresult1)      ! Sample mode
Sampchtoaquire=10
Result=FNHx_cfgimptiming((Taskid),(Samplemode),(Sampchtoaquire))
```

Read Functions

FNHx_readanalgf64 NI-DAQmx: **DAQmxReadAnalogF64**

Purpose: Reads multiple floating-point samples from a task that contains one or more analog input channels

HTBasic Definition:

```
LONG Result=FNHx_readanalgf64((LONG Taskid), (LONG Numpersam), (REAL T_out),
                              (LONG Fillmode), REAL Data_in(*), (LONG Asize),
                               LONG Sperchnread, LONG Reserved)
```

```
Input:
 Taskid            The task to read samples from
 Numpersam         The number of samples, per channel, to read
 T_out             The amount of time, in seconds, to wait for the function to read
                   the sample(s)
 Fillmode          Specifies whether or not the samples are interleaved
 Asize             The size of the array, in samples, into which samples are read
 Reserved          Reserved for future use. Pass 0 to this parameter

Return value:   0 = Function was successful
                      Sperchnread = Number of read REAL values
                      Data_in(*)  = Measurements
              <>0 = Error occured
```

Example:

```
 INTEGER Iresult1
 LONG Result, Taskid, Numpersam, Sperchnread, Reserved
 REAL T_out
 REAL Data_in(32767)
 DIM Tmp$[255]
 !
 Fillmode=FNHx_get_constant("DAQmx_Val_GroupByScanNumber",Iresult1)
 T_out=5.0
 Asize=SIZE(Data_in,1)          ! Size of array
 Numpersam=100                  ! read 100 samples
 Reserved=0
 Result=FNHx_readanalgf64((Taskid),(Numpersam),(T_out),(Fillmode),Data_in(*),(Asize),
                          Sperchnread,Reserved)
 !
 IF Result=0 THEN
    !... subsequent commands...
 END IF
```

FNHx_readanalsf64 NI-DAQmx: **DAQmxReadAnalogScalarF64**

Purpose: Reads a single floating-point sample from a task that contains a single analog input channel

HTBasic Definition:

```
LONG Result=FNHx_readanalsf64((LONG Taskid), (REAL T_out), REAL Data_in, LONG Reserved)
```

```
Input:
 Taskid            The task to read samples from
 T_out             The amount of time, in seconds, to wait for the function to read
                   the sample(s)
 Reserved          Reserved for future use. Pass 0 to this parameter

Return value:   0 = Function was successful
                    REAL variable Data_in contains the measurement
              <>0 = Error occured
```

Example:

```
 LONG Result, Taskid, Reserved
 REAL T_out, Avalue
 !
 T_out=5.0
 Reserved=0
 Result=FNHx_readanalgs64((Taskid),(T_out),Avalue, Reserved)
```

FNHx_rdbinaryi16 NI-DAQmx: **DAQmxReadBinaryI16**

Purpose: Reads multiple unscaled, signed 16-bit integer samples

HTBasic Definition:

```
LONG Result=FNHx_rdbinaryi16((LONG Taskid), (LONG Numpersam), (REAL T_out),
                             (LONG Fillmode), INTEGER Data_in(*), (LONG Asize),
                              LONG Sperchnread, LONG Reserved)
```

```
Input:
 Taskid            The task to read samples from
 Numpersam         The number of samples, per channel, to read
 T_out             The amount of time, in seconds, to wait for the function to read
                   the sample(s)
 Fillmode          Specifies whether or not the samples are interleaved
 Asize             The size of the array, in samples, into which samples are read
 Reserved          Reserved for future use. Pass 0 to this parameter

Return value:   0 = Function was successful
                    Sperchnread = Number of read INTEGER values from each channel
                    Data_in(*)  = 16 Bit unscaled measurements
              <>0 = Error occured
```

FNHx_rdbinaryi32 NI-DAQmx: **DAQmxReadBinaryI32**

Purpose: Reads multiple unscaled, signed 32-bit integer samples

HTBasic Definition:

```
LONG Result=FNHx_rdbinaryi32((LONG Taskid), (LONG Numpersam), (REAL T_out),
                             (LONG Fillmode), LONG Data_in(*), (LONG Asize),
                              LONG Sperchnread, LONG Reserved)
```

```
Input:
 Taskid            The task to read samples from
 Numpersam         The number of samples, per channel, to read
 T_out             The amount of time, in seconds, to wait for the function to read
                   the sample(s)
 Fillmode          Specifies whether or not the samples are interleaved
 Asize             The size of the array, in samples, into which samples are read
 Reserved          Reserved for future use. Pass 0 to this parameter

Return value:   0 = Function was successful
                    Sperchnread = Number of read INTEGER values from each channel
                    Data_in(*)  = 32 Bit unscaled measurements
              <>0 = Error occured
```

FNHx_readdigitu16 NI-DAQmx: **DAQmxReadDigitalU16**

Purpose: Reads multiple 16-bit integer samples

HTBasic Definition:

```
LONG Result=FNHx_readdigitu16((LONG Taskid), (LONG Sampsperchan), (REAL T_out),
                              (LONG Fillmode), INTEGER Dig_in(*), (LONG Asize),
                               LONG Samsreaddig, LONG Reserved)
```

```
Input:
 Taskid            The task to read samples from
 Numpersam         The number of samples, per channel, to read
 T_out             The amount of time, in seconds, to wait for the function to read
                   the sample(s)
 Fillmode          Specifies whether or not the samples are interleaved
 Asize             The size of the array, in samples, into which samples are read
 Reserved          Reserved for future use. Pass 0 to this parameter

Return value:   0 = Function was successful
                    Dig_in      = 16 Bit digital measurements
                    Samsreaddig = Number of read INTEGER values from each channel
              <>0 = Error occured
```

FNHx_readdigitu32 NI-DAQmx: **DAQmxReadDigitalU32**

Purpose: Reads multiple 32-bit integer samples

HTBasic Definition:

```
LONG Result=FNHx_readdigitu32((LONG Taskid), (LONG Sampsperchan), (REAL T_out),
                              (LONG Fillmode), INTEGER Dig_in(*), (LONG Asize),
                               LONG Samsreaddig, LONG Reserved)
```

```
Input:
 Taskid            The task to read samples from
 Numpersam         The number of samples, per channel, to read
 T_out             The amount of time, in seconds, to wait for the function to read
                   the sample(s)
 Fillmode          Specifies whether or not the samples are interleaved
 Asize             The size of the array, in samples, into which samples are read
 Reserved          Reserved for future use. Pass 0 to this parameter

Return value:   0 = Function was successful
                    Dig_in      = 32 Bit digital measurements
                    Samsreaddig = Number of read INTEGER values from each channel
              <>0 = Error occured
```

Example:

```
 INTEGER Iresult1, Dig_in(2000)
 LONG Result, Taskid, Sampsperchan, Asize, Samsreaddig, Reserved
 REAL T_out
 !
 Sampsperchan=4              ! number of samples per channel
 T_out=10.0                  ! 10 seconds timeout
 Asize=SIZE(Dig_in,1)        ! size of array
 MAT Dig_in=(0)
 Reserved=0
 !
 Fillmode=FNHx_get_constant("DAQmx_Val_GroupByChannel",Iresult1)  ! constant 1
 Result=DEF FNHx_readdigitu32((Taskid), (Sampsperchan), (T_out), (Fillmode), Dig_in(*),
                              (Asize), Samsreaddig, Reserved)
```

FNHx_readdigits32 NI-DAQmx: **DAQmxReadDigitalScalarU32**

Purpose: Reads a single 32-bit integer sample

HTBasic Definition:

```
LONG Result=FNHx_readdigits32((LONG Taskid), (REAL T_out), LONG Value, LONG Reserved)
```

```
Input:
 Taskid            The task to read samples from
 T_out             The amount of time, in seconds, to wait for the function to read
                   the sample(s)
 Reserved          Reserved for future use. Pass 0 to this parameter

Return value:   0 = Function was successful
                    Value = Measurement
              <>0 = Error occured
```

Example:

```
 LONG Result, Taskid, Value, Reserved
 REAL T_out
 !
 T_out=10.0                  ! 10s Timeout
 Reserved=0
 !
 Result=DEF FNHx_readdigitu32((Taskid), (T_out), Value, Reserved)
```

FNHx_readdiglines — NI-DAQmx: DAQmxReadDigitalLines

Purpose: Reads multiple samples from each digital line in a task

HTBasic Definition:

```
LONG Result=FNHx_readdiglines((LONG Taskid), (LONG Numpersam), (REAL T_out),
                              (LONG Fillmode), INTEGER Dig_in(*), (LONG Asize),
                               LONG Samsreaddig, LONG Numbytespersamp, LONG Reserved)
```

Input:

Taskid	The task to read samples from
Numpersam	The number of samples, per channel, to read
T_out	The amount of time, in seconds, to wait for the function to read the sample(s)
Fillmode	Specifies whether or not the samples are interleaved
Asize	The size of the array, in samples, into which samples are read
Reserved	Reserved for future use. Pass 0 to this parameter

Return value:

```
   0 = Function was successful
         Dig_in          = Measurements
         Samsreaddig     = Number of read samples from each channel
         Numbytespersamp = The number of elements in readArray that
                           constitutes a sample per channel. For each sample
                           per channel, numBytesPerSamp is the number
                           of bytes that channel consists of.
<>0 = Error occured
```

Example:

```
INTEGER Iresult1, Dig_in(2000)
LONG Result, Taskid, Sampsperchan, Asize, Samsreaddig, Numbytespersamp, Reserved
REAL T_out
!
T_out=10.0
Sampsperchan=4              ! number of samples (Bits) per channel
Asize=SIZE(Dig_in,1)        ! array size
MAT Dig_in=(0)
!
Fillmode=FNHx_get_constant("DAQmx_Val_GroupByChannel",Iresult1)  ! constant 1
Result=FNHx_readdiglines((Taskid),(Sampsperchan),(T_out),(Fillmode),Dig_in(*),(Asize),
                         Samsreaddig,Numbytespersamp,Reserved)
```

FNHx_readcntrf64 — NI-DAQmx: DAQmxReadCounterF64

Purpose: Reads multiple floating-point samples from a counter task

HTBasic Definition:

```
LONG Result=FNHx_readcntrf64((LONG Taskid), (LONG Numpersam), (REAL T_out),
                             REAL Data(*), (LONG Asize), LONG Samsread, LONG Reserved)
```

Input:

Taskid	The task to read samples from
Numpersam	The number of samples, per channel, to read
T_out	The amount of time, in seconds, to wait for the function to read the sample(s)
Asize	The size of the array, in samples, into which samples are read
Reserved	Reserved for future use. Pass 0 to this parameter

Return value:

```
   0 = Function was successful
         Data     = Array filled with samples
         Samsread = Number of read INTEGER values from each channel
<>0 = Error occured
```

FNHx_readcntru32 NI-DAQmx: **DAQmxReadCounterU32**

Purpose: Reads multiple 32-bit integer samples from a counter task

HTBasic Definition:

```
LONG Result=FNHx_readcntru32((LONG Taskid), (LONG Numpersam), (REAL T_out),
                             LONG Data(*), (LONG Asize), LONG Samsread, LONG Reserved)
```

```
Input:
 Taskid            The task to read samples from
 Numpersam         The number of samples, per channel, to read
 T_out             The amount of time, in seconds, to wait for the function to read
                   the sample(s)
 Asize             The size of the array, in samples, into which samples are read
 Reserved          Reserved for future use. Pass 0 to this parameter

Return value:   0 = Function was successful
                    Data      = Array filled with samples
                    Samsread  = Number of read INTEGER values from each channel
              <>0 = Error occured
```

FNHx_rdcntscalf64 NI-DAQmx: **DAQmxReadCounterScalarF64**

Purpose: Reads a single floating-point sample from a counter task

HTBasic Definition:

```
LONG Result=FNHx_rdcntscalf64((LONG Taskid), (REAL Timeout), REAL Value, LONG Reserved)
```

```
Input:
 Taskid            The task to read samples from
 T_out             The amount of time, in seconds, to wait for the function to read
                   the sample(s)
 Reserved          Reserved for future use. Pass 0 to this parameter

Return value:   0 = Function was successful
                    Value = Measurement
              <>0 = Error occured
```

Example:

```
 LONG Result, Taskid, Reserved
 REAL T_out, Value
 T_out=10.0                  ! 10s Timeout
 Reserved=0
 Result=FNHx_rdcntscalu32((Taskid),T_out,Value,Reserved)
```

FNHx_rdcntscalu32 NI-DAQmx: **DAQmxReadCounterScalarU32**

Purpose: Reads a 32-bit integer sample from a counter task

HTBasic Definition:

```
LONG Result=FNHx_rdcntscalu32((LONG Taskid), (REAL Timeout), LONG Value, LONG Reserved)
```

```
Input:
 Taskid            The task to read samples from
 T_out             The amount of time, in seconds, to wait for the function to read
                   the sample(s)
 Reserved          Reserved for future use. Pass 0 to this parameter

Return value:   0 = Function was successful
                    Value = Measurement
              <>0 = Error occured
```

Example:

```
 LONG Result, Taskid, Value, Reserved
 REAL T_out
 T_out=10.0                  ! 10s Timeout
 Reserved=0
 Result=FNHx_rdcntscalu32((Taskid),(T_out),Value,Reserved)
```

Write Functions

FNHx_writeanalf64 NI-DAQmx: **DAQmxWriteAnalogF64**

Purpose: Writes multiple floating-point samples to a task

HTBasic Definition:

```
LONG Result=FNHx_writeanalf64((LONG Taskid), (LONG Numpersam), (LONG Autostart),
                              (REAL T_out), (LONG Dlayout), REAL Data_out(*),
                               LONG Smpchwrit, LONG Reserved)

Input:
 Taskid            The task to write the samples to
 Numpersam         The number of samples, per channel, to read
 Autostart         Specifies whether or not this function automatically starts the task
                   if you do not start it
 T_out             The amount of time, in seconds, to wait for the function to read
                   the sample(s)
 Dlayout           Specifies how the samples are arranged (interleaved / noninterleaved)
 Data_out(*)       The array of 64-bit samples to write to the task
 Reserved          Reserved for future use. Pass 0 to this parameter

Return value:   0 = Function was successful
                    Data from Data_out(*) written to the buffer
                    Smpchwrit  The actual number of samples per channel successfully
                    written to the buffer
              <>0 = Error occured
```

Example:

```
 INTEGER Iresult1
 LONG Result, Taskid, Numpersam, Autostart, Sampsperchan, Dlayout, Reserved, Smpchwrit
 REAL T_out, Data_out(32767)
 DIM Tmp$[255]
 !
 Sampsperchan=1
 Autostart=1
 T_out=10.0
 Data_out(1)=PI
 Dlayout=FNHx_get_constant("DAQmx_Val_GroupByChannel",Iresult1)  ! get constant
 Result=FNHx_writeanalf64((Taskid),(Sampsperchan),(Autostart),(T_out),(Dlayout),
                          Data_out(1),Smpchwrit,Reserved)
```

FNHx_writeanasf64 NI-DAQmx: **DAQmxWriteAnalogScalarF64**

Purpose: Writes a floating-point sample to a task

HTBasic Definition:

```
LONG Result=FNHx_writeanasf64((LONG Taskid), (LONG Autostart), (REAL T_out),
                              (REAL Value), LONG Reserved)

Input:
 Taskid            The task to write the samples to
 Autostart         Specifies whether or not this function automatically starts the task
                   if you do not start it
 T_out             The amount of time, in seconds, to wait for the function to read
                   the sample(s)
 Value             The 64-bit REAL variable to write to the task
 Reserved          Reserved for future use. Pass 0 to this parameter

Return value:   0 = Function was successful
                    Data from Data_out(*) written to the buffer
              <>0 = Error occured
```

Example:

```
 LONG Result, Taskid, Autostart, Reserved
 REAL T_out, Value
 !
 Autostart=1
 T_out=10.0
 Value=1.654
 Reserved=0
 Result=FNHx_writeanasf64((Taskid),(Autostart),(T_out),(Value),Reserved)
```

FNHx_writebini16 NI-DAQmx: **DAQmxWriteBinaryI16**

Purpose: Writes unscaled signed integer samples to the task

HTBasic Definition:

```
LONG Result=FNHx_writebini16((LONG Taskid), (LONG Numpersam), (LONG Autostart),
                             (REAL T_out), (LONG Dlayout), INTEGER Data(*),
                              LONG Sperchnwrit, LONG Reserved)

Input:
 Taskid            The task to write the samples to
 Numpersam         The number of samples, per channel, to read
 Autostart         Specifies whether or not this function automatically starts the task
 T_out             The amount of time, in seconds, to wait for the function to write
                   the sample(s)
 Dlayout           Specifies whether or not the samples are interleaved
 Data(*)           Array of 16-Bit samples to write to the task
 Reserved          Reserved for future use. Pass 0 to this parameter

Return value:   0 = Function was successful
                      Sperchnwrit = Number of INTEGER values written to buffer
              <>0 = Error occured
```

FNHx_writebini32 NI-DAQmx: **DAQmxWriteBinaryI32**

Purpose: Writes unscaled 32-bit integer samples to the task

HTBasic Definition:

```
LONG Result=FNHx_writebini32((LONG Taskid), (LONG Numpersam), (LONG Autostart),
                             (REAL T_out), (LONG Dlayout), LONG Data(*),
                              LONG Sperchnwrit, LONG Reserved)

Input:
 Taskid            The task to write the samples to
 Numpersam         The number of samples, per channel, to write
 Autostart         Specifies whether or not this function automatically starts the task
 T_out             The amount of time, in seconds, to wait for the function to write
                   the sample(s)
 Dlayout           Specifies whether or not the samples are interleaved
 Data(*)           Array of 32-Bit samples to write to the task
 Reserved          Reserved for future use. Pass 0 to this parameter

Return value:   0 = Function was successful
                      Sperchnwrit = Number of LONG values written to buffer
              <>0 = Error occured
```

FNHx_writdigitu16 NI-DAQmx: **DAQmxWriteDigitalU16**

Purpose: Writes multiple 16-bit unsigned integer samples to a task that contains one or more digital output channels. Use this format for devices with up to 16 lines per port

HTBasic Definition:

```
LONG Result=FNHx_writdigitu16((LONG Taskid), (LONG Sampsperchan), (LONG Autostart),
                              (REAL T_out), (LONG Datalayout), INTEGER Dig_out(*),
                               LONG Sampswritten, LONG Reserved)

Input:
 Taskid            The task to write the samples to
 Sampsperchan      The number of samples, per channel, to write
 Autostart         Specifies whether or not this function automatically starts the task
 T_out             The amount of time, in seconds, to wait for the function to write
                   the sample(s)
 Datalayout        Specifies whether or not the samples are interleaved
 Dig_out(*)        INTEGER array with data to be written to the buffer
 Reserved          Reserved for future use. Pass 0 to this parameter

Return value:   0 = Function was successful
                      Sampswritten = Number of values written to buffer for each channel
              <>0 = Error occured
```

FNHx_writdigitu32 NI-DAQmx: **DAQmxWriteDigitalU32**

Purpose: Writes multiple 32-bit unsigned integer samples to a task that contains one or more digital output channels. Use this format for devices with up to 32 lines per port

HTBasic Definition:

```
LONG Result=FNHx_writdigitu32((LONG Taskid), (LONG Sampsperchan), (LONG Autostart),
                              (REAL T_out), (LONG Datalayout), (LONG Asize),
                               LONG Dig_out(*), LONG Sampswritten, LONG Reserved)

Input:
 Taskid            The task to write the samples to
 Sampsperchan      The number of samples, per channel, to write
 Autostart         Specifies whether or not this function automatically starts the task
 T_out             The amount of time, in seconds, to wait for the function to write
                   the sample(s)
 Datalayout        Specifies whether or not the samples are interleaved
 Dig_out(*)        LONG array with data to be written to the buffer
 Reserved          Reserved for future use. Pass 0 to this parameter

Return value:   0 = Function was successful
                    Sampswritten = Number of values written to buffer for each channel
              <>0 = Error occured
```

FNHx_writdigscu32 NI-DAQmx: **DAQmxWriteDigitalScalarU32**

Purpose: Writes a single 32-bit unsigned integer sample to a task

HTBasic Definition:

```
LONG Result=FNHx_writdigscu32((LONG Taskid), (LONG Autostart), (REAL T_out), LONG Value,
                               LONG Reserved)

Input:
 Taskid            The task to write the samples to
 Autostart         Specifies whether or not this function automatically starts the task
 T_out             The amount of time, in seconds, to wait for the function to write
 Value             32-Bit sample to write to the task
 Reserved          Reserved for future use. Pass 0 to this parameter

Return value:   0 = Function was successful
              <>0 = Error occured
```

Example:

```
 LONG Result, Taskid, Value, Reserved
 REAL T_out
 !
 Reserved=0
 Autostart=1
 T_out=5
 Value=255
 Result=FNHx_writdigscu32((Digitalout_task),(Autostart),(T_out),Value,Reserved)
```

FNHx_writdiglines NI-DAQmx: **DAQmxWriteDigitalLines**

Purpose: Writes multiple samples to each digital line in a task

HTBasic Definition:

```
LONG Result=FNHx_writdiglines((LONG Taskid), (LONG Sampsperchan), (LONG Autostart),
                              (REAL T_out), (LONG Datalayout), (LONG Asize),
                               INTEGER Dig_out(*), LONG Sampswritten, LONG Reserved)

Input:
 Taskid            The task to write the samples to
 Sampsperchan      The number of samples, per channel, to write
 Autostart         Specifies whether or not this function automatically starts the task
 T_out             The amount of time, in seconds, to wait for the function to write
                   the sample(s)
 Datalayout        Specifies whether or not the samples are interleaved
 Asize             Size of data array
 Dig_out(*)        INTEGER array with data to be written to the buffer
 Reserved          Reserved for future use. Pass 0 to this parameter

Return value:   0 = Function was successful
                    Sampswritten = Number of values written to buffer for each channel
              <>0 = Error occured
```

Example:

```
INTEGER Iresult1
LONG Result, Taskid, Sampsperchan, Autostart, Asize, Sampswritten, Datalayout
LONG Dig_out(32767), Reserved
REAL T_out
!
Sampsperchan=2              ! number of sample (Bits) per channel
Autostart=1
T_out=10.0
Asize=SIZE(Dig_out,1)       ! array size
Dig_out(1)=255              ! 1. output value
Dig_out(2)=0                ! 2. output value
Reserved=0
!
Datalayout=FNHx_get_constant("DAQmx_Val_GroupByChannel",Iresult1)  ! constant 1
Result=FNHx_writdiglines((Taskid),(Sampsperchan),(Autostart),(T_out),
                         (Datalayout),(Asize),Dig_out(*),Sampswritten,Reserved)
```

FNHx_wrctrfreq — NI-DAQmx: DAQmxWriteCtrFreq

Purpose: Writes a new pulse frequency and duty cycle to each channel

HTBasic Definition:

```
LONG Result=FNHx_wrctrfreq((LONG Taskid), (LONG Sampsperchan), (LONG Autostart),
                           (REAL T_out), (LONG Dlayout), (REAL Freq), (REAL Dutycycle),
                            LONG Sampswritten, LONG Reserved)
```

Input:

Taskid	The task to write the values to
Sampsperchan	The number of samples, per channel, to write
Autostart	Specifies whether or not this function automatically starts the task
T_out	The amount of time, in seconds, to wait for the function to write the sample(s)
Dlayout	Specifies whether or not the samples are interleaved
Freq	The frequency at which to generate pulses
Dutycycle	The width of the pulse divided by the pulse period. NI-DAQmx uses this ratio, combined with frequency, to determine pulse width and the interval between pulses
Reserved	Reserved for future use. Pass 0 to this parameter

Return value: 0 = Function was successful
Sampswritten = Number of values written to buffer for each channel
<>0 = Error occured

FNHx_wrctrfrqscal NI-DAQmx: **DAQmxWriteCtrFreqScalar**

Purpose: Writes a new pulse frequency and duty cycle to each channel

HTBasic Definition:

```
LONG Result=FNHx_wrctrfrqscal((LONG Taskid), (LONG Autostart), (REAL T_out),
                              (REAL Freq), (REAL Dutycycle), LONG Reserved)

Input:
 Taskid            The task to write the values to
 Autostart         Specifies whether or not this function automatically starts the task
 T_out             The amount of time, in seconds, to wait for the function to write
                   the sample(s)
 Freq              The frequency at which to generate pulses
 Dutycycle         The width of the pulse divided by the pulse period.
                   NI-DAQmx uses this ratio, combined with frequency, to determine pulse
                   width and the interval between pulses
 Reserved          Reserved for future use. Pass 0 to this parameter

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_wrctrtime NI-DAQmx: **DAQmxWriteCtrTime**

Purpose: Writes a new pulse high time and low time to each channel

HTBasic Definition:

```
LONG Result=FNHx_wrctrtime((LONG Taskid), (LONG Sampsperchan), (LONG Autostart),
                           (REAL T_out), (LONG Dlayout), (REAL Hightime),
                           (REAL Lowtime), LONG Sampswritten, LONG Reserved)

Input:
 Taskid            The task to write the values to
 Sampsperchan      The number of samples, per channel, to write
 Autostart         Specifies whether or not this function automatically starts the task
 T_out             The amount of time, in seconds, to wait for the function to write
                   the sample(s)
 Dlayout           Specifies whether or not the samples are interleaved
 Hightime          The amount of time the pulse is high, in seconds
 Lowtime           The amount of time the pulse is low, in seconds
 Reserved          Reserved for future use. Pass 0 to this parameter

Return value:   0 = Function was successful
                    Sampswritten = Number of values written to buffer for each channel
              <>0 = Error occured
```

FNHx_wrctrtimscal NI-DAQmx: **DAQmxWriteCtrTimeScalar**

Purpose: Writes a new pulse high time and low time to each channel

HTBasic Definition:

```
LONG Result=FNHx_wrctrtimscal((LONG Taskid), (LONG Autostart), (REAL T_out),
                              (REAL Hightime), (REAL Lowtime), LONG Reserved)

Input:
 Taskid            The task to write the values to
 Autostart         Specifies whether or not this function automatically starts the task
 T_out             The amount of time, in seconds, to wait for the function to write
                   the sample(s)
 Hightime          The amount of time the pulse is high, in seconds
 Lowtime           The amount of time the pulse is low, in seconds
 Reserved          Reserved for future use. Pass 0 to this parameter

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_wrctrticks NI-DAQmx: **DAQmxWriteCtrTicks**

Purpose: Writes new pulse high tick counts and low tick counts to each channel

HTBasic Definition:

```
LONG Result=FNHx_wrctrticks((LONG Taskid), (LONG Sampsperchan), (LONG Autostart),
                            (REAL T_out), (LONG Dlayout), (REAL Highticks),
                            (REAL Lowticks), LONG Sampswritten, LONG Reserved)

Input:
 Taskid            The task to write the values to
 Sampsperchan      The number of samples, per channel, to write
 Autostart         Specifies whether or not this function automatically starts the task
 T_out             The amount of time, in seconds, to wait for the function to write
                   the sample(s)
 Dlayout           Specifies whether or not the samples are interleaved
 Highticks         The number of timebase ticks the pulse is high
 Lowticks          The number of timebase ticks the pulse is low
 Reserved          Reserved for future use. Pass 0 to this parameter

Return value:   0 = Function was successful
                    Sampswritten = Number of values written to buffer for each channel
              <>0 = Error occured
```

FNHx_wrctrtickssc NI-DAQmx: **DAQmxWriteCtrTicksScalar**

Purpose: Writes a new pulse high time and low time to each channel

HTBasic Definition:

```
LONG Result=FNHx_wrctrtickssc((LONG Taskid), (LONG Autostart), (REAL T_out),
                              (REAL Highticks), (REAL Lowticks), LONG Reserved)

Input:
 Taskid            The task to write the values to
 Autostart         Specifies whether or not this function automatically starts the task
 T_out             The amount of time, in seconds, to wait for the function to write
                   the sample(s)
 Highticks         The number of timebase ticks the pulse is high
 Lowticks          The number of timebase ticks the pulse is low
 Reserved          Reserved for future use. Pass 0 to this parameter

Return value:   0 = Function was successful
              <>0 = Error occured
```

Triggering Functions

FNHx_disstarttrig — NI-DAQmx: **DAQmxDisableStartTrig**

Purpose: Configures the task to start acquiring or generating samples immediately upon starting the task

HTBasic Definition:

```
LONG Result=FNHx_disstarttrig((LONG Taskid))

Input:
 Taskid            The task used in this function

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_cfdgedsttrig — NI-DAQmx: **DAQmxCfgDigEdgeStartTrig**

Purpose: Configures the task to start acquiring or generating samples on a rising or falling edge of a digital signal

HTBasic Definition:

```
LONG Result=FNHx_cfdgedsttrig((LONG Taskid), (Triggersrc$), (LONG Triggeredge))

Input:
 Taskid            The task used in this function
 Triggersrc$       The name of a terminal where there is a digital signal to use as the
                   source of the trigger
 Triggeredge       Specifies on which edge of a digital signal to start acquiring or
                   generating samples

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_cfganedstrig — NI-DAQmx: **DAQmxCfgAnlgEdgeStartTrig**

Purpose: Configures the task to start acquiring or generating samples when an analog signal crosses the level you specify

HTBasic Definition:

```
LONG Result=FNHx_cfganedstrig((LONG Taskid), (Triggersrc$), (LONG Triggerslop),
                              (REAL Triggerlev))

Input:
 Taskid            The task used in this function
 Triggersrc$       The name of a channel or terminal where there is an analog signal to
                   use as the source of the trigger
 Triggerslop       Specifies on which slope of the signal to start acquiring or
                   generating samples when the signal crosses Triggerlev
 Triggerlev        The threshold at which to start acquiring or generating samples

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_cfganwistrig — NI-DAQmx: **DAQmxCfgAnlgWindowStartTrig**

Purpose: Configures the task to start acquiring or generating samples when an analog signal enters or leaves a range you specify

HTBasic Definition:

```
LONG Result=FNHx_cfganwistrig((LONG Taskid), (Triggersrc$), (LONG Triggerwhen),
                              (REAL Windowtop), (REAL Windowbottom))

Input:
 Taskid            The task used in this function
 Triggersrc$       The name of a channel or terminal where there is an analog signal to
                   use as the source of the trigger
 Triggerwhen       Specifies whether the task starts measuring or generating samples
                   when the signal enters the window or when it leaves the window
 Windowtop         The upper limit of the window
 Windowbottom      The lower limit of the window

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_cfgdiptstrig NI-DAQmx: **DAQmxCfgDigPatternStartTrig**

Purpose: Configures a task to start acquiring or generating samples when a digital pattern is matched

HTBasic Definition:

```
LONG Result=FNHx_cfgdiptstrig((LONG Taskid), (Triggersrc$), (Triggerpat$),
                              (LONG Triggerwhen))

Input:
 Taskid            The task used in this function
 Triggersrc$       Specifies the physical channels to use for pattern matching
 Triggerpat$       Specifies the digital pattern that must be met for the trigger
                   to occur
 Triggerwhen       Specifies the conditions under which the trigger occurs

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_disreftrig NI-DAQmx: **DAQmxDisableRefTrig**

Purpose: Disables reference triggering for the measurement or generation

HTBasic Definition:

```
LONG Result=FNHx_disreftrig((LONG Taskid))

Input:
 Taskid            The task used in this function

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_cfgdiedrtrig NI-DAQmx: **DAQmxCfgDigEdgeRefTrig**

Purpose: Configures the task to stop the acquisition when the device acquires all pretrigger samples, detects a rising or falling edge of a digital signal, and acquires all posttrigger samples

HTBasic Definition:

```
LONG Result=FNHx_cfgdiedrtrig((LONG Taskid), (Triggersrc$), (LONG Triggeredge),
                              (LONG Pretrigsam))

Input:
 Taskid            The task used in this function
 Triggersrc$       Specifies the name of a terminal where there is a digital signal to
                   use as the source of the trigger
 Triggeredge       Specifies on which edge of the digital signal the Reference
                   Trigger occurs
 Pretrigsam        The minimum number of samples per channel to acquire before
                   recognizing the Reference Trigger.

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_cfganedrtrig NI-DAQmx: **DAQmxCfgAnlgEdgeRefTrig**

Purpose: Configures the task to start acquiring or generating samples when an analog signal crosses the level you specify

HTBasic Definition:

```
LONG Result=FNHx_cfganedrtrig((LONG Taskid), (Triggersrc$), (LONG Triggerslop),
                              (REAL Triggerlev), (LONG Pretrigsam))

Input:
 Taskid            The task used in this function
 Triggersrc$       The name of a channel or terminal where there is an analog signal to
                   use as the source of the trigger
 Triggerslop       Specifies on which slope of the signal to start acquiring or
                   generating samples when the signal crosses Triggerlev
 Triggerlev        The threshold at which to start acquiring or generating samples
 Pretrigsam        The minimum number of samples per channel to acquire before
                   recognizing the Reference Trigger

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_cfganwirtrig — NI-DAQmx: DAQmxCfgAnlgWindowRefTrig

Purpose: Configures the task to stop the acquisition when the device acquires all pretrigger samples, an analog signal enters or leaves a range you specify, and the device acquires all post-trigger samples

HTBasic Definition:

```
LONG Result=FNHx_cfganwirtrig((LONG Taskid), (Triggersrc$), (LONG Triggerwhen),
                              (REAL Windowtop), (REAL Windowbottom), (LONG Pretrigsam))

Input:
 Taskid            The task used in this function
 Triggersrc$       The name of a virtual channel or terminal where there is an analog
                   signal to use as the source of the trigger
 Triggerwhen       Specifies whether the task starts measuring or generating samples
                   when the signal enters the window or when it leaves the window
 Windowtop         The upper limit of the window
 Windowbottom      The lower limit of the window
 Pretrigsam        The minimum number of samples per channel to acquire before
                   recognizing the Reference Trigger

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_cfgdipartrig — NI-DAQmx: DAQmxCfgDigPatternRefTrig

Purpose: Configures the task to stop the acquisition when the device acquires all pretrigger samples, matches or does not match a digital pattern, and acquires all posttrigger samples

HTBasic Definition:

```
LONG Result=FNHx_cfgdipartrig((LONG Taskid), (Triggersrc$), (Triggerpat$),
                              (LONG Triggerwhen), (LONG Pretrigsam))

Input:
 Taskid            The task used in this function
 Triggersrc$       Specifies the physical channels to use for pattern matching
 Triggerpat$       Specifies the digital pattern that must be met for the trigger
                   to occur
 Triggerwhen       Specifies the conditions under which the trigger occurs
 Pretrigsam        The minimum number of samples per channel to acquire before
                   recognizing the Reference Trigger

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_disadvtrig — NI-DAQmx: DAQmxDisableAdvTrig

Purpose: Disables the advance triggering for the task

HTBasic Definition:

```
LONG Result=FNHx_disadvtrig((LONG Taskid))

Input:
 Taskid            The task used in this function

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_cfgdiedatrig — NI-DAQmx: DAQmxCfgDigEdgeAdvTrig

Purpose: Configures a switch task to advance to the next entry in a scan list on a rising or falling edge of a digital signal

HTBasic Definition:

```
LONG Result=FNHx_cfgdiedatrig((LONG Taskid), (Triggersrc$), (LONG Triggeredge))

Input:
 Taskid            The task used in this function
 Triggersrc$       Specifies the name of a terminal where there is a digital signal to
                   use as the source of the trigger
 Triggeredge       Specifies on which edge of the digital signal the Reference

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_sendsofttrig — NI-DAQmx: DAQmxSendSoftwareTrigger

Purpose: Generates the specified software trigger

HTBasic Definition:

```
LONG Result=FNHx_sendsofttrig((LONG Taskid), (LONG Triggerid))

Input:
 Taskid            The task used in this function
 Triggerid         Specifies which software trigger to generate

Return value:   0 = Function was successful
              <>0 = Error occured
```

Calibration

FNHx_selfcal NI-DAQmx: **DAQmxSelfCal**

Purpose: Measures the onboard reference voltage of the device and adjusts the self-calibration constants to account for any errors caused by short-term fluctuations in the operating environment

HTBasic Definition:

```
LONG Result=FNHx_selfcal(Devicename$)

Input:
 Devicename$       The name of the device, as configured in MAX, to which this
                   operation applies

Return value:   0 = Function was successful
              <>0 = Error occured
```

Example:

```
 LONG Result
 DIM Devicename$[255]
 !
 Devicename$="Dev1"
 Result=FNHx_selfcal(Devicename$)
 !
 IF Result=0 THEN
    PRINT "Device '"&Devicename$&"' has been self-calibrated."
 END IF
```

FNHx_perfbrofnlca NI-DAQmx: **DAQmxPerformBridgeOffsetNullingCal**

Purpose: Measures the onboard reference voltage of the device and adjusts the self-calibration constants to account for any errors caused by short-term fluctuations in the operating environment

HTBasic Definition:

```
LONG Result=FNHx_perfbrofnlca((LONG Taskid), Channel$)

Input:
 Taskid            The task used in this function
 Channel$          A subset of virtual channels in the task that you want to calibrate.
                   Use this parameter if you do not want to calibrate all the channels
                   in the task or if some channels in the task measure non-bridge-based
                   sensors

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_getcadevtemp NI-DAQmx: **DAQmxGetCalDevTemp**

Purpose: Returns in degrees Celsius the current temperature of the device

HTBasic Definition:

```
LONG Result=FNHx_getcadevtemp((Devicename$), REAL Data)

Input:
 Devicename$       The name of the device for this function

Return value:   0 = Function was successful
                    Data: Current temperature of the device in degrees
              <>0 = Error occured
```

FNHx_getsfcaltemp NI-DAQmx: **DAQmxGetSelfCalLastTemp**

Purpose: Returns in degrees Celsius the temperature of the device at the time of the last self calibration. Compare this temperature to the current onboard temperature to determine if you should perform another calibration.

HTBasic Definition:

```
LONG Result=FNHx_getsfcaltemp((Devicename$), REAL Data)

Input:
 Devicename$       The name of the device for this function

Return value:   0 = Function was successful
                    Data: Temperature of the device at the time of last calibration
              <>0 = Error occured
```

FNHx_getexcaltemp NI-DAQmx: **DAQmxGetExtCalLastTemp**

Purpose: Returns in degrees Celsius the temperature of the device at the time of the last external calibration.

HTBasic Definition:

```
LONG Result=FNHx_getexcaltemp((Devicename$), REAL Data)

Input:
 Devicename$       The name of the device for this function

Return value:   0 = Function was successful
                    Data: Temperature of the device at the time of last external
                          calibration
              <>0 = Error occured
```

FNHx_initextcal NI-DAQmx: **DAQmxInitExtCal**

Purpose: Starts an external calibration session on a device

HTBasic Definition:

```
LONG Result=FNHx_initextcal((Devicename$), (Password$), LONG Calhandle)

Input:
 Devicename$       The name of the device, as configured in MAX, to which this
                   operation applies
 Password$         The current calibration password for the device (case sensitive)

Return value:   0 = Function was successful
                    Calhandle contains a reference to the calibration session
              <>0 = Error occured
```

FNHx_closeextcal NI-DAQmx: **DAQmxCloseExtCal**

Purpose: Closes an open external calibration session

HTBasic Definition:

```
LONG Result=FNHx_closeextcal((LONG Calhandle), (LONG Action))

Input:
 Calhandle         A reference to the calibration session that you created using the
                   DAQmxInitExtCal function
 Action            Specifies how to close the calibration session

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_getscaldtati NI-DAQmx: **DAQmxGetSelfCalLastDateAndTime**

Purpose: Indicates the last date and time that the device underwent a self calibration

HTBasic Definition:

```
LONG Result=FNHx_getscaldtati((Devname$), LONG Year, LONG Month, LONG Day, LONG Hour,
                              LONG Minute)

Input:
 Devname$          The name of the device for this function
 Year              The last year that the device underwent a self-calibration
 Month             The last month that the device underwent a self-calibration
 Day               The last day that the device underwent a self-calibration
 Hour              The last hour, on a 24-hour clock, that the device underwent
                   a self-calibration
 Minute            The last minute that the device underwent a self-calibration

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_getecaldtati NI-DAQmx: **DAQmxGetExtCalLastDateAndTime**

Purpose: Indicates the last date and time that the device underwent an external calibration

HTBasic Definition:

```
LONG Result=FNHx_getecaldtati((Devname$), LONG Year, LONG Month, LONG Day, LONG Hour,
                              LONG Minute)

Input:
 Devname$          The name of the device for this function
 Year              The last year that the device underwent an external calibration
 Month             The last month that the device underwent an external calibration
 Day               The last day that the device underwent an external calibration
 Hour              The last hour, on a 24-hour clock, that the device underwent
                   an external calibration
 Minute            The last minute that the device underwent an external calibration

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_reslexcalcon NI-DAQmx: **DAQmxRestoreLastExtCalConst**

Purpose: Sets the self calibration constants of the device to the the current external calibration constants

HTBasic Definition:

```
LONG Result=FNHx_reslexcalcon((Devname$))

Input:
 Devname$          The name of the device for this function

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_chnextcalpwd NI-DAQmx: **DAQmxChangeExtCalPassword**

Purpose: Changes the external calibration password of the device

HTBasic Definition:

```
LONG Result=FNHx_chnextcalpwd((Devname$),(Oldpassword$),(Newpassword$))

Input:
 Devname$          The name of the device for this function
 Oldpassword$      The current calibration password for the device
 Newpassword$      The new calibration password for the device (four chars max.)

Return value:   0 = Function was successful
              <>0 = Error occurred
```

FNHx_getecalrecin NI-DAQmx: **DAQmxGetExtCalRecommendedInterval**

Purpose: Indicates in months the National Instruments recommended interval between each external calibration

HTBasic Definition:

```
LONG Result=FNHx_getecalrecin((Devicename$), LONG Data)
```

Input:

Devicename$	The name of the device, as configured in MAX, to which this operation applies

Return value:

```
   0 = Function was successful
         Data contains the recommended interval between each external
              calibration
 <>0 = Error occured
```

FNHx_adjdsaaical NI-DAQmx: **DAQmxAdjustDSAAICal**

Purpose: Adjusts the external calibration constants for the analog input section of a DSA device

HTBasic Definition:

```
LONG Result=FNHx_adjdsaaical((LONG Calhandle), (REAL Refvoltage))
```

Input:

Calhandle	A reference to the calibration handle that you created using the DAQmxInitExtCal function
Refvoltage	The known voltage, in volts, to use as a reference for calibration

Return value:

```
   0 = Function was successful
 <>0 = Error occured
```

FNHx_adjdsaaocal NI-DAQmx: **DAQmxAdjustDSAAOCal**

Purpose: Adjusts the external calibration constants for the analog output section of a DSA device

HTBasic Definition:

```
LONG Result=FNHx_adjdsaaocal((LONG Calhandle), (LONG Channel), (REAL Reqlowvolt),
                             (REAL Actlowvolt), (REAL Reqhigvolt), (REAL Acthigvolt),
                             (REAL Gainset))
```

Input:

Calhandle	A reference to the calibration handle that you created using the DAQmxInitExtCal function
Channel	The number of the channel to calibrate. This number is the numeric portion of the physical channel name, not the full physical channel name
Reqlowvolt	The low voltage you attempted to generate at the gain setting you specified
Actlowvolt	The actual low voltage as measured by an external sensor
Reqhigvolt	The high voltage you attempted to generate at the gain setting you specified
Acthigvolt	The actual high voltage as measured by an external sensor
Gainset	The gain setting you used when you attempted to generate the requested high voltage and requested low voltage

Return value:

```
   0 = Function was successful
 <>0 = Error occured
```

FNHx_adjdsatbcal NI-DAQmx: **DAQmxAdjustDSATimebaseCal**

Purpose: Adjusts the external calibration constant for the timebase of a DSA device with an adjustable oscillator

HTBasic Definition:

```
LONG Result=FNHx_adjdsatbcal((LONG Calhandle), (REAL Reffreq))

Input:
 Calhandle          A reference to the calibration handle that you created using
                    the DAQmxInitExtCal function
 Reffreq            The frequency, in hertz, of the signal to use as a reference
                    for calibration

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_adj4204cal NI-DAQmx: **DAQmxAdjust4204Cal**

Purpose: Adjusts the internal and external calibration constants for the SCMP pod on the PXI-4204 device

HTBasic Definition:

```
LONG Result=FNHx_adj4204cal((LONG Calhandle), (Channames$), (REAL Lpfreq),
                            (LONG Trackholden, (REAL Inputval))

Input:
 Calhandle          A reference to the calibration handle that you created using
                    the DAQmxInitExtCal function
 Channames$         The physical channel(s) to calibrate
 Lpfreq             The low pass cutoff frequency, in hertz, (6 or 10000) on the SCMP pod
                    to calibrate
 Trackholden        Specifies whether calibrating for trackHold is enabled or disabled
 Inputval           The known voltage, in volts, to use as a reference for calibration

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_adj4220cal NI-DAQmx: **DAQmxAdjust4220Cal**

Purpose: Adjusts the internal and external calibration constants for the SCMP pod on the PXI-4220 device

HTBasic Definition:

```
LONG Result=FNHx_adj4220cal((LONG Calhandle), (Channames$), (REAL Gain),
                            (REAL Inputval))

Input:
 Calhandle          A reference to the calibration handle that you created using
                    the DAQmxInitExtCal function
 Channames$         The physical channel(s) to calibrate
 Gain               The gain value on the SCMP pod to calibrate
Inputval           The known voltage, in volts, to use as a reference for calibration

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_adj4224cal NI-DAQmx: **DAQmxAdjust4224Cal**

Purpose: Adjusts the internal and external calibration constants for the SCMP pod on the PXI-4224 device

HTBasic Definition:

```
LONG Result=FNHx_adj4224cal((LONG Calhandle), (Channames$), (REAL Gain),
                            (REAL Inputval))

Input:
 Calhandle          A reference to the calibration handle that you created using
                    the DAQmxInitExtCal function
 Channames$         The physical channel(s) to calibrate
 Gain               The gain value on the SCMP pod to calibrate
 Inputval           The known voltage, in volts, to use as a reference for calibration

Return value:   0 = Function was successful
               <>0 = Error occured
```

FNHx_adj4225cal NI-DAQmx: **DAQmxAdjust4225Cal**

Purpose: Adjusts the internal and external calibration constants for the SCMP pod on the PXI-4225 device

HTBasic Definition:

```
LONG Result=FNHx_adj4225cal((LONG Calhandle), (Channames$), (REAL Gain),
                            (REAL Inputval))

Input:
 Calhandle          A reference to the calibration handle that you created using
                    the DAQmxInitExtCal function
 Channames$         The physical channel(s) to calibrate
 Gain               The gain value on the SCMP pod to calibrate
 Inputval           The known voltage, in volts, to use as a reference for calibration

Return value:   0 = Function was successful
               <>0 = Error occured
```

FNHx_esercaladjst NI-DAQmx: **DAQmxESeriesCalAdjust**

Purpose: Adjusts the external calibration constants on an E-Series device

HTBasic Definition:

```
LONG Result=FNHx_esercaladjst((LONG Calhandle), (REAL Refvoltage))

Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Refvoltage         The known voltage, in volts, to use as a reference for calibration.
                    This voltage should be between +6.000 V and +9.999 V

Return value:   0 = Function was successful
               <>0 = Error occured
```

FNHx_msercaladjst NI-DAQmx: **DAQmxMSeriesCalAdjust**

Purpose: Adjusts the external calibration constants for an M Series device

HTBasic Definition:

```
LONG Result=FNHx_msercaladjst((LONG Calhandle), (REAL Refvoltage))

Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Refvoltage         The known voltage, in volts, to use as a reference for calibration.
                    This voltage should be between +6.000 V and +9.999 V

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_ssercaladjst NI-DAQmx: **DAQmxSSeriesCalAdjust**

Purpose: Adjusts the external calibration constants for an S Series device

HTBasic Definition:

```
LONG Result=FNHx_ssercaladjst((LONG Calhandle), (REAL Refvoltage))

Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Refvoltage         The known voltage, in volts, to use as a reference for calibration.
                    This voltage should be between +6.000 V and +9.999 V

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_scbbcaladjst NI-DAQmx: **DAQmxSCBaseboardCalAdjust**

Purpose: Adjusts the external calibration constants on for the baseboard of an SC Series device

HTBasic Definition:
```
LONG Result=FNHx_scbbcaladjst((LONG Calhandle), (REAL Refvoltage))
```

```
Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Refvoltage         The known voltage, in volts, to use as a reference for calibration.
                    This voltage should be between +6.000 V and +9.999 V

Return value:    0 = Function was successful
                <>0 = Error occured
```

FNHx_aosecaladjst NI-DAQmx: **DAQmxAOSeriesCalAdjust**

Purpose: Adjusts the external calibration constants on an AO Series device

HTBasic Definition:
```
LONG Result=FNHx_aosecaladjst((LONG Calhandle), (REAL Refvoltage))
```

```
Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Refvoltage         The known voltage, in volts, to use as a reference for calibration.
                    This voltage should be between +6.000 V and +9.999 V

Return value:    0 = Function was successful
                <>0 = Error occured
```

FNHx_devsuppcal NI-DAQmx: **DAQmxDeviceSupportsCal**

Purpose: Checks wether or not calibration is supported by device

HTBasic Definition:
```
LONG Result=FNHx_devsuppcal(Devicename$, (LONG Calsupported))
```

```
Input:
 Devicename$        The name of the device, as configured in MAX, to which this
                    operation applies

Return value:    0 = Function was successful
                     Calsupported  returns 1 if calibration is supported, otherwise 0
                <>0 = Error occured
```

FNHx_selfcalsupp NI-DAQmx: **DAQmxGetSelfCalSupported**

Purpose: Checks wether or not self calibration is supported by device

HTBasic Definition:
```
LONG Result=FNHx_selfcalsupp(Devicename$, (LONG Calsupported))
```

```
Input:
 Devicename$        The name of the device, as configured in MAX, to which this
                    operation applies

Return value:    0 = Function was successful
                     Calsupported  returns 1 if calibration is supported, otherwise 0
                <>0 = Error occured
```

FNHx_set1104cal NI-DAQmx 8.5: **DAQmxSetup1104Cal**

Purpose: Specifies the channel on the SCXI-1104 module for calibration

HTBasic Definition:

```
LONG Result=FNHx_set1104cal((LONG Calhandle), (Channel$))
```

```
Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Channel$           The physical channel to calibrate

Return value:    0 = Function was successful
                <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_adj1104cal NI-DAQmx 8.5: **DAQmxAdjust1104Cal**

Purpose: Adjusts the calibration constants for the SCXI-1104 module

HTBasic Definition:

```
LONG Result=FNHx_adj1104cal((LONG Calhandle), (REAL Refvoltage), (REAL Measoutput))
```

```
Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Refvoltage         The known voltage, in volts, to use as a reference for calibration
 Measoutput         The voltage measured at the output of the module

Return value:    0 = Function was successful
                <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_set1112cal NI-DAQmx 8.5: **DAQmxSetup1112Cal**

Purpose: Specifies the channel on the SCXI-1112 module for calibration

HTBasic Definition:

```
LONG Result=FNHx_set1112cal((LONG Calhandle), (Channel$))
```

```
Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Channel$           The physical channel to calibrate

Return value:    0 = Function was successful
                <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_adj1112cal NI-DAQmx 8.5: **DAQmxAdjust1112Cal**

Purpose: Adjusts the calibration constants for the SCXI-1112 module

HTBasic Definition:

```
LONG Result=FNHx_adj1112cal((LONG Calhandle), (REAL Refvoltage), (REAL Measoutput))
```

```
Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Refvoltage         The known voltage, in volts, to use as a reference for calibration
 Measoutput         The voltage measured at the output of the module

Return value:    0 = Function was successful
                <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_set1124cal NI-DAQmx 8.5: **DAQmxSetup1124Cal**

Purpose: Writes the specified binary value to the D/A circuitry on the specified channel at the specified range.

HTBasic Definition:

```
LONG Result=FNHx_set1124cal((LONG Calhandle), (Channel$), (REAL Range), (REAL Daqvalue))
```

```
Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Channel$           The physical channel to calibrate
 Range              The range to calibrate
 Daqvalue           The binary number to write to the DAC circuitry

Return value:    0 = Function was successful
                <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_adj1124cal NI-DAQmx 8.5: **DAQmxAdjust1124Cal**

Purpose: Adjusts the calibration constants for the SCXI-1124 module.

HTBasic Definition:

```
LONG Result=FNHx_adj1124cal((LONG Calhandle), (REAL Measoutput))
```

```
Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Measoutput         The voltage measured at the output of the module

Return value:    0 = Function was successful
                <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_set1125cal NI-DAQmx 8.5: **DAQmxSetup1125Cal**

Purpose: Sets the SCXI-1125 module to the specified gain value.

HTBasic Definition:

```
LONG Result=FNHx_set1125cal((LONG Calhandle), (Channel$), (REAL Gain))
```

```
Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Channel$           The physical channel to calibrate
 Gain               The gain value to calibrate

Return value:    0 = Function was successful
                <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_adj1125cal NI-DAQmx 8.5: **DAQmxAdjust1125Cal**

Purpose: Adjusts the external calibration constants for the SCXI-1125 module

HTBasic Definition:

```
LONG Result=FNHx_adj1125cal((LONG Calhandle), (REAL Refvoltage), (REAL Measoutput))
```

```
Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Refvoltage         The known voltage, in volts, to use as a reference for calibration
 Measoutput         The voltage measured at the output of the module

Return value:    0 = Function was successful
                <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_set1126cal NI-DAQmx 8.3: **DAQmxSetup1126Cal**

Purpose: Specifies the channel and upper frequency limit on the SCXI-1126 module for calibration

HTBasic Definition:

```
LONG Result=FNHx_set1126cal((LONG Calhandle), (Channel$), (REAL Upperfreqlimit))

Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Channel$           The physical channel to calibrate
 Upperfreqlimit     The high frequency limit in hertz, with 0 Hz as the low frequency
                    limit, which most closely encapsulates the ranges to be calibrated


Return value:    0 = Function was successful
                <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_adj1126cal NI-DAQmx 8.3: **DAQmxAdjust1126Cal**

Purpose: Adjusts the calibration constants for the SCXI-1126 module

HTBasic Definition:

```
LONG Result=FNHx_adj1126cal((LONG Calhandle), (REAL Reffreq), (REAL Measoutput))

Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Reffreq            The known frequency, in Hertz, to use as a reference for calibration
 Measoutput         The voltage measured at the output of the module


Return value:    0 = Function was successful
                <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_set1141cal NI-DAQmx 8.3: **DAQmxSetup1141Cal**

Purpose: Sets the SCXI-1141 module to the specified gain value.

HTBasic Definition:

```
LONG Result=FNHx_set1141cal((LONG Calhandle), (Channel$), (REAL Gain))

Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Channel$           The physical channel to calibrate
 Gain               The gain value to calibrate


Return value:    0 = Function was successful
                <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_adj1141cal NI-DAQmx 8.3: **DAQmxAdjust1141Cal**

Purpose: Adjusts the external calibration constants for the SCXI-1141 module

HTBasic Definition:

```
LONG Result=FNHx_adj1141cal((LONG Calhandle), (REAL Refvoltage), (REAL Measoutput))

Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Refvoltage         The known voltage, in volts, to use as a reference for calibration
 Measoutput         The voltage measured at the output of the module


Return value:    0 = Function was successful
                <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_set1142cal NI-DAQmx 8.3: **DAQmxSetup1142Cal**

Purpose: Sets the SCXI-1142 module to the specified gain value.

HTBasic Definition:

```
LONG Result=FNHx_set1142cal((LONG Calhandle), (Channel$), (REAL Gain))

Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Channel$           The physical channel to calibrate
 Gain               The gain value to calibrate

Return value:    0 = Function was successful
                <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_adj1142cal NI-DAQmx 8.3: **DAQmxAdjust1142Cal**

Purpose: Adjusts the external calibration constants for the SCXI-1142 module

HTBasic Definition:

```
LONG Result=FNHx_adj1142cal((LONG Calhandle), (REAL Refvoltage), (REAL Measoutput))

Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Refvoltage         The known voltage, in volts, to use as a reference for calibration
 Measoutput         The voltage measured at the output of the module

Return value:    0 = Function was successful
                <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_set1143cal NI-DAQmx 8.3: **DAQmxSetup1143Cal**

Purpose: Sets the SCXI-1143 module to the specified gain value.

HTBasic Definition:

```
LONG Result=FNHx_set1143cal((LONG Calhandle), (Channel$), (REAL Gain))

Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Channel$           The physical channel to calibrate
 Gain               The gain value to calibrate

Return value:    0 = Function was successful
                <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_adj1143cal NI-DAQmx 8.3: **DAQmxAdjust1143Cal**

Purpose: Adjusts the external calibration constants for the SCXI-1143 module

HTBasic Definition:

```
LONG Result=FNHx_adj1143cal((LONG Calhandle), (REAL Refvoltage), (REAL Measoutput))

Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Refvoltage         The known voltage, in volts, to use as a reference for calibration
 Measoutput         The voltage measured at the output of the module

Return value:    0 = Function was successful
                <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_set1502cal NI-DAQmx 8.5: **DAQmxSetup1502Cal**

Purpose: Sets the SCXI-1502 module to the specified gain value.

HTBasic Definition:

```
LONG Result=FNHx_set1502cal((LONG Calhandle), (Channel$), (REAL Gain))
```

```
Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Channel$           The physical channel to calibrate
 Gain               The gain value to calibrate


Return value:    0 = Function was successful
                <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_adj1502cal NI-DAQmx 8.5: **DAQmxAdjust1502Cal**

Purpose: Adjusts the external calibration constants for the SCXI-1502 module

HTBasic Definition:

```
LONG Result=FNHx_adj1502cal((LONG Calhandle), (REAL Refvoltage), (REAL Measoutput))
```

```
Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Refvoltage         The known voltage, in volts, to use as a reference for calibration
 Measoutput         The voltage measured at the output of the module


Return value:    0 = Function was successful
                <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_set1503cal NI-DAQmx 8.5: **DAQmxSetup1503Cal**

Purpose: Sets the SCXI-1503 module to the specified gain value.

HTBasic Definition:

```
LONG Result=FNHx_set1503cal((LONG Calhandle), (Channel$), (REAL Gain))
```

```
Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Channel$           The physical channel to calibrate
 Gain               The gain value to calibrate


Return value:    0 = Function was successful
                <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_adj1503cal NI-DAQmx 8.5: **DAQmxAdjust1503Cal**

Purpose: Adjusts the external calibration constants for the SCXI-1503 module

HTBasic Definition:

```
LONG Result=FNHx_adj1503cal((LONG Calhandle), (REAL Refvoltage), (REAL Measoutput))
```

```
Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Refvoltage         The known voltage, in volts, to use as a reference for calibration
 Measoutput         The voltage measured at the output of the module


Return value:    0 = Function was successful
                <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_adj1503ccl — NI-DAQmx 8.5: **DAQmxAdjust1503CurrentCal**

Purpose: Adjusts the external calibration constants for the SCXI-1503 module

HTBasic Definition:

```
LONG Result=FNHx_adj1503ccl((LONG Calhandle), (Channel$), (REAL Measurrent))

Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Channel$           The physical channel to calibrate
 Measurrent         The current measured at the output of the module

Return value:    0 = Function was successful
                <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_set1520cal — NI-DAQmx 8.5: **DAQmxSetup1520Cal**

Purpose: Sets the SCXI-1520 module to the specified gain value.

HTBasic Definition:

```
LONG Result=FNHx_set1503cal((LONG Calhandle), (Channel$), (REAL Gain))

Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Channel$           The physical channel to calibrate
 Gain               The gain value to calibrate

Return value:    0 = Function was successful
                <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_adj1520cal — NI-DAQmx 8.5: **DAQmxAdjust1520Cal**

Purpose: Adjusts the external calibration constants for the SCXI-1520 module

HTBasic Definition:

```
LONG Result=FNHx_adj1520cal((LONG Calhandle), (REAL Refvoltage), (REAL Measoutput))

Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Refvoltage         The known voltage, in volts, to use as a reference for calibration
 Measoutput         The voltage measured at the output of the module

Return value:    0 = Function was successful
                <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_set1102cal NI-DAQmx 8.5: **DAQmxSetup1102Cal**

Purpose: Sets the SCXI-1102 module to the specified gain value.

HTBasic Definition:

```
LONG Result=FNHx_set1102cal((LONG Calhandle), (Channel$), (REAL Gain))

Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Channel$           The physical channel to calibrate
 Gain               The gain value to calibrate


Return value:    0 = Function was successful
                <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_adj1102cal NI-DAQmx 8.5: **DAQmxAdjust1102Cal**

Purpose: Adjusts the external calibration constants for the SCXI-1102 module

HTBasic Definition:

```
LONG Result=FNHx_adj1102cal((LONG Calhandle), (REAL Refvoltage), (REAL Measoutput))

Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Refvoltage         The known voltage, in volts, to use as a reference for calibration
 Measoutput         The voltage measured at the output of the module


Return value:    0 = Function was successful
                <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_set153xcal NI-DAQmx 8.3: **DAQmxSetup153xCal**

Purpose: Sets the SCXI-153x module to the specified gain value.

HTBasic Definition:

```
LONG Result=FNHx_set153xcal((LONG Calhandle), (Channel$), (REAL Gain))

Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Channel$           The physical channel to calibrate
 Gain               The gain value to calibrate


Return value:    0 = Function was successful
                <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_adj153xcal NI-DAQmx 8.3: **DAQmxAdjust153xCal**

Purpose: Adjusts the external calibration constants for the SCXI-153x module

HTBasic Definition:

```
LONG Result=FNHx_adj153xcal((LONG Calhandle), (REAL Refvoltage), (REAL Measoutput))

Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Refvoltage         The known voltage, in volts, to use as a reference for calibration
 Measoutput         The voltage measured at the output of the module


Return value:    0 = Function was successful
                <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_set1540cal NI-DAQmx 8.3: **DAQmxSetup1540Cal**

Purpose: Sets the SCXI-1540 module to the specified gain value.

HTBasic Definition:

```
LONG Result=FNHx_set1540cal((LONG Calhandle), (Channel$), (REAL Excvolt),
                            (REAL Excfreq))

Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Channel$           The physical channel to calibrate
 Excvolt            The RMS value of the internal AC excitation voltage
 Excfreq            The frequency of the internal AC excitation voltage

Return value:   0 = Function was successful
               <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_adj1540cal NI-DAQmx 8.3: **DAQmxAdjust1540Cal**

Purpose: Adjusts the external calibration constants for the SCXI-1540 module

HTBasic Definition:

```
LONG Result=FNHx_adj1540cal((LONG Calhandle), (REAL Refvoltage), (REAL Measoutput),
                            (LONG Inputcalsrc))

Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Refvoltage         The excitation RMS voltage measured from the front of the module
 Measoutput         The voltage measured at the output of the module
 Inputcalsrc        The calibration input source selection

Return value:   0 = Function was successful
               <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_set1122cal NI-DAQmx 8.7: **DAQmxSetup1122Cal**

Purpose: Sets the SCXI-1122 module to the specified gain value.

HTBasic Definition:

```
LONG Result=FNHx_set1122cal((LONG Calhandle), (Channel$), (REAL Gain))

Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Channel$           The physical channel to calibrate
 Gain               The gain value to calibrate

Return value:   0 = Function was successful
               <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.7 and higher.

FNHx_adj1122cal NI-DAQmx 8.7: **DAQmxAdjust1122Cal**

Purpose: Adjusts the external calibration constants for the SCXI-1122 module

HTBasic Definition:
```
LONG Result=FNHx_adj1122cal((LONG Calhandle), (REAL Refvoltage), (REAL Measoutput))
```

```
Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Refvoltage         The known voltage, in volts, to use as a reference for calibration
 Measoutput         The voltage measured at the output of the module

Return value:    0 = Function was successful
                <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.7 and higher.

FNHx_set1521cal NI-DAQmx 8.7: **DAQmxSetup1521Cal**

Purpose: Sets the SCXI-1521 module to the specified gain value.

HTBasic Definition:
```
LONG Result=FNHx_set1521cal((LONG Calhandle), (Channel$), (REAL Gain))
```

```
Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Channel$           The physical channel to calibrate
 Gain               The gain value to calibrate

Return value:    0 = Function was successful
                <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.7 and higher.

FNHx_adj1521cal NI-DAQmx 8.7: **DAQmxAdjust1521Cal**

Purpose: Adjusts the external calibration constants for the SCXI-1521 module

HTBasic Definition:
```
LONG Result=FNHx_adj1521cal((LONG Calhandle), (REAL Refvoltage), (REAL Measoutput))
```

```
Input:
 Calhandle          A reference to the calibration session that you created using the
                    DAQmxInitExtCal function
 Refvoltage         The known voltage, in volts, to use as a reference for calibration
 Measoutput         The voltage measured at the output of the module

Return value:    0 = Function was successful
                <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.7 and higher.

FNHx_perfbrofncex NI-DAQmx 8.7: **DAQmxPerformBridgeOffsetNullingCalEx**

Purpose: Performs a bridge offset nulling calibration on the channels in the task

HTBasic Definition:
```
LONG Result=FNHx_perfbrofncex((LONG Taskid), Channel$, (LONG Skipsupchans))
```

```
Input:
 Taskid             The task used in this function
 Channel$           A subset of virtual channels in the task that you want to calibrate
 Skipsupchans       Specifies whether or not to skip channels that do not support
                    calibration

Return value:    0 = Function was successful
                <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.7 and higher.

FNHx_perfstrshcal — NI-DAQmx 8.7: **DAQmxPerformStrainShuntCal**

Purpose: Performs shunt calibration for the specified channels using a strain gage sensor

HTBasic Definition:

```
LONG Result=FNHx_perfstrshcal((LONG Taskid), Chan$, (REAL Shuntresval),
                              (LONG Shuntresloc), (LONG Skipsupchans))
```

```
Input:
 Taskid            The task used in this function
 Chan$             A subset of virtual channels in the task that you want to calibrate
 Shuntresval       The resistance, in ohms, of the shunt resistor
 Shuntresloc       The location of the shunt resistor
 Skipsupchans      Specifies whether or not to skip channels that do not support
                   calibration

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.7 and higher.

FNHx_perfbrishcal — NI-DAQmx 8.7: **DAQmxPerformBridgeShuntCal**

Purpose: Performs shunt calibration for the specified channels using a bridge sensor

HTBasic Definition:

```
LONG Result=FNHx_perfbrishcal((LONG Taskid), Chan$, (REAL Shuntresval),
                              (LONG Shuntresloc), (REAL Bridgeres), (LONG Skipsupchans))
```

```
Input:
 Taskid            The task used in this function
 Chan$             A subset of virtual channels in the task that you want to calibrate
 Shuntresval       The resistance, in ohms, of the shunt resistor
 Shuntresloc       The location of the shunt resistor
 Bridgeres         The resistance, in ohms, of the bridge sensor
 Skipsupchans      Specifies whether or not to skip channels that do not support
                   calibration

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.7 and higher.

FNHx_getaichcpfco — **DAQmxGetAIChanCalPolyForwardCoeff**

Purpose: Retrieves the forward polynomial values used for calibrating the channel

HTBasic Definition:

```
LONG Result=FNHx_getaichcpfco((LONG Taskid), Channel$, REAL Data, (LONG Arraysizesam))
```

```
Input:
 Taskid            The task ID for which the function should be executed
 Channel$          The physical channel to calibrate
 Data              The value
 Arraysizesam      Array size in samples

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_setaichcpfco — DAQmxSetAIChanCalPolyForwardCoeff

Purpose: Specifies the forward polynomial values used for calibrating the channel

HTBasic Definition:

```
LONG Result=FNHx_setaichcpfco((LONG Taskid), Channel$, REAL Data, (LONG Arraysizesam))
```

```
Input:
 Taskid            The task ID for which the function should be executed
 Channel$          The physical channel to calibrate
 Data              The value to calibrate
 Arraysizesam      Array size in samples

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_rstaichcpfco — DAQmxResetAIChanCalPolyForwardCoeff

Purpose: Resets the forward polynomial values used for calibrating the channel

HTBasic Definition:

```
LONG Result=FNHx_rstaichcpfco((LONG Taskid), Channel$)
```

```
Input:
 Taskid            The task ID for which the function should be executed
 Channel$          The physical channel to calibrate

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_getaidevscco — DAQmxGetAIDevScalingCoeff

Purpose: Indicates the coefficients of a polynomial equation that NI-DAQmx uses to scale values from the native format of the device to volts. Each element of the array corresponds to a term of the equation. For example, if index two of the array is 4, the third term of the equation is 4x^2. Scaling coefficients do not account for any custom scales or sensors contained by the channel.

HTBasic Definition:

```
LONG Result=FNHx_getaidevscco((LONG Taskid), Channel$, REAL Data, (LONG Arraysizesam))
```

```
Input:
 Taskid            The task ID for which the function should be executed
 Channel$          The physical channel
 Data              The value
 Arraysizesam      Array size in samples

Return value:   0 = Function was successful
              <>0 = Error occured
```

NI-DAQmx Specific Attribute Get/Set/Reset Function Declarations

FNHx_getphyschnam — NI-DAQmx: **DAQmxGetPhysicalChanName**

Purpose: Returns the physical channel name of the task

HTBasic Definition:

```
LONG Result=FNHx_getphyschnam((LONG Taskid), Channel$, Dat$, (LONG Bufsize))
```

```
Input:
 Taskid            The task used in this function
 Channel$          Logical channel name
 Bufsize           Buffer size of Dat$


Return value:   0 = Function was successful
                    Dat$ contains the physical channel name
              <>0 = Error occured
```

Example:

```
 LONG Result, Taskid, Bufsize
 DIM Channel$[255], Dat$[255]
 !
 Bufsize=MAXLEN(Dat$)                                    ! buffersize
 Channel$=""
 Result=FNHx_getphyschnam((Taskid),Channel$,Dat$,(Bufsize))
 !
 IF Result=0 THEN
    PRINT "Physical channel name: ";Dat$
 END IF
```

FNHx_setciprescal — NI-DAQmx: **DAQmxSetCIPrescaler**

Purpose: Sets the Counter Input Prescaler property

HTBasic Definition:

```
LONG Result=FNHx_setciprescal((LONG Taskid), Channel$, (LONG Prescaler))
```

```
Input:
 Taskid            The task used in this function
 Channel$          Logical channel name
 Prescaler         Specifies the divisor to apply to the signal you connect to the
                   counter source terminal


Return value:   0 = Function was successful
              <>0 = Error occured
```

Example:

```
 LONG Result, Taskid, Prescaler
 DIM Channel$[255]
 !
 Channel$="Dev1/ctr0"
 Prescaler=2                          ! sets prescaler (=divisor) to 2
 Result=FNHx_setciprescal((Taskid),Channel$,(Prescaler))
```

Device Control

FNHx_resetdevice	NI-DAQmx: **DAQmxResetDevice**

Purpose: Immediately aborts all tasks associated with a device and returns the device to an initialized state. Aborting a task stops and releases any resources the task reserved

HTBasic Definition:

```
LONG Result=FNHx_resetdevice(Devicename$)

Input:
 Devicename$       The name of the device, as configured in MAX, to which this
                   operation applies


Return value:   0 = Function was successful
              <>0 = Error occured
```

Example:

```
 LONG Result
 DIM Devicename$[255]
 !
 Devicename$="Dev1"
 Result=FNHx_resetdevice(Devicename$)
 !
 IF Result=0 THEN
    PRINT "Device '"&Devicename$&"' has been reset."
 END IF
```

Switch Functions

FNHx_swcrscanlist — NI-DAQmx: **DAQmxSwitchCreateScanList**

Purpose: Creates a new switch scanning task that uses the specified scan list and applies it to the specified task

HTBasic Definition:

```
LONG Result=FNHx_swcrscanlist((Scanlist$), LONG Taskid)
```

```
Input:
 Scanlist$          Uses a special syntax to specify the sequence of connections and
                    disconnections for the task

Return value:    0 = Function was successful
                       Taskid now contains Task ID of newly created task
                 <>0 = Error occured
```

FNHx_swconnect — NI-DAQmx: **DAQmxSwitchConnect**

Purpose: Makes a connection between two switch channels. When using this function, specify only the two connection endpoints using **Swchan1** and **Swchan2**

HTBasic Definition:

```
LONG Result=FNHx_swconnect((Swchan1$), (Swchan2$), (LONG Waitforsettling))
```

```
Input:
 Swchan1$           The first channel to connect
 Swchan2$           The second channel to connect
 Waitforsettling    If TRUE, this function waits for the switches to settle before
                    returning. If FALSE, the function returns immediately after the
                    operation

Return value:    0 = Function was successful
                 <>0 = Error occured
```

FNHx_swconnectmul — NI-DAQmx: **DAQmxSwitchConnectMulti**

Purpose: Makes one or more connections specified by a connection list. You can specify only the two endpoints, or you can specify the explicit path between two endpoints

HTBasic Definition:

```
LONG Result=FNHx_swconnectmul((Connlist$), (LONG Waitforsettling))
```

```
Input:
 Connlist$          A list of connections, using a special syntax, to make between switch
                    channels
 Waitforsettling    If TRUE, this function waits for the switches to settle before
                    returning. If FALSE, the function returns immediately after the
                    operation

Return value:    0 = Function was successful
                 <>0 = Error occured
```

FNHx_swdisconnect — NI-DAQmx: **DAQmxSwitchDisconnect**

Purpose: Disconnects two switch channels

HTBasic Definition:

```
LONG Result=FNHx_swdisconnect((Swchan1$), (Swchan2$), (LONG Waitforsettling))
```

```
Input:
 Swchan1$           The first channel to disconnect
 Swchan2$           The second channel to disconnect
 Waitforsettling    If TRUE, this function waits for the switches to settle before
                    returning. If FALSE, the function returns immediately after the
                    operation

Return value:    0 = Function was successful
                 <>0 = Error occured
```

FNHx_swdisconnall NI-DAQmx: **DAQmxSwitchDisconnectAll**

Purpose: Terminates all active connections on the device, which places the relays into the topology reset state

HTBasic Definition:

```
LONG Result=FNHx_swdisconnall((Devicename$), (LONG Waitforsettling))

Input:
 Devicename$       The name of the device, as configured in MAX, to which this
                   operation applies
 Waitforsettling   If TRUE, this function waits for the switches to settle before
                   returning. If FALSE, the function returns immediately after the
                   operation

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_swsettopares NI-DAQmx: **DAQmxSwitchSetTopologyAndReset**

Purpose: Resets a switch device and sets its topology to the one specified with newTopology

HTBasic Definition:

```
LONG Result=FNHx_swsettopares((Devicename$), (Newtopology$))

Input:
 Devicename$       The name of the device, as configured in MAX, to which this
                   operation applies
 Newtopology$      The switch topology to use on the device

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_swfindpath NI-DAQmx: **DAQmxSwitchFindPath**

Purpose: Returns information about the path between `Swchan1$` and `Swchan2$`

HTBasic Definition:

```
LONG Result=FNHx_swfindpath((Swchan1$), (Swchan2$), Path$, (LONG Pathbufsize),
                            (LONG Pathstatus))

Input:
 Swchan1$          The first channel to connect
 Swchan2$          The second channel to connect
 Pathbufsize       The size of path. If you pass 0, this function returns the size of
                   the buffer needed to allocate

Return value:   0 = Function was successful
                    Pathstatus   The status of the requested path
                    Path$        The existing path or an available path between
                                 Swchan1$ and Swchan2$
              <>0 = Error occured
```

FNHx_swopenrelays NI-DAQmx: **DAQmxSwitchOpenRelays**

Purpose: Opens the specified relays

HTBasic Definition:

```
LONG Result=FNHx_swopenrelays((Relays$), (LONG Waitforsettling))

Input:
 Relays$           A set of relays to open
 Waitforsettling   If TRUE, this function waits for the switches to settle before
                   returning. If FALSE, the function returns immediately after the
                   operation

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_swclosrelays — NI-DAQmx: DAQmxSwitchCloseRelays

Purpose: Closes the specified relays

HTBasic Definition:

```
LONG Result=FNHx_swclosrelays((Relays$), (LONG Waitforsettling))
```

```
Input:
 Relays$           A set of relays to close
 Waitforsettling   If TRUE, this function waits for the switches to settle before
                   returning. If FALSE, the function returns immediately after the
                   operation

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_swwaitforset — NI-DAQmx: DAQmxSwitchWaitForSettling

Purpose: Waits for the settling time on the device to expire

HTBasic Definition:

```
LONG Result=FNHx_swwaitforset((Devicename$))
```

```
Input:
 Devicename$       The name of the device, as configured in MAX, to which this
                   operation applies

Return value:   0 = Function was successful
              <>0 = Error occured
```

Signal Routing

FNHx_connterms NI-DAQmx: **DAQmxConnectTerms**

Purpose: Creates a route between a source and destination terminal

HTBasic Definition:

```
LONG Result=FNHx_connterms((Srcterm$), (Destterm$), (LONG Signalmodifiers))
```

```
Input:
 Srcterm$          The originating terminal of the route. You can specify a
                   terminal name
 Destterm$         The receiving terminal of the route. You can specify a terminal name
 Signalmodifiers   Specifies whether or not to invert the signal routed from the
                   sourceTerminal to the destinationTerminal

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_disconnterms NI-DAQmx: **DAQmxDisconnectTerms**

Purpose: Removes signal routes previously created using DAQmxConnectTerms

HTBasic Definition:

```
LONG Result=FNHx_disconnterms((Srcterm$), (Destterm$))
```

```
Input:
 Srcterm$          The originating terminal of the route. You can specify a
                   terminal name
 Destterm$         The receiving terminal of the route. You can specify a terminal name

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_tristoutterm NI-DAQmx: **DAQmxTristateOutputTerm**

Purpose: Removes signal routes previously created using DAQmxConnectTerms

HTBasic Definition:

```
LONG Result=FNHx_tristoutterm((Outputterm$))
```

```
Input:
 Outputterm$       The terminal on the I/O connector to set to high-impedance state.
                   You can specify a terminal name

Return value:   0 = Function was successful
              <>0 = Error occured
```

TEDS

FNHx_configteds NI-DAQmx: **DAQmxConfigureTEDS**

Purpose: Associates TEDS information with the physical channel you specify. If you do not specify the filename of a data sheet in the filePath parameter, this function attempts to find a TEDS sensor connected to the physical channel. This function temporarily overrides any TEDS configuration for the physical channel that you performed in MAX

HTBasic Definition:

```
LONG Result=FNHx_configteds(Physchan$, Filepath$)

Input:
 Physchan$          The name of the physical channel you want to configure
 Filepath$          The path to a Virtual TEDS data sheet that you want to associate with
                    the physical channel. If you do not specify the filename of a data
                    sheet, this function attempts to find a TEDS sensor connected to
                    the physical channel

Return value:    0 = Function was successful
                <>0 = Error occured
```

FNHx_clearteds NI-DAQmx: **DAQmxClearTEDS**

Purpose: Removes TEDS information from the physical channel you specify. This function temporarily overrides any TEDS configuration for the physical channel that you performed in MAX

HTBasic Definition:

```
LONG Result=FNHx_clearteds(Physchan$)

Input:
 Physchan$          The name of the physical channel you want to clear

Return value:    0 = Function was successful
                <>0 = Error occured
```

FNHx_exportsignal	NI-DAQmx: **DAQmxExportSignal**

Purpose: Creates a new switch scanning task that uses the specified scan list and applies it to the specified task

HTBasic Definition:

```
LONG Result=FNHx_exportsignal((LONG Taskid), (LONG Signalid), (Outputterminal$))

Input:
 Taskid            The task used in this function
 Signalid          The name of the trigger, clock, or event to export
 Outputterminal$   The destination terminal of the exported signal


Return value:   0 = Function was successful
              <>0 = Error occured
```

Buffer

FNHx_getbufinpsiz NI-DAQmx: **DAQmxGetBufInputBufSize**

Purpose: Determines the number of samples the input buffer can hold for each channel in the task. Zero indicates to allocate no buffer. Use a buffer size of 0 to perform a hardware-timed operation without using a buffer. Setting this property overrides the automatic input buffer allocation that NI-DAQmx performs

HTBasic Definition:

```
LONG Result=FNHx_getbufinpsiz(((LONG Taskid), LONG Bufsize)

Input:
 Taskid            The task used in this function

Return value:   0 = Function was successful
                    Bufsize   Size of buffer in samples
              <>0 = Error occured
```

FNHx_setbufinpsiz NI-DAQmx: **DAQmxSetBufInputBufSize**

Purpose: Specifies the number of samples the input buffer can hold for each channel in the task. Zero indicates to allocate no buffer. Use a buffer size of 0 to perform a hardware-timed operation without using a buffer. Setting this property overrides the automatic input buffer allocation that NI-DAQmx performs

HTBasic Definition:

```
LONG Result=FNHx_getbufinpsiz(((LONG Taskid), LONG Bufsize)

Input:
 Taskid            The task used in this function
 Bufsize           Size of buffer in samples

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_getbufoutsiz NI-DAQmx: **DAQmxGetBufOutputBufSize**

Purpose: Determines the number of samples the output buffer can hold for each channel in the task. Zero indicates to allocate no buffer. Use a buffer size of 0 to perform a hardware-timed operation without using a buffer. Setting this property overrides the automatic output buffer allocation that NI-DAQmx performs

HTBasic Definition:

```
LONG Result=FNHx_getbufoutsiz(((LONG Taskid), LONG Bufsize)

Input:
 Taskid            The task used in this function

Return value:   0 = Function was successful
                    Bufsize   Size of buffer in samples
              <>0 = Error occured
```

FNHx_setbufoutsiz NI-DAQmx: **DAQmxSetBufOutputBufSize**

Purpose: Specifies the number of samples the output buffer can hold for each channel in the task. Zero indicates to allocate no buffer. Use a buffer size of 0 to perform a hardware-timed operation without using a buffer. Setting this property overrides the automatic output buffer allocation that NI-DAQmx performs

HTBasic Definition:

```
LONG Result=FNHx_setbufoutsiz(((LONG Taskid), LONG Bufsize)

Input:
 Taskid            The task used in this function
 Bufsize           Size of buffer in samples

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_getobufoutsz NI-DAQmx: **DAQmxGetBufOutputOnbrdBufSize**

Purpose: Determines in samples per channel the size of the onboard output buffer of the device

HTBasic Definition:

```
LONG Result=FNHx_getobufoutsz(((LONG Taskid), LONG Bufsize)

Input:
 Taskid            The task used in this function

Return value:   0 = Function was successful
                    Bufsize   Size of buffer in samples
              <>0 = Error occured
```

FNHx_setobufoutsz NI-DAQmx: **DAQmxSetBufOutputOnbrdBufSize**

Purpose: Specifies in samples per channel the size of the onboard output buffer of the device

HTBasic Definition:

```
LONG Result=FNHx_setobufoutsz(((LONG Taskid), LONG Bufsize)

Input:
 Taskid            The task used in this function
 Bufsize           Size of buffer in samples

Return value:   0 = Function was successful
              <>0 = Error occured
```

Other Get/Set specific functions (Part I)

FNHx_getaimax NI-DAQmx: **DAQmxGetAIMax**

Purpose: Returns the coerced maximum value that the device can measure with the current settings

HTBasic Definition:

```
LONG Result=FNHx_getaimax((LONG Taskid), (Channel$), REAL Aidata)

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to query

Return value:   0 = Function was successful
                      Aidata    Maximum value you expect to measure (in Units)
              <>0 = Error occured
```

FNHx_getaimin NI-DAQmx: **DAQmxGetAIMin**

Purpose: Returns the coerced minimum value that the device can measure with the current settings

HTBasic Definition:

```
LONG Result=FNHx_getaimin((LONG Taskid), (Channel$), REAL Aidata)

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to query

Return value:   0 = Function was successful
                      Aidata    Minimum value you expect to measure (in Units)
              <>0 = Error occured
```

FNHx_getailpcofrq NI-DAQmx: **DAQmxGetAILowpassCutoffFreq**

Purpose: Returns the frequency in Hertz that corresponds to the -3dB cutoff of the filter

HTBasic Definition:

```
LONG Result=FNHx_getailpcofrq((LONG Taskid), (Channel$), REAL Fdata)

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to query

Return value:   0 = Function was successful
                      Fdata   Frequency that corresponds to the -3dB cutoff of the filter
              <>0 = Error occured
```

FNHx_setailpcofrq NI-DAQmx: **DAQmxSetAILowpassCutoffFreq**

Purpose: Specifies the frequency in Hertz that corresponds to the -3dB cutoff of the filter

HTBasic Definition:

```
LONG Result=FNHx_setailpcofrq((LONG Taskid), (Channel$), (REAL Fdata))

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to set
 Fdata             Frequency that corresponds to the -3dB cutoff of the filter

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_getailwpenab NI-DAQmx: **DAQmxGetAILowpassEnable**

Purpose: Determines whether to enable the lowpass filter of the channel

HTBasic Definition:

```
LONG Result=FNHx_getailwpenab((LONG Taskid), (Channel$), LONG Ldata)

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to set

Return value:   0 = Function was successful
                      Ldata    Specifies whether to enable the lowpass filter
                               of the channel
              <>0 = Error occured
```

FNHx_setailwpenab NI-DAQmx: **DAQmxSetAILowpassEnable**

Purpose: Specifies whether to enable the lowpass filter of the channel

HTBasic Definition:

```
LONG Result=FNHx_setailwpenab((LONG Taskid), (Channel$), (LONG Ldata))

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to set
 Ldata             Specifies whether to enable the lowpass filter of the channel

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_getanedsthys NI-DAQmx: **DAQmxGetAnlgEdgeStartTrigHyst**

Purpose: Determines the hysteresis level in the units of the measurement or generation

HTBasic Definition:

```
LONG Result=FNHx_getanedsthys((LONG Taskid), (Channel$), REAL Hdata)

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to set

Return value:   0 = Function was successful
                      Hdata   Hysteresis level in the units of the measurement
                           or generation
              <>0 = Error occured
```

FNHx_setanedsthys NI-DAQmx: **DAQmxSetAnlgEdgeStartTrigHyst**

Purpose: Specifies a hysteresis level in the units of the measurement or generation

HTBasic Definition:

```
LONG Result=FNHx_setanedsthys((LONG Taskid), (Channel$), (REAL Hdata))

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to set
 Hdata             Specifies whether to enable the lowpass filter of the channel

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_getaiacexsen NI-DAQmx: **DAQmxGetAIACExcitSyncEnable**

Purpose: Determines whether to synchronize the AC excitation source of the channel to that of another channel

HTBasic Definition:

```
LONG Result=FNHx_getaiacexsen((LONG Taskid), (Channel$), LONG Edata)

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to get

Return value:   0 = Function was successful
                      Edata   Determines whether to synchronize the AC excitation source
                              of the channel to that of another channel
              <>0 = Error occured
```

FNHx_setaiacexsen NI-DAQmx: **DAQmxSetAIACExcitSyncEnable**

Purpose: Specifies whether to synchronize the AC excitation source of the channel to that of another channel

HTBasic Definition:

```
LONG Result=FNHx_setaiacexsen((LONG Taskid), (Channel$), (LONG Edata))

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to set
 Edata             Specifies whether to synchronize the AC excitation source of the
                   channel to that of another channel

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_getrdovchexi NI-DAQmx: **DAQmxGetReadOverloadedChansExist**

Purpose: Indicates if the device detected an overload in any channel in the task

HTBasic Definition:

```
LONG Result=FNHx_getrdovchexi((LONG Taskid), LONG Odata)

Input:
 Taskid            The task used in this function

Return value:   0 = Function was successful
                      Odata    Indicates if the device detected an overload in any
                               channel in the task
              <>0 = Error occured
```

FNHx_getrdovchans NI-DAQmx: **DAQmxGetReadOverloadedChans**

Purpose: Indicates the names of any overloaded virtual channels in the task. You must read Overloaded Channels Exist before you read this property. Otherwise, you will receive an error

HTBasic Definition:

```
LONG Result=FNHx_getrdovchans((LONG Taskid), Odata$, (LONG Buffersize))

Input:
 Taskid            The task used in this function
 Buffersize        Size of the buffer Odata$

Return value:   0 = Function was successful
                      Odata$   Names of overloaded virtual channels in the task
              <>0 = Error occured
```

FNHx_getdiglvptsc NI-DAQmx: **DAQmxGetDigLvlPauseTrigSrc**

Purpose: Determines the name of a terminal where there is a digital signal to use as the source of the Pause Trigger

HTBasic Definition:

```
LONG Result=FNHx_getdiglvptsc((LONG Taskid), Ldata$, (LONG Buffersize))

Input:
 Taskid            The task used in this function
 Buffersize        Size of the buffer Odata$

Return value:   0 = Function was successful
                    Ldata$   Determines the name of the terminal where there is a
                             digital signal to use as the source of the Pause Trigger
              <>0 = Error occured
```

FNHx_setdiglvptsc NI-DAQmx: **DAQmxSetDigLvlPauseTrigSrc**

Purpose: Specifies the name of a terminal where there is a digital signal to use as the source of the Pause Trigger

HTBasic Definition:

```
LONG Result=FNHx_setdiglvptsc((LONG Taskid), Ldata$)

Input:
 Taskid            The task used in this function
 Ldata$            Specifies the name of the terminal where there is a digital
                   signal to use as the source of the Pause Trigger

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_getpaustrtyp NI-DAQmx: **DAQmxGetPauseTrigType**

Purpose: Determines the type of trigger to use to pause a task

HTBasic Definition:

```
LONG Result=FNHx_getpaustrtyp((LONG Taskid), LONG Tdata)

Input:
 Taskid            The task used in this function

Return value:   0 = Function was successful
                    Tdata    Returns the type of trigger to use to pause a task
              <>0 = Error occured
```

FNHx_setpaustrtyp NI-DAQmx: **DAQmxSetPauseTrigType**

Purpose: Specifies the type of trigger to use to pause a task

HTBasic Definition:

```
LONG Result=FNHx_setpaustrtyp((LONG Taskid), (LONG Tdata))

Input:
 Taskid            The task used in this function
 Tdata             Specifies the type of trigger to use to pause a task

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_getdlpatrwhn NI-DAQmx: **DAQmxGetDigLvlPauseTrigWhen**

Purpose: Determines whether the task pauses while the signal is high or low

HTBasic Definition:

```
LONG Result=FNHx_getdlpatrwhn((LONG Taskid), LONG Wdata)

Input:
 Taskid            The task used in this function

Return value:   0 = Function was successful
                    Wdata    Determines whether the task pauses while the signal is
                             high or low
              <>0 = Error occured
```

FNHx_setdlpatrwhn NI-DAQmx: **DAQmxSetDigLvlPauseTrigWhen**

Purpose: Specifies whether the task pauses while the signal is high or low

HTBasic Definition:

```
LONG Result=FNHx_setdlpatrwhn((LONG Taskid), (LONG Wdata))

Input:
 Taskid            The task used in this function
 Wdata             Specifies whether the task pauses while the signal is high or low

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_getcigpssysr NI-DAQmx: **DAQmxGetCIGPSSyncSrc**

Purpose: Determines the terminal to which the GPS synchronization signal is connected

HTBasic Definition:

```
LONG Result=FNHx_getcigpssysr((LONG Taskid), (Channel$), Gdata$, (LONG Buffersize))

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to work with
 Buffersize        Size of the buffer Gdata$

Return value:   0 = Function was successful
                    Gdata$   Determines the terminal to which the GPS synchronization
                             signal is connected
              <>0 = Error occured
```

FNHx_setcigpssysr NI-DAQmx: **DAQmxSetCIGPSSyncSrc**

Purpose: Specifies the terminal to which the GPS synchronization signal is connected

HTBasic Definition:

```
LONG Result=FNHx_setcigpssysr((LONG Taskid), (Channel$), Gdata$)

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to set
 Gdata$            Specifies the terminal to which the GPS synchronization signal is
                   connected

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_gethsdelaxfr NI-DAQmx: **DAQmxGetHshkDelayAfterXfer**

Purpose: Determines the number of seconds to wait after a handshake cycle before starting a new handshake cycle

HTBasic Definition:

```
LONG Result=FNHx_gethsdelaxfr((LONG Taskid), REAL Ddata)

Input:
 Taskid            The task used in this function

Return value:   0 = Function was successful
                    Ddata   Number of seconds to wait after a handshake cycle before
                            starting a new handshake cycle
              <>0 = Error occured
```

FNHx_sethsdelaxfr NI-DAQmx: **DAQmxSetHshkDelayAfterXfer**

Purpose: Specifies the number of seconds to wait after a handshake cycle before starting a new handshake cycle

HTBasic Definition:

```
LONG Result=FNHx_sethsdelaxfr((LONG Taskid), (REAL Ddata))

Input:
 Taskid            The task used in this function
 Ddata             Number of seconds to wait after a handshake cycle before starting
                   a new handshake cycle

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_getexhseiddl NI-DAQmx: **DAQmxGetExportedHshkEventInterlockedDeassertDelay**

Purpose: Determines in seconds the amount of time to wait after the Handshake Trigger asserts before deasserting the Handshake Event if Output Behavior is DAQmx_Val_Interlocked

HTBasic Definition:

```
LONG Result=FNHx_getexhseiddl((LONG Taskid), REAL Idata)

Input:
 Taskid            The task used in this function

Return value:   0 = Function was successful
                    Idata   Returns in seconds the amount of time to wait after the
                            Handshake Trigger asserts before deasserting the Handshake
                            Event if Output Behavior is DAQmx_Val_Interlocked
              <>0 = Error occured
```

FNHx_setexhseiddl NI-DAQmx: **DAQmxSetExportedHshkEventInterlockedDeassertDelay**

Purpose: Specifies in seconds the amount of time to wait after the Handshake Trigger asserts before deasserting the Handshake Event if Output Behavior is DAQmx_Val_Interlocked

HTBasic Definition:

```
LONG Result=FNHx_setexhseiddl((LONG Taskid), (REAL Idata))

Input:
 Taskid            The task used in this function
 Idata             Specifies in seconds the amount of time to wait after the Handshake
                   Trigger asserts before deasserting the Handshake Event if Output
                   Behavior is DAQmx_Val_Interlocked

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_getdidgflten NI-DAQmx: **DAQmxGetDIDigFltrEnable**

Purpose: Determines whether to enable the digital filter for the line(s) or port(s). You can enable the filter on a line-by-line basis. You do not have to enable the filter for all lines in a channel

HTBasic Definition:

```
LONG Result=FNHx_getdidgflten((LONG Taskid), (Channel$), LONG Fdata)

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to work with

Return value:   0 = Function was successful
                      Fdata   Determines whether to enable the digital filter for the
                              line(s) or port(s)
              <>0 = Error occured
```

FNHx_setdidgflten NI-DAQmx: **DAQmxSetDIDigFltrEnable**

Purpose: Specifies whether to enable the digital filter for the line(s) or port(s). You can enable the filter on a line-by-line basis. You do not have to enable the filter for all lines in a channel

HTBasic Definition:

```
LONG Result=FNHx_setdidgflten((LONG Taskid), (Channel$), (LONG Fdata))

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to work with
 Fdata             Specifies whether to enable the digital filter for line(s) / port(s)

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_getdidgflmpw NI-DAQmx: **DAQmxGetDIDigFltrMinPulseWidth**

Purpose: Determines in seconds the minimum pulse width the filter recognizes as a valid high or low state transition

HTBasic Definition:

```
LONG Result=FNHx_getdidgflmpw((LONG Taskid), (Channel$), REAL Mdata)

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to work with

Return value:   0 = Function was successful
                      Mdata   Minimum pulse width (in seconds) the filter recognizes as
                              a valid high or low state transition
              <>0 = Error occured
```

FNHx_setdidgflmpw NI-DAQmx: **DAQmxSetDIDigFltrMinPulseWidth**

Purpose: Specifies in seconds the minimum pulse width the filter recognizes as a valid high or low state transition

HTBasic Definition:

```
LONG Result=FNHx_setdidgflmpw((LONG Taskid), (Channel$), (REAL Mdata))

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to work with
 Mdata             Minimum pulse width (in seconds) the filter recognizes as a valid
                   high or low state transition

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_getadvtrgtyp NI-DAQmx: **DAQmxGetAdvTrigType**

Purpose: Determines the type of trigger to use to advance to the next entry in a switch scan list

HTBasic Definition:

```
LONG Result=FNHx_getadvtrgtyp((LONG Taskid), LONG Adata)

Input:
 Taskid            The task used in this function

Return value:   0 = Function was successful
                    Adata   Type of trigger to use to advance to the next entry in a
                            switch scan list
              <>0 = Error occured
```

FNHx_setadvtrgtyp NI-DAQmx: **DAQmxSetAdvTrigType**

Purpose: Specifies the type of trigger to use to advance to the next entry in a switch scan list

HTBasic Definition:

```
LONG Result=FNHx_setadvtrgtyp((LONG Taskid), (LONG Adata))

Input:
 Taskid            The task used in this function
 Adata             Type of trigger to use to advance to the next entry in a switch
                   scan list

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_getswscrpmod NI-DAQmx: **DAQmxGetSwitchScanRepeatMode**

Purpose: Determines if the task advances through the scan list multiple times

HTBasic Definition:

```
LONG Result=FNHx_getswscrpmod((LONG Taskid), LONG Rdata)

Input:
 Taskid            The task used in this function

Return value:   0 = Function was successful
                    Rdata   Determines if the task advances through the scan list
                            multiple times
              <>0 = Error occured
```

FNHx_setswscrpmod NI-DAQmx: **DAQmxSetSwitchScanRepeatMode**

Purpose: Specifies if the task advances through the scan list multiple times

HTBasic Definition:

```
LONG Result=FNHx_getswscrpmod((LONG Taskid), (LONG Rdata))

Input:
 Taskid            The task used in this function
 Rdata             Specifies if the task advances through the scan list multiple times

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_getmastbasrc NI-DAQmx: **DAQmxGetMasterTimebaseSrc**

Purpose: Determines the terminal of the signal to use as the Master Timebase. On an E Series device, you can choose only between the onboard 20MHz Timebase or the RTSI7 terminal

HTBasic Definition:

```
LONG Result=FNHx_getmastbasrc((LONG Taskid), Tdata$, (LONG Buffersize))

Input:
 Taskid            The task used in this function
 Buffersize        Size of the buffer Tdata$

Return value:   0 = Function was successful
                    Tdata$   Determines the terminal of the signal to use as the
                             Master Timebase
              <>0 = Error occured
```

FNHx_setmastbasrc NI-DAQmx: **DAQmxSetMasterTimebaseSrc**

Purpose: Specifies the terminal of the signal to use as the Master Timebase. On an E Series device, you can choose only between the onboard 20MHz Timebase or the RTSI7 terminal

HTBasic Definition:

```
LONG Result=FNHx_setmastbasrc((LONG Taskid), Tdata$)

Input:
 Taskid            The task used in this function
 Tdata$            Specifies the terminal of the signal to use as the Master Timebase

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_getmastbrate NI-DAQmx: **DAQmxGetMasterTimebaseRate**

Purpose: Determines the rate of the Master Timebase

HTBasic Definition:

```
LONG Result=FNHx_getmastbrate((LONG Taskid), (Channel$), REAL Mdata)

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to work with

Return value:   0 = Function was successful
                    Mdata    Determines the rate of the Master Timebase
              <>0 = Error occured
```

FNHx_setmastbrate NI-DAQmx: **DAQmxSetMasterTimebaseRate**

Purpose: Specifies the rate of the Master Timebase

HTBasic Definition:

```
LONG Result=FNHx_setmastbrate((LONG Taskid), (REAL Mdata))

Input:
 Taskid            The task used in this function
 Mdata             Specifies the rate of the Master Timebase

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_getrefclksrc NI-DAQmx: **DAQmxGetRefClkSrc**

Purpose: Determines the terminal of the signal to use as the Reference Clock

HTBasic Definition:

```
LONG Result=FNHx_getrefclksrc((LONG Taskid), Tdata$, (LONG Buffersize))

Input:
 Taskid            The task used in this function
 Buffersize        Size of the buffer Tdata$

Return value:   0 = Function was successful
                    Tdata$   The terminal of the signal to use as the Reference Clock
              <>0 = Error occured
```

FNHx_setrefclksrc NI-DAQmx: **DAQmxSetRefClkSrc**

Purpose: Specifies the terminal of the signal to use as the Reference Clock

HTBasic Definition:

```
LONG Result=FNHx_setrefclksrc((LONG Taskid), Tdata$)

Input:
 Taskid            The task used in this function
 Tdata$            The terminal of the signal to use as the Reference Clock

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_getrefclkrat NI-DAQmx: **DAQmxGetRefClkRate**

Purpose: Determines the frequency of the Reference Clock

HTBasic Definition:

```
LONG Result=FNHx_getrefclkrat((LONG Taskid), (Channel$), REAL Fdata)

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to work with

Return value:   0 = Function was successful
                    Fdata   Determines the frequency of the Reference Clock
              <>0 = Error occured
```

FNHx_setrefclkrat NI-DAQmx: **DAQmxSetRefClkRate**

Purpose: Specifies the frequency of the Reference Clock

HTBasic Definition:

```
LONG Result=FNHx_setrefclkrat((LONG Taskid), (REAL Fdata))

Input:
 Taskid            The task used in this function
 Fdata             Specifies the frequency of the Reference Clock

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_getsclktbsrc NI-DAQmx: **DAQmxGetSampClkTimebaseSrc**

Purpose: Determines the terminal of the signal to use as the Sample Clock Timebase

HTBasic Definition:

```
LONG Result=FNHx_getsclktbsrc((LONG Taskid), Tdata$, (LONG Buffersize))

Input:
 Taskid            The task used in this function
 Buffersize        Size of the buffer Tdata$

Return value:   0 = Function was successful
                    Tdata$   The terminal of the signal to use as the Sample Clock
                             Timebase
              <>0 = Error occured
```

FNHx_setsclktbsrc NI-DAQmx: **DAQmxSetSampClkTimebaseSrc**

Purpose: Specifies the terminal of the signal to use as the Sample Clock Timebase

HTBasic Definition:

```
LONG Result=FNHx_setsclktbsrc((LONG Taskid), Tdata$)

Input:
 Taskid            The task used in this function
 Tdata$            The terminal of the signal to use as the Sample Clock Timebase

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_getsynpulsrc NI-DAQmx: **DAQmxGetSyncPulseSrc**

Purpose: Determines the terminal of the signal to use as the synchronization pulse. The synchronization pulse resets the clock dividers and the ADCs/DACs on the device

HTBasic Definition:

```
LONG Result=FNHx_getsynpulsrc((LONG Taskid), Tdata$, (LONG Buffersize))

Input:
 Taskid            The task used in this function
 Buffersize        Size of the buffer Tdata$

Return value:   0 = Function was successful
                    Tdata$   The terminal of the signal to use as the synchronization
                             pulse
              <>0 = Error occured
```

FNHx_setsynpulsrc NI-DAQmx: **DAQmxSetSyncPulseSrc**

Purpose: Specifies the terminal of the signal to use as the synchronization pulse. The synchronization pulse resets the clock dividers and the ADCs/DACs on the device

HTBasic Definition:

```
LONG Result=FNHx_setsynpulsrc((LONG Taskid), Tdata$)

Input:
 Taskid            The task used in this function
 Tdata$            The terminal of the signal to use as the synchronization pulse

Return value:   0 = Function was successful
              <>0 = Error occured
```

Other Get/Set specific functions (Part II)

FNHx_getaivolbref NI-DAQmx 8.5: **DAQmxGetAIVoltagedBRef**

Purpose: Determines the decibel reference level in the units of the channel

HTBasic Definition:

```
LONG Result=FNHx_getaivolbref((LONG Taskid), (Channel$), REAL Data)

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with

Return value:   0 = Function was successful
                  Data: Specifies the decibel reference level in the units of
                        the channel. When you read samples as a waveform, the decibel
                        reference level is included in the waveform attributes
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_setaivolbref NI-DAQmx 8.5: **DAQmxSetAIVoltagedBRef**

Purpose: Specifies the decibel reference level in the units of the channel

HTBasic Definition:

```
LONG Result=FNHx_setaivolbref((LONG Taskid), (Channel$), (REAL Data))

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with
 Data              Specifies the decibel reference level in the units of the channel.
                   When you read samples as a waveform, the decibel reference level is
                   included in the waveform attributes


Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_rstaivolbref NI-DAQmx 8.5: **DAQmxResetAIVoltagedBRef**

Purpose: Resets the decibel reference level in the units of the channel

HTBasic Definition:

```
LONG Result=FNHx_rstaivolbref((LONG Taskid), (Channel$))

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with


Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_getaitcsctyp NI-DAQmx 8.3: **DAQmxGetAIThrmcplScaleType**

Purpose: Determines the method or equation form that the thermocouple scale uses

HTBasic Definition:

```
LONG Result=FNHx_getaitcsctyp((LONG Taskid), (Channel$), LONG Data)

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with

Return value:   0 = Function was successful
                    Data: Specifies the method or equation form that the thermocouple
                          scale uses

              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_setaitcsctyp NI-DAQmx 8.3: **DAQmxSetAIThrmcplScaleType**

Purpose: Specifies the method or equation form that the thermocouple scale uses

HTBasic Definition:

```
LONG Result=FNHx_setaitcsctyp((LONG Taskid), (Channel$), (LONG Data))

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with
 Data              Specifies the method or equation form that the thermocouple
                   scale uses. Possible values: DAQmx_Val_Polynomial, DAQmx_Val_Table


Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_rstaitcsctyp NI-DAQmx 8.3: **DAQmxResetAIThrmcplScaleType**

Purpose: Resets the method or equation form that the thermocouple scale uses

HTBasic Definition:

```
LONG Result=FNHx_rstaitcsctyp((LONG Taskid), (Channel$))

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with


Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getaispbyref NI-DAQmx 8.5: **DAQmxGetAISoundPressuredBRef**

Purpose: Determines the decibel reference level in the units of the channel

HTBasic Definition:

```
LONG Result=FNHx_getaispbyref((LONG Taskid), (Channel$), REAL Data)

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with

Return value:   0 = Function was successful
                    Data: Specifies the decibel reference level in the units of the
                          channel. When you read samples as a waveform, the decibel
                          reference level is included in the waveform attributes

              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_setaispbyref NI-DAQmx 8.5: **DAQmxSetAISoundPressuredBRef**

Purpose: Specifies the decibel reference level in the units of the channel

HTBasic Definition:

```
LONG Result=FNHx_setaispbyref((LONG Taskid), (Channel$), (REAL Data))

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with
 Data              Specifies the decibel reference level in the units of the channel.
                   When you read samples as a waveform, the decibel reference level is
                   included in the waveform attributes

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_rstaispbyref NI-DAQmx 8.5: **DAQmxResetAISoundPressuredBRef**

Purpose: Resets the decibel reference level in the units of the channel

HTBasic Definition:

```
LONG Result=FNHx_rstaispbyref((LONG Taskid), (Channel$))

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with


Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_getaiacbyref NI-DAQmx 8.5: **DAQmxGetAIAcceldBRef**

Purpose: Determines the decibel reference level in the units of the channel

HTBasic Definition:

```
LONG Result=FNHx_getaiacbyref((LONG Taskid), (Channel$), REAL Data)

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with

Return value:   0 = Function was successful
                    Data: Specifies the decibel reference level in the units of the
                          channel. When you read samples as a waveform, the decibel
                          reference level is included in the waveform attributes
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_setaiacbyref NI-DAQmx 8.5: **DAQmxSetAIAcceldBRef**

Purpose: Specifies the decibel reference level in the units of the channel

HTBasic Definition:

```
LONG Result=FNHx_setaiacbyref((LONG Taskid), (Channel$), (REAL Data))

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with
 Data              Specifies the decibel reference level in the units of the channel.
                   When you read samples as a waveform, the decibel reference level is
                   included in the waveform attributes

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_rstaiacbyref NI-DAQmx 8.5: **DAQmxResetAIAcceldBRef**

Purpose: Resets the decibel reference level in the units of the channel

HTBasic Definition:

```
LONG Result=FNHx_rstaiacbyref((LONG Taskid), (Channel$))

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_getaiisteds NI-DAQmx 8.3: **DAQmxGetAIIsTEDS**

Purpose: Indicates if the virtual channel was initialized using a TEDS bitstream from the corresponding physical channel

HTBasic Definition:

```
LONG Result=FNHx_getaiisteds((LONG Taskid), (Channel$), LONG Data)

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          The source terminal of the Sample Clock

Return value:   0 = Function was successful
                    Data: Indicates if the virtual channel was initialized using a
                          TEDS bitstream from the corresponding physical channel
              <>0 = Error occurred
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getaiadctimd NI-DAQmx 8.5: **DAQmxGetAIADCTimingMode**

Purpose: Determines the ADC Timing mode, controlling the tradeoff between speed and effective resolution

HTBasic Definition:

```
LONG Result=FNHx_getaiadctimd((LONG Taskid), (Channel$), LONG Data)

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with

Return value:   0 = Function was successful
                    Data: Specifies the ADC timing mode, controlling the tradeoff
                          between speed and effective resolution. Some ADC timing
                          modes provide increased powerline noise rejection.

              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_setaiadctimd NI-DAQmx 8.5: **DAQmxSetAIADCTimingMode**

Purpose: Specifies the ADC Timing mode, controlling the tradeoff between speed and effective resolution

HTBasic Definition:

```
LONG Result=FNHx_setaiadctimd((LONG Taskid), (Channel$), (LONG Data))

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with
 Data              Specifies the ADC timing mode, controlling the tradeoff between speed
                   and effective resolution. Some ADC timing modes provide increased
                   powerline noise rejection

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_rstaiadctimd NI-DAQmx 8.5: **DAQmxResetAIADCTimingMode**

Purpose: Resets the ADC Timing mode, controlling the tradeoff between speed and effective resolution

HTBasic Definition:

```
LONG Result=FNHx_rstaiadctimd((LONG Taskid), (Channel$))

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with


Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_getdilogifam NI-DAQmx 8.3: **DAQmxGetDILogicFamily**

Purpose: Specifies the logic family to use for acquisition

HTBasic Definition:

```
LONG Result=FNHx_getdilogifam((LONG Taskid), (Channel$), LONG Data)

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with

Return value:   0 = Function was successful
                    Data: Specifies the logic family to use for acquisition. A logic
                          family corresponds to voltage thresholds that are compatible
                          with a group of voltage standards.

             <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_setdilogifam NI-DAQmx 8.3: **DAQmxSetDILogicFamily**

Purpose: Specifies the logic family to use for acquisition

HTBasic Definition:

```
LONG Result=FNHx_setdilogifam((LONG Taskid), (Channel$), (LONG Data))

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with
 Data              Specifies the logic family to use for acquisition. A logic family
                   corresponds to voltage thresholds that are compatible with a group of
                   voltage standards

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_rstdilogifam NI-DAQmx 8.3: **DAQmxResetDILogicFamily**

Purpose: Resets the logic family to use for acquisition

HTBasic Definition:

```
LONG Result=FNHx_rstdilogifam((LONG Taskid), (Channel$))

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with


Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdimmapena NI-DAQmx 8.3: **DAQmxGetDIMemMapEnable**

Purpose: Determines if NI-DAQmx maps hardware registers to the memory space of the customer process, if possible

HTBasic Definition:

```
LONG Result=FNHx_getdimmapena((LONG Taskid), (Channel$), LONG Data)

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with

Return value:   0 = Function was successful
                    Data: Specifies if NI-DAQmx maps hardware registers to the memory
                          space of the customer process, if possible.

              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_setdimmapena NI-DAQmx 8.3: **DAQmxSetDIMemMapEnable**

Purpose: Specifies if NI-DAQmx maps hardware registers to the memory space of the customer process, if possible

HTBasic Definition:

```
LONG Result=FNHx_setdimmapena((LONG Taskid), (Channel$), (LONG Data))

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with
 Data              Specifies if NI-DAQmx maps hardware registers to the memory space of
                   the customer process, if possible.

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_rstdimmapena NI-DAQmx 8.3: **DAQmxResetDIMemMapEnable**

Purpose: Resets NI-DAQmx hardware register mappings, if possible

HTBasic Definition:

```
LONG Result=FNHx_rstdimmapena((LONG Taskid), (Channel$))

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with


Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdiacquion NI-DAQmx 8.3: **DAQmxGetDIAcquireOn**

Purpose: Determines on which edge of the sample clock to acquire samples

HTBasic Definition:

```
LONG Result=FNHx_getdiacquion((LONG Taskid), (Channel$), LONG Data)

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with

Return value:   0 = Function was successful
                      Data: Specifies on which edge of the sample clock to acquire
                            samples.

              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_setdiacquion NI-DAQmx 8.3: **DAQmxSetDIAcquireOn**

Purpose: Specifies on which edge of the sample clock to acquire samples

HTBasic Definition:

```
LONG Result=FNHx_setdiacquion((LONG Taskid), (Channel$), (LONG Data))

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with
 Data              Specifies on which edge of the sample clock to acquire samples.

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_rstdiacquion NI-DAQmx 8.3: **DAQmxResetDIAcquireOn**

Purpose: Resets behaviour on which edge of the sample clock to acquire samples

HTBasic Definition:

```
LONG Result=FNHx_rstdiacquion((LONG Taskid), (Channel$))

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with


Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdolsstrts NI-DAQmx 8.3: **DAQmxGetDOLineStatesStartState**

Purpose: Determines the state of the lines in a digital output task when the task starts

HTBasic Definition:

```
LONG Result=FNHx_getdolsstrts((LONG Taskid), (Channel$), LONG Data)

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with

Return value:   0 = Function was successful
                    Data: Specifies the state of the lines in a digital output task
                          when the task starts.

              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_setdolsstrts NI-DAQmx 8.3: **DAQmxSetDOLineStatesStartState**

Purpose: Specifies the state of the lines in a digital output task when the task starts

HTBasic Definition:

```
LONG Result=FNHx_setdolsstrts((LONG Taskid), (Channel$), (LONG Data))

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with
 Data              Specifies the state of the lines in a digital output task when the
                   task starts.

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_rstdolsstrts NI-DAQmx 8.3: **DAQmxResetDOLineStatesStartState**

Purpose: Resets the state of the lines in a digital output task when the task starts to default value

HTBasic Definition:

```
LONG Result=FNHx_rstdolsstrts((LONG Taskid), (Channel$))

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with


Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdolspauss NI-DAQmx 8.3: **DAQmxGetDOLineStatesPausedState**

Purpose: Determines the state of the lines in a digital output task when the task pauses

HTBasic Definition:

```
LONG Result=FNHx_getdolspauss((LONG Taskid), (Channel$), LONG Data)

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with

Return value:   0 = Function was successful
                    Data: Specifies the state of the lines in a digital output task
                          when the task pauses.

              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_setdolspauss NI-DAQmx 8.3: **DAQmxSetDOLineStatesPausedState**

Purpose: Specifies the state of the lines in a digital output task when the task pauses

HTBasic Definition:

```
LONG Result=FNHx_setdolspauss((LONG Taskid), (Channel$), (LONG Data))

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with
 Data              Specifies the state of the lines in a digital output task when the
                   task pauses.

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_rstdolspauss NI-DAQmx 8.3: **DAQmxResetDOLineStatesPausedState**

Purpose: Resets the state of the lines in a digital output task when the task pauses to default value

HTBasic Definition:

```
LONG Result=FNHx_rstdolspauss((LONG Taskid), (Channel$))

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with


Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdolsdones NI-DAQmx 8.3: **DAQmxGetDOLineStatesDoneState**

Purpose: Determines the state of the lines in a digital output task when the task completes execution

HTBasic Definition:

```
LONG Result=FNHx_getdolsdones((LONG Taskid), (Channel$), LONG Data)

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with

Return value:   0 = Function was successful
                    Data: Specifies the state of the lines in a digital output task
                          when the task completes execution.
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_setdolsdones NI-DAQmx 8.3: **DAQmxSetDOLineStatesDoneState**

Purpose: Specifies the state of the lines in a digital output task when the task completes execution

HTBasic Definition:

```
LONG Result=FNHx_setdolsdones((LONG Taskid), (Channel$), (LONG Data))

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with
 Data              Specifies the state of the lines in a digital output task when the
                   task completes execution.

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_rstdolsdones NI-DAQmx 8.3: **DAQmxResetDOLineStatesDoneState**

Purpose: Resets the state of the lines in a digital output task when the task completes execution to default value

HTBasic Definition:

```
LONG Result=FNHx_rstdolsdones((LONG Taskid), (Channel$))

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_setdlfampust NI-DAQmx 8.3: **DAQmxSetDigitalLogicFamilyPowerUpState**

Purpose: Sets the digital logic family to use when the device powers up

HTBasic Definition:

```
LONG Result=FNHx_setdlfampust((Devicename$), (LONG Logicfamily))

Input:
 Devicename$       The name of the device to which this poperation applies
 Logicfamily       Specifies the logic family to set the device to when it powers up

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdologifam NI-DAQmx 8.3: **DAQmxGetDOLogicFamily**

Purpose: Determines the logic family to use for generation

HTBasic Definition:

```
LONG Result=FNHx_getdologifam((LONG Taskid), (Channel$), LONG Data)

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with

Return value:   0 = Function was successful
                    Data: Specifies the logic family to use for generation

              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_setdologifam NI-DAQmx 8.3: **DAQmxSetDOLogicFamily**

Purpose: Specifies the logic family to use for generation

HTBasic Definition:

```
LONG Result=FNHx_setdologifam((LONG Taskid), (Channel$), (LONG Data))

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with
 Data              Specifies the logic family to use for generation

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_rstdologifam NI-DAQmx 8.3: **DAQmxResetDOLogicFamily**

Purpose: Resets the logic family to use for generation to default value

HTBasic Definition:

```
LONG Result=FNHx_rstdologifam((LONG Taskid), (Channel$))

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with


Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdommapena NI-DAQmx 8.3: **DAQmxGetDOMemMapEnable**

Purpose: Determines if NI-DAQmx maps hardware registers to the memory space of the customer process, if possible

HTBasic Definition:

```
LONG Result=FNHx_getdommapena((LONG Taskid), (Channel$), LONG Data)

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with

Return value:   0 = Function was successful
                    Data: Specifies if NI-DAQmx maps hardware registers to the memory
                          space of the customer process, if possible

              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_setdommapena NI-DAQmx 8.3: **DAQmxSetDOMemMapEnable**

Purpose: Specifies if NI-DAQmx maps hardware registers to the memory space of the customer process, if possible

HTBasic Definition:

```
LONG Result=FNHx_setdommapena((LONG Taskid), (Channel$), (LONG Data))

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with
 Data              Specifies if NI-DAQmx maps hardware registers to the memory space of
                   the customer process, if possible

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_rstdommapena NI-DAQmx 8.3: **DAQmxResetDOMemMapEnable**

Purpose: Resets NI-DAQmx hardware registers mappings to default values

HTBasic Definition:

```
LONG Result=FNHx_rstdommapena((LONG Taskid), (Channel$))

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_getdogeneron NI-DAQmx 8.3: **DAQmxGetDOGenerateOn**

Purpose: Determines on which edge of the sample clock to generate samples

HTBasic Definition:

```
LONG Result=FNHx_getdogeneron((LONG Taskid), (Channel$), LONG Data)

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with

Return value:   0 = Function was successful
                    Data: Edge of the sample clock on which samples are generated

              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_setdogeneron NI-DAQmx 8.3: **DAQmxSetDOGenerateOn**

Purpose: Specifies on which edge of the sample clock to generate samples

HTBasic Definition:

```
LONG Result=FNHx_setdogeneron((LONG Taskid), (Channel$), (LONG Data))

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with
 Data              Specifies the edge of the sample clock on which samples are generated

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_rstdogeneron NI-DAQmx 8.3: **DAQmxResetDOGenerateOn**

Purpose: Resets behaviour on which edge of the sample clock to generate samples to default value

HTBasic Definition:

```
LONG Result=FNHx_rstdogeneron((LONG Taskid), (Channel$))

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getcocongmod NI-DAQmx 8.5: **DAQmxGetCOConstrainedGenMode**

Purpose: Determines constraints to apply when the counter generates pulses

HTBasic Definition:

```
LONG Result=FNHx_getcocongmod((LONG Taskid), (Channel$), LONG Data)

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with

Return value:   0 = Function was successful
                    Data: Constraints to apply when the counter generates pulses

              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_setcocongmod NI-DAQmx 8.5: **DAQmxSetCOConstrainedGenMode**

Purpose: Specifies constraints to apply when the counter generates pulses

HTBasic Definition:

```
LONG Result=FNHx_setcocongmod((LONG Taskid), (Channel$), (LONG Data))

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with
 Data              Specifies constraints to apply when the counter generates pulses

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_rstcocongmod NI-DAQmx 8.5: **DAQmxResetCOConstrainedGenMode**

Purpose: Resets the constraints to apply when the counter generates pulses

HTBasic Definition:

```
LONG Result=FNHx_rstcocongmod((LONG Taskid), (Channel$))

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_getexscdloff NI-DAQmx 8.5: **DAQmxGetExportedSampClkDelayOffset**

Purpose: Determines in seconds the amount of time to offset the exported Sample clock

HTBasic Definition:

```
LONG Result=FNHx_getexscdloff((LONG Taskid), REAL Data)

Input:
 Taskid            The task to which to add the channels that this function creates

Return value:   0 = Function was successful
                    Data: Returns in seconds the amount of time to offset the exported
                          sample clock

              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_setexscdloff NI-DAQmx 8.5: **DAQmxSetExportedSampClkDelayOffset**

Purpose: Specifies in seconds the amount of time to offset the exported Sample clock

HTBasic Definition:

```
LONG Result=FNHx_setexscdloff((LONG Taskid), (REAL Data))

Input:
 Taskid            The task to which to add the channels that this function creates
 Data              Specifies the amount of time to offset the exported Sample clock

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_rstexscdloff NI-DAQmx 8.5: **DAQmxResetExportedSampClkDelayOffset**

Purpose: Resets the amount of time to offset the exported Sample clock to default value

HTBasic Definition:

```
LONG Result=FNHx_rstexscdloff((LONG Taskid))

Input:
 Taskid            The task to which to add the channels that this function creates

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_setphyschnam NI-DAQmx 8.3: **DAQmxSetPhysicalChanName**

Purpose: Specifies the name of the physical channel upon which this virtual channel is based

HTBasic Definition:

```
LONG Result=FNHx_setphyschnnam((LONG Taskid), (Channel$), (Data$))

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with
 Data$             The name of the physical channel upon which this virtual channel
                   is based

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getexptouttm NI-DAQmx 8.3: **DAQmxGetExportedPauseTrigOutputTerm**

Purpose: Determines the terminal to which to route the Pause Trigger

HTBasic Definition:

```
LONG Result=FNHx_getexptouttm((LONG Taskid), Data$, (LONG Buffersize))

Input:
 Taskid            The task to which to add the channels that this function creates
 Buffersize        Size of Data$ string buffer

Return value:   0 = Function was successful
                    Data$: Terminal to which to route the Pause Trigger
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.35 and higher.

FNHx_setexptouttm NI-DAQmx 8.3: **DAQmxSetExportedPauseTrigOutputTerm**

Purpose: Specifies the terminal to which to route the Pause Trigger

HTBasic Definition:

```
LONG Result=FNHx_setexptouttm((LONG Taskid), (Data$))

Input:
 Taskid            The task to which to add the channels that this function creates
 Data$             Terminal to which the Pause Trigger is routed

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_rstexptouttm NI-DAQmx 8.3: **DAQmxResetExportedPauseTrigOutputTerm**

Purpose: Resets to the default terminal to which to route the Pause Trigger

HTBasic Definition:

```
LONG Result=FNHx_rstexptouttm((LONG Taskid))

Input:
 Taskid            The task to which to add the channels that this function creates

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getexptlaclv — NI-DAQmx 8.3: DAQmxGetExportedPauseTrigLvlActiveLvl

Purpose: Determines the active level of the exported Pause Trigger

HTBasic Definition:

```
LONG Result=FNHx_getexptlaclv((LONG Taskid), LONG Data)

Input:
 Taskid            The task to which to add the channels that this function creates

Return value:   0 = Function was successful
                    Data: Active level of the exported Pause Trigger

              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_setexptlaclv — NI-DAQmx 8.3: DAQmxSetExportedPauseTrigLvlActiveLvl

Purpose: Specifies the active level of the exported Pause Trigger

HTBasic Definition:

```
LONG Result=FNHx_setexptlaclv((LONG Taskid), (LONG Data))

Input:
 Taskid            The task to which to add the channels that this function creates
 Data              Specifies active level of the exported Pause Trigger

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_rstexptlaclv — NI-DAQmx 8.3: DAQmxResetExportedPauseTrigLvlActiveLvl

Purpose: Resets the active level of the exported Pause Trigger to it's default state

HTBasic Definition:

```
LONG Result=FNHx_rstexptlaclv((LONG Taskid))

Input:
 Taskid            The task to which to add the channels that this function creates
 Channel$          A subset of virtual channels in the task that you want to work with

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getexrftrppo NI-DAQmx 8.3: **DAQmxGetExportedRefTrigPulsePolarity**

Purpose: Determines the polarity of the exported Reference Trigger

HTBasic Definition:

```
LONG Result=FNHx_getexrftrppo((LONG Taskid), LONG Data)

Input:
 Taskid            The task to which to add the channels that this function creates

Return value:   0 = Function was successful
                    Data: The polarity of the exported Reference Trigger

              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_setexrftrppo NI-DAQmx 8.3: **DAQmxSetExportedRefTrigPulsePolarity**

Purpose: Specifies polarity of the exported Reference Trigger

HTBasic Definition:

```
LONG Result=FNHx_setexrftrppo((LONG Taskid), (LONG Data))

Input:
 Taskid            The task to which to add the channels that this function creates
 Data              Specifies the polarity of the exported Reference Trigger

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_rstexrftrppo NI-DAQmx 8.3: **DAQmxResetExportedRefTrigPulsePolarity**

Purpose: Resets the polarity of the exported Reference Trigger to it's default state

HTBasic Definition:

```
LONG Result=FNHx_rstexrftrppo((LONG Taskid))

Input:
 Taskid            The task to which to add the channels that this function creates

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getexsttrppo NI-DAQmx 8.3: **DAQmxGetExportedStartTrigPulsePolarity**

Purpose: Determines the polarity of the exported Start Trigger

HTBasic Definition:

```
LONG Result=FNHx_getexsttrppo((LONG Taskid), LONG Data)

Input:
 Taskid            The task to which to add the channels that this function creates

Return value:   0 = Function was successful
                    Data: The polarity of the exported Start Trigger

              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_setexsttrppo NI-DAQmx 8.3: **DAQmxSetExportedStartTrigPulsePolarity**

Purpose: Specifies polarity of the exported Start Trigger

HTBasic Definition:

```
LONG Result=FNHx_setexsttrppo((LONG Taskid), (LONG Data))

Input:
 Taskid            The task to which to add the channels that this function creates
 Data              Specifies the polarity of the exported Start Trigger

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_rstexsttrppo NI-DAQmx 8.3: **DAQmxResetExportedStartTrigPulsePolarity**

Purpose: Resets the polarity of the exported Start Trigger to it's default state

HTBasic Definition:

```
LONG Result=FNHx_rstexsttrppo((LONG Taskid))

Input:
 Taskid            The task to which to add the channels that this function creates

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getexrdfxedc — NI-DAQmx 8.3: DAQmxGetExportedRdyForXferEventDeassertCond

Purpose: Determines when the ready for transfer event deasserts

HTBasic Definition:

```
LONG Result=FNHx_getexrdfxedc((LONG Taskid), LONG Data)

Input:
 Taskid            The task to which to add the channels that this function creates

Return value:   0 = Function was successful
                    Data: Returns when the ready for transfer event deasserts

              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_setexrdfxedc — NI-DAQmx 8.3: DAQmxSetExportedRdyForXferEventDeassertCond

Purpose: Specifies when the ready for transfer event deasserts

HTBasic Definition:

```
LONG Result=FNHx_setexrdfxedc((LONG Taskid), (LONG Data))

Input:
 Taskid            The task to which to add the channels that this function creates
 Data              Specifies when the ready for transfer event deasserts

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_rstexrdfxedc — NI-DAQmx 8.3: DAQmxResetExportedRdyForXferEventDeassertCond

Purpose: Resets the ready for transfer event deassertion to it's default state

HTBasic Definition:

```
LONG Result=FNHx_rstexrdfxedc((LONG Taskid))

Input:
 Taskid            The task to which to add the channels that this function creates

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getexrdyfxed
NI-DAQmx 8.3: **DAQmxGetExportedRdyForXferEventDeassertCondCustomThreshold**

Purpose: Determines in samples the threshold below which the Ready for Transfer Event deasserts

HTBasic Definition:

```
LONG Result=FNHx_getexrdyfxed((LONG Taskid), LONG Data)

Input:
 Taskid            The task to which to add the channels that this function creates

Return value:   0 = Function was successful
                    Data: The treshhold below which the Ready Transition deasserts

              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_setexrdyfxed
NI-DAQmx 8.3: **DAQmxSetExportedRdyForXferEventDeassertCondCustomThreshold**

Purpose: Specifies in samples the threshold below which the Ready for Transfer Event deasserts

HTBasic Definition:

```
LONG Result=FNHx_setexrdyfxed((LONG Taskid), (LONG Data))

Input:
 Taskid            The task to which to add the channels that this function creates
 Data              Specifies the treshhold below which the Ready Transition deasserts

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_rstexrdyfxed
NI-DAQmx 8.3: **DAQmxResetExportedRdyForXferEventDeassertCondCustomThreshold**

Purpose: Resets the threshold below which the Ready for Transfer Event deasserts to it's default state

HTBasic Definition:

```
LONG Result=FNHx_rstexrdyfxed((LONG Taskid))

Input:
 Taskid            The task to which to add the channels that this function creates

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getexdaceout NI-DAQmx 8.3: **DAQmxGetExportedDataActiveEventOutputTerm**

Purpose: Determines the terminal to which to export the Data Active Event

HTBasic Definition:

```
LONG Result=FNHx_getexdaceout((LONG Taskid), Data$, (LONG Buffersize))

Input:
 Taskid            The task to which to add the channels that this function creates
 Buffersize        Size of Data$ string buffer

Return value:   0 = Function was successful
                    Data$: The terminal to which to export the Data Active Event
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_setexdaceout NI-DAQmx 8.3: **DAQmxSetExportedDataActiveEventOutputTerm**

Purpose: Specifies the terminal to which to export the Data Active Event

HTBasic Definition:

```
LONG Result=FNHx_setexdaceout((LONG Taskid), (Data$))

Input:
 Taskid            The task to which to add the channels that this function creates
 Data$             Terminal to which to export the Data Active Event

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_rstexdaceout NI-DAQmx 8.3: **DAQmxResetExportedDataActiveEventOutputTerm**

Purpose: Resets the terminal to which to export the Data Active Event to it's default state

HTBasic Definition:

```
LONG Result=FNHx_rstexdaceout((LONG Taskid))

Input:
 Taskid            The task to which to add the channels that this function creates

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getexdaevlal NI-DAQmx 8.3: **DAQmxGetExportedDataActiveEventLvlActiveLvl**

Purpose: Determines the polarity of the exported Data Active Event

HTBasic Definition:

```
LONG Result=FNHx_getexdaevlal((LONG Taskid), LONG Data)

Input:
 Taskid            The task to which to add the channels that this function creates

Return value:   0 = Function was successful
                    Data: The polarity of the exported Data Active Event

              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_setexdaevlal NI-DAQmx 8.3: **DAQmxSetExportedDataActiveEventLvlActiveLvl**

Purpose: Specifies the polarity of the exported Data Active Event

HTBasic Definition:

```
LONG Result=FNHx_setexdaevlal((LONG Taskid), (LONG Data))

Input:
 Taskid            The task to which to add the channels that this function creates
 Data              Specifies the polarity of the exported Data Active Event

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_rstexdaevlal NI-DAQmx 8.3: **DAQmxResetExportedDataActiveEventLvlActiveLvl**

Purpose: Resets the polarity of the exported Data Active Event to it's default state

HTBasic Definition:

```
LONG Result=FNHx_rstexdaevlal((LONG Taskid))

Input:
 Taskid            The task to which to add the channels that this function creates

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getexrfseotm NI-DAQmx 8.3: **DAQmxGetExportedRdyForStartEventOutputTerm**

Purpose: Determines the terminal to which to route the Ready for Start Event

HTBasic Definition:

```
LONG Result=FNHx_getexrfseotm((LONG Taskid), Data$, (LONG Buffersize))

Input:
 Taskid            The task to which to add the channels that this function creates
 Buffersize        Size of Data$ string buffer

Return value:   0 = Function was successful
                  Data$: The terminal to which to route the Ready for Start Event
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_setexrfseotm NI-DAQmx 8.3: **DAQmxSetExportedRdyForStartEventOutputTerm**

Purpose: Specifies the terminal to which to route the Ready for Start Event

HTBasic Definition:

```
LONG Result=FNHx_setexrfseotm((LONG Taskid), (Data$))

Input:
 Taskid            The task to which to add the channels that this function creates
 Data$             The terminal to which the Ready for Start Event is routed

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_rstexrfseotm NI-DAQmx 8.3: **DAQmxResetExportedRdyForStartEventOutputTerm**

Purpose: Resets the terminal to which to route the Ready for Start Event to it's default state

HTBasic Definition:

```
LONG Result=FNHx_rstexrfseotm((LONG Taskid))

Input:
 Taskid            The task to which to add the channels that this function creates

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getexrfselal NI-DAQmx 8.3: **DAQmxGetExportedRdyForStartEventLvlActiveLvl**

Purpose: Determines the polarity of the exported Ready for Start Event

HTBasic Definition:

```
LONG Result=FNHx_getexrfselal((LONG Taskid), LONG Data)

Input:
 Taskid            The task to which to add the channels that this function creates

Return value:   0 = Function was successful
                    Data: The polarity of the exported Ready for Start event

              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_setexrfselal NI-DAQmx 8.3: **DAQmxSetExportedRdyForStartEventLvlActiveLvl**

Purpose: Specifies the polarity of the exported Ready for Start Event

HTBasic Definition:

```
LONG Result=FNHx_setexrfselal((LONG Taskid), (LONG Data))

Input:
 Taskid            The task to which to add the channels that this function creates
 Data              Specifies the polarity of the exported Ready for Start event

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_rstexrfselal NI-DAQmx 8.3: **DAQmxResetExportedRdyForStartEventLvlActiveLvl**

Purpose: Resets the polarity of the exported Ready for Start Event to it's default state

HTBasic Definition:

```
LONG Result=FNHx_rstexrfselal((LONG Taskid))

Input:
 Taskid            The task to which to add the channels that this function creates

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note:
This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevprdcat NI-DAQmx 8.3: **DAQmxGetDevProductCategory**

Purpose: Indicates the product category of the device

HTBasic Definition:

```
LONG Result=FNHx_getdevprdcat((Device$), LONG Data)

Input:
 Device$           The device to be used for this function

Return value:   0 = Function was successful
                    Data: The product category of the device
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevchmdnm NI-DAQmx 8.3: **DAQmxGetDevChassisModuleDevNames**

Purpose: Indicates an array containing the names of the modules in the chassis

HTBasic Definition:

```
LONG Result=FNHx_getdevchmdnm((Device$), Data$, (LONG Buffersize))

Input:
 Device$           The device to be used for this function
 Buffersize        Size of Data$ string buffer

Return value:   0 = Function was successful
                    Data$: Names of the modules in the chassis
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevatrsup NI-DAQmx 8.3: **DAQmxGetDevAnlgTrigSupported**

Purpose: Indicates if the device supports analog triggering

HTBasic Definition:

```
LONG Result=FNHx_getdevatrsup((Device$), LONG Data)

Input:
 Device$           The device to be used for this function

Return value:   0 = Function was successful
                    Data: Indicates of the device supports analog triggering
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevdtrsup NI-DAQmx 8.3: **DAQmxGetDevDigTrigSupported**

Purpose: Indicates if the device supports digital triggering

HTBasic Definition:

```
LONG Result=FNHx_getdevdtrsup((Device$), LONG Data)

Input:
 Device$           The device to be used for this function

Return value:   0 = Function was successful
                    Data: Indicates of the device supports digital triggering
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevaimscr NI-DAQmx 8.3: **DAQmxGetDevAIMaxSingleChanRate**

Purpose: Indicates the maximum rate for an analog input task if the task contains only a single channel from this device

HTBasic Definition:

```
LONG Result=FNHx_getdevaimscr((Device$), REAL Data)

Input:
 Device$           The device to be used for this function

Return value:   0 = Function was successful
                    Data: Maximum rate for an analog input task
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevaimmcr NI-DAQmx 8.3: **DAQmxGetDevAIMaxMultiChanRate**

Purpose: Indicates the maximum rate for an analog input task if the task contains multiple channels from this device

HTBasic Definition:

```
LONG Result=FNHx_getdevaimmcr((Device$), REAL Data)

Input:
 Device$           The device to be used for this function

Return value:   0 = Function was successful
                    Data: Maximum rate for an analog input task
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevaimrat NI-DAQmx 8.3: **DAQmxGetDevAIMinRate**

Purpose: Indicates the minimum rate for an analog input task on this device

HTBasic Definition:

```
LONG Result=FNHx_getdevaimrat((Device$), REAL Data)

Input:
 Device$           The device to be used for this function

Return value:   0 = Function was successful
                    Data: Minimum rate for an analog input task
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevasssup NI-DAQmx 8.3: **DAQmxGetDevAISimultaneousSamplingSupported**

Purpose: Indicates if the device supports simultaneous sampling

HTBasic Definition:

```
LONG Result=FNHx_getdevasssup((Device$), LONG Data)

Input:
 Device$           The device to be used for this function

Return value:   0 = Function was successful
                    Data: Indicates if the device supports simultaneous sampling
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevaitrus — NI-DAQmx 8.3: **DAQmxGetDevAITrigUsage**

Purpose: Indicates the triggers supported by this device for an analog input task

HTBasic Definition:

```
LONG Result=FNHx_getdevaitrus((Device$), LONG Data)

Input:
 Device$           The device to be used for this function

Return value:   0 = Function was successful
                    Data: Indicates the triggers supported by this device for an
                          analog input task
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevavrang — NI-DAQmx 8.3: **DAQmxGetDevAIVoltageRngs**

Purpose: Indicates pairs of input voltage ranges supported by this device

HTBasic Definition:

```
LONG Result=FNHx_getdevavrang((Device$), REAL Data(*), (LONG Arraysize))

Input:
 Device$           The device to be used for this function
 Arraysize         Size of the Data array in samples

Return value:   0 = Function was successful
                    Data(*): Pairs of input voltage ranges (low / high value etc.)
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevaviedv — NI-DAQmx 8.3: **DAQmxGetDevAIVoltageIntExcitDiscreteVals**

Purpose: Indicates the set of discrete internal voltage excitation values supported by this device

HTBasic Definition:

```
LONG Result=FNHx_getdevaviedv((Device$), REAL Data(*), (LONG Arraysize))

Input:
 Device$           The device to be used for this function
 Arraysize         Size of the Data array in samples

Return value:   0 = Function was successful
                    Data(*): Set of discrete internal voltage excitation values
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevavierv — NI-DAQmx 8.3: **DAQmxGetDevAIVoltageIntExcitRangeVals**

Purpose: Indicates pairs of internal voltage excitation ranges supported by this device

HTBasic Definition:

```
LONG Result=FNHx_getdevavierv((Device$), REAL Data(*), (LONG Arraysize))

Input:
 Device$           The device to be used for this function
 Arraysize         Size of the Data array in samples

Return value:   0 = Function was successful
                    Data(*): Pairs of internal voltage excitation ranges
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevacurra — NI-DAQmx 8.3: **DAQmxGetDevAICurrentRngs**

Purpose: Indicates the pairs of current input ranges supported by this device

HTBasic Definition:

```
LONG Result=FNHx_getdevacurra((Device$), REAL Data(*), (LONG Arraysize))

Input:
 Device$           The device to be used for this function
 Arraysize         Size of the Data array in samples

Return value:   0 = Function was successful
                    Data(*): Pairs of current input ranges (low / high value etc.)
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevaciedv — NI-DAQmx 8.3: **DAQmxGetDevAICurrentIntExcitDiscreteVals**

Purpose: Indicates the set of discrete internal current excitation values supported by this device

HTBasic Definition:

```
LONG Result=FNHx_getdevaciedv((Device$), REAL Data(*), (LONG Arraysize))

Input:
 Device$           The device to be used for this function
 Arraysize         Size of the Data array in samples

Return value:   0 = Function was successful
                    Data(*): Set of discrete internal current excitation values
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevafrqrn — NI-DAQmx 8.3: **DAQmxGetDevAIFreqRngs**

Purpose: Indicates the pairs of frequency input ranges supported by this device

HTBasic Definition:

```
LONG Result=FNHx_getdevafrqrn((Device$), REAL Data(*), (LONG Arraysize))

Input:
 Device$           The device to be used for this function
 Arraysize         Size of the Data array in samples

Return value:   0 = Function was successful
                    Data(*): Pairs of frequency input ranges (low / high value etc.)
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevaigain — NI-DAQmx 8.3: **DAQmxGetDevAIGains**

Purpose: Indicates the input gain settings supported by this device

HTBasic Definition:

```
LONG Result=FNHx_getdevaigain((Device$), REAL Data(*), (LONG Arraysize))

Input:
 Device$           The device to be used for this function
 Arraysize         Size of the Data array in samples

Return value:   0 = Function was successful
                    Data(*): Input gain settings
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevaicoup NI-DAQmx 8.3: **DAQmxGetDevAICouplings**

Purpose: Indicates the coupling types supported by this device

HTBasic Definition:

```
LONG Result=FNHx_getdevaicoup((Device$), LONG Data)

Input:
 Device$           The device to be used for this function

Return value:   0 = Function was successful
                    Data: Indicates the coupling types supported by this device
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevalcfdv NI-DAQmx 8.3: **DAQmxGetDevAILowpassCutoffFreqDiscreteVals**

Purpose: Indicates the set of discrete lowpass cutoff frequencies supported by this device

HTBasic Definition:

```
LONG Result=FNHx_getdevalcfdv((Device$), REAL Data(*), (LONG Arraysize))

Input:
 Device$           The device to be used for this function
 Arraysize         Size of the Data array in samples

Return value:   0 = Function was successful
                    Data(*): Set of discrete lowpass cutoff frequencies
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevalcfrv NI-DAQmx 8.3: **DAQmxGetDevAILowpassCutoffFreqRangeVals**

Purpose: Indicates pairs of lowpass cutoff frequency ranges supported by this device

HTBasic Definition:

```
LONG Result=FNHx_getdevalcfrv((Device$), REAL Data(*), (LONG Arraysize))

Input:
 Device$           The device to be used for this function
 Arraysize         Size of the Data array in samples

Return value:   0 = Function was successful
                    Data(*): Pairs of lowpass cutoff frequency ranges
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevaophch NI-DAQmx 8.3: **DAQmxGetDevAOPhysicalChans**

Purpose: Indicates an array containing the names of the analog output physical channels available on the device

HTBasic Definition:

```
LONG Result=FNHx_getdevaophch((Device$), Data$, (LONG Buffersize))

Input:
 Device$           The device to be used for this function
 Buffersize        Size of Data$ string buffer

Return value:   0 = Function was successful
                    Data$: Array with names of analog output physical channels
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevaoscsp NI-DAQmx 8.3: **DAQmxGetDevAOSampClkSupported**

Purpose: Indicates if the device supports the sample clock timing type for analog output tasks

HTBasic Definition:

```
LONG Result=FNHx_getdevaoscsp((Device$), LONG Data)

Input:
 Device$           The device to be used for this function

Return value:   0 = Function was successful
                    Data: Indicates if the device supports the sample clock timing
                          type for analog output tasks
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevaomart NI-DAQmx 8.3: **DAQmxGetDevAOMaxRate**

Purpose: Indicates the maximum analog output rate of the device

HTBasic Definition:

```
LONG Result=FNHx_getdevaomart((Device$), REAL Data)

Input:
 Device$           The device to be used for this function

Return value:   0 = Function was successful
                    Data: Indicates the maximum analog output rate
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevaomirt NI-DAQmx 8.3: **DAQmxGetDevAOMinRate**

Purpose: Indicates the minimum analog output rate of the device

HTBasic Definition:

```
LONG Result=FNHx_getdevaomirt((Device$), REAL Data)

Input:
 Device$           The device to be used for this function

Return value:   0 = Function was successful
                    Data: Indicates the minimum analog output rate
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevaotrus NI-DAQmx 8.3: **DAQmxGetDevAOTrigUsage**

Purpose: Indicates the triggers supported by this device for analog output tasks

HTBasic Definition:

```
LONG Result=FNHx_getdevaotrus((Device$), LONG Data)

Input:
 Device$           The device to be used for this function

Return value:   0 = Function was successful
                    Data: Indicates the triggers supported by this device for analog
                          output tasks
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevaovorg NI-DAQmx 8.3: **DAQmxGetDevAOVoltageRngs**

Purpose: Indicates pairs of output voltage ranges supported by this device

HTBasic Definition:

```
LONG Result=FNHx_getdevaovorg((Device$), REAL Data(*), (LONG Arraysize))

Input:
 Device$           The device to be used for this function
 Arraysize         Size of the Data array in samples

Return value:   0 = Function was successful
                    Data(*): Pairs of output voltage ranges supported by this device
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevaocurg NI-DAQmx 8.3: **DAQmxGetDevAOCurrentRngs**

Purpose: Indicates pairs of output current ranges supported by this device

HTBasic Definition:

```
LONG Result=FNHx_getdevaocurg((Device$), REAL Data(*), (LONG Arraysize))

Input:
 Device$           The device to be used for this function
 Arraysize         Size of the Data array in samples

Return value:   0 = Function was successful
                    Data(*): Pairs of output current ranges supported by this device
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevaogain NI-DAQmx 8.3: **DAQmxGetDevAOGains**

Purpose: Indicates the output gain settings supported by this device

HTBasic Definition:

```
LONG Result=FNHx_getdevaogain((Device$), REAL Data(*), (LONG Arraysize))

Input:
 Device$           The device to be used for this function
 Arraysize         Size of the Data array in samples

Return value:   0 = Function was successful
                    Data(*): Output gain settings supported by this device
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevdimart NI-DAQmx 8.3: **DAQmxGetDevDIMaxRate**

Purpose: Indicates the maximum digital input rate of the device

HTBasic Definition:

```
LONG Result=FNHx_getdevdimart((Device$), REAL Data)

Input:
 Device$           The device to be used for this function

Return value:   0 = Function was successful
                    Data: Indicates the maximum digital input rate of the device
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevditrus — NI-DAQmx 8.3: **DAQmxGetDevDITrigUsage**

Purpose: Indicates the triggers supported by this device for digital input tasks

HTBasic Definition:

```
LONG Result=FNHx_getdevditrus((Device$), LONG Data)

Input:
 Device$           The device to be used for this function

Return value:   0 = Function was successful
                    Data: Indicates the triggers supported by this device for
                          digital input tasks
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevdomart — NI-DAQmx 8.3: **DAQmxGetDevDOMaxRate**

Purpose: Indicates the maximum digital output rate of the device

HTBasic Definition:

```
LONG Result=FNHx_getdevdomart((Device$), REAL Data)

Input:
 Device$           The device to be used for this function

Return value:   0 = Function was successful
                    Data: Indicates the maximum digital output rate of the device
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevdotrus — NI-DAQmx 8.3: **DAQmxGetDevDOTrigUsage**

Purpose: Indicates the triggers supported by this device for digital output tasks

HTBasic Definition:

```
LONG Result=FNHx_getdevdotrus((Device$), LONG Data)

Input:
 Device$           The device to be used for this function

Return value:   0 = Function was successful
                    Data: Indicates the triggers supported by this device for
                          digital output tasks
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevcitrus — NI-DAQmx 8.3: **DAQmxGetDevCITrigUsage**

Purpose: Indicates the triggers supported by this device for counter input tasks

HTBasic Definition:

```
LONG Result=FNHx_getdevcitrus((Device$), LONG Data)

Input:
 Device$           The device to be used for this function

Return value:   0 = Function was successful
                    Data: Indicates the triggers supported by this device for
                          counter input tasks
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevciscsp — NI-DAQmx 8.3: **DAQmxGetDevCISampClkSupported**

Purpose: Indicates if the device supports the sample clock timing type for counter input tasks

HTBasic Definition:

```
LONG Result=FNHx_getdevciscsp((Device$), LONG Data)

Input:
 Device$           The device to be used for this function

Return value:   0 = Function was successful
                    Data: Indicates if the device supports the sample clock timing
                          for counter input tasks
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevcimasz — NI-DAQmx 8.5: **DAQmxGetDevCIMaxSize**

Purpose: Indicates in bits the size of the counters on the device

HTBasic Definition:

```
LONG Result=FNHx_getdevcimasz((Device$), LONG Data)

Input:
 Device$           The device to be used for this function

Return value:   0 = Function was successful
                    Data: Indicates in bits the size of the counters on the device
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_getdevcimatb — NI-DAQmx 8.3: **DAQmxGetDevCIMaxTimebase**

Purpose: Indicates in Hertz the maximum counter timebase frequency

HTBasic Definition:

```
LONG Result=FNHx_getdevcimatb((Device$), REAL Data)

Input:
 Device$           The device to be used for this function

Return value:   0 = Function was successful
                    Data: Indicates in Hertz the maximum counter timebase frequency
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevcotrus — NI-DAQmx 8.3: **DAQmxGetDevCOTrigUsage**

Purpose: Indicates the triggers supported by this device for counter output tasks

HTBasic Definition:

```
LONG Result=FNHx_getdevcotrus((Device$), LONG Data)

Input:
 Device$           The device to be used for this function

Return value:   0 = Function was successful
                    Data: Indicates the triggers supported by this device for
                          counter output tasks
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevcomasz NI-DAQmx 8.5: **DAQmxGetDevCOMaxSize**

Purpose: Indicates the size of the counters on the device

HTBasic Definition:

```
LONG Result=FNHx_getdevcomasz((Device$), LONG Data)

Input:
 Device$           The device to be used for this function

Return value:   0 = Function was successful
                    Data: Indicates the size of the counters on the device
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_getdevcomatb NI-DAQmx 8.3: **DAQmxGetDevCOMaxTimebase**

Purpose: Indicates in Hertz the maximum counter timebase frequency

HTBasic Definition:

```
LONG Result=FNHx_getdevcomatb((Device$), REAL Data)

Input:
 Device$           The device to be used for this function

Return value:   0 = Function was successful
                    Data: Indicates in Hertz the maximum counter timebase frequency
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevnmdchn NI-DAQmx 8.3: **DAQmxGetDevNumDMAChans**

Purpose: Indicates the number of DMA channels on the device

HTBasic Definition:

```
LONG Result=FNHx_getdevnmdchn((Device$), LONG Data)

Input:
 Device$           The device to be used for this function

Return value:   0 = Function was successful
                    Data: Indicates the number of DMA channels on the device
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevcdcdnm NI-DAQmx 8.3: **DAQmxGetDevCompactDAQChassisDevName**

Purpose: Indicates the name of the CompactDAQ chassis that contains this module

HTBasic Definition:

```
LONG Result=FNHx_getdevcdcdnm((Device$), Data$, (LONG Buffersize))

Input:
 Device$           The device to be used for this function
 Buffersize        Size of Data$ string buffer

Return value:   0 = Function was successful
                    Data$: Name of the CompactDAQ chassis that contains this module
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdevcdslnm NI-DAQmx 8.3: **DAQmxGetDevCompactDAQSlotNum**

Purpose: Indicates the slot number in which this module is located in the CompactDAQ chassis

HTBasic Definition:

```
LONG Result=FNHx_getdevcdslnm((Device$), LONG Data)

Input:
 Device$           The device to be used for this function

Return value:   0 = Function was successful
                      Data: Indicates the slot number in which this module is located
                            in the CompactDAQ chassis
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getrdocuchex NI-DAQmx 8.5: **DAQmxGetReadOvercurrentChansExist**

Purpose: Indicates if the device(s) detected an overcurrent condition for any virtual channel in the task

HTBasic Definition:

```
LONG Result=FNHx_getrdocuchex((LONG Taskid), LONG Data)

Input:
 Taskid            The task to be used for this function

Return value:   0 = Function was successful
                      Data: Indicates if the device(s) detected an overcurrent condition
                            for any virtual channel in the task
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_getrdovcuchn NI-DAQmx 8.5: **DAQmxGetReadOvercurrentChans**

Purpose: Indicates the names of any virtual channels in the task for which an overcurrent condition has been detected

HTBasic Definition:

```
LONG Result=FNHx_getrdovcuchn((LONG Taskid), Data$, (LONG Buffersize))

Input:
 Taskid            The task to be used for this function
 Buffersize        Size of Data$ string buffer

Return value:   0 = Function was successful
                     Data$: Returns the names of any virtual channels in the task
                            for which an overcurrent condition has been detected
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_getrdoculcex NI-DAQmx 8.5: **DAQmxGetReadOpenCurrentLoopChansExist**

Purpose: Indicates if the device(s) detected an open current loop for any virtual channel in the task

HTBasic Definition:

```
LONG Result=FNHx_getrdoculcex((LONG Taskid), LONG Data)

Input:
 Taskid            The task to be used for this function

Return value:   0 = Function was successful
                      Data: Indicates if the device(s) detected an open current loop
                            for any virtual channel in the task
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_getrdopculch — NI-DAQmx 8.5: **DAQmxGetReadOpenCurrentLoopChans**

Purpose: Indicates the names of any virtual channels in the task for which the devices detected an open current loop

HTBasic Definition:

```
LONG Result=FNHx_getrdopculch((LONG Taskid), Data$, (LONG Buffersize))

Input:
 Taskid            The task to be used for this function
 Buffersize        Size of Data$ string buffer

Return value:   0 = Function was successful
                    Data$: Returns the names of any virtual channels in the task
                           for which the devices detected an open current loop
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_gettsknumdev — NI-DAQmx 8.3: **DAQmxGetTaskNumDevices**

Purpose: Indicates the number of devices in the task

HTBasic Definition:

```
LONG Result=FNHx_gettsknumdev((LONG Taskid), LONG Data)

Input:
 Taskid            The task to be used for this function

Return value:   0 = Function was successful
                    Data: Indicates the number of devices in the task
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getsmclufbeh — NI-DAQmx 8.3: **DAQmxGetSampClkUnderflowBehavior**

Purpose: Determines the action to take when the onboard memory of the device becomes empty

HTBasic Definition:

```
LONG Result=FNHx_getsmclufbeh((LONG Taskid), LONG Data)

Input:
 Taskid            The task to be used for this function

Return value:   0 = Function was successful
                    Data: Determines the action to take when the onboard memory
                          of the device becomes empty
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_setsmclufbeh — NI-DAQmx 8.3: **DAQmxSetSampClkUnderflowBehavior**

Purpose: Specifies the action to take when the onboard memory of the device becomes empty

HTBasic Definition:

```
LONG Result=FNHx_setsmclufbeh((LONG Taskid), (LONG Data))

Input:
 Taskid            The task to be used for this function

Return value:   0 = Function was successful
                    Data: Specifies the action to take when the onboard memory
                          of the device becomes empty
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_rstsmclufbeh NI-DAQmx 8.3: **DAQmxResetSampClkUnderflowBehavior**

Purpose: Resets the action to take when the onboard memory of the device becomes empty to it's default state

HTBasic Definition:

```
LONG Result=FNHx_rstsmclufbeh((LONG Taskid))

Input:
 Taskid            The task to be used for this function

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getaicvratex NI-DAQmx 8.3: **DAQmxGetAIConvRateEx**

Purpose: Determines the rate at which to clock the analog-to-digital converter

HTBasic Definition:

```
LONG Result=FNHx_getaicvratex((LONG Taskid), (Device$), REAL Data)

Input:
 Taskid            The task to be used for this function
 Device$           The device to be used for this function

Return value:   0 = Function was successful
                    Data: Determines the rate at which to clock the analog-to-digital
                          converter
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_setaicvratex NI-DAQmx 8.3: **DAQmxSetAIConvRateEx**

Purpose: Specifies the rate at which to clock the analog-to-digital converter

HTBasic Definition:

```
LONG Result=FNHx_setaicvratex((LONG Taskid), (Device$), (REAL Data))

Input:
 Taskid            The task to be used for this function
 Device$           The device to be used for this function
 Data              Specifies the rate at which to clock the analog-to-digital converter

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_rstaicvratex NI-DAQmx 8.3: **DAQmxResetAIConvRateEx**

Purpose: Resets the rate at which to clock the analog-to-digital converter to it's default state

HTBasic Definition:

```
LONG Result=FNHx_rstaicvratex((LONG Taskid), (Device$))

Input:
 Taskid            The task to be used for this function
 Device$           The device to be used for this function

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getaicvmrtex NI-DAQmx 8.3: **DAQmxGetAIConvMaxRateEx**

Purpose: Indicates the maximum convert rate supported by the task, given the current devices and channel count

HTBasic Definition:

```
LONG Result=FNHx_getaicvmrtex((LONG Taskid), (Device$), REAL Data)

Input:
 Taskid            The task to be used for this function
 Device$           The device to be used for this function

Return value:   0 = Function was successful
                    Data: Indicates the maximum convert rate supported by the task,
                          given the current devices and channel count
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getaicvsrcex NI-DAQmx 8.3: **DAQmxGetAIConvSrcEx**

Purpose: Determines the terminal of the signal to use as the AI Convert Clock

HTBasic Definition:

```
LONG Result=FNHx_getaicvsrcex((LONG Taskid), (Device$), Data$, (LONG Buffersize))

Input:
 Taskid            The task to be used for this function
 Device$           The device to be used for this function
 Buffersize        Size of Data$ string buffer

Return value:   0 = Function was successful
                    Data$: The terminal of the signal used as the AI Convert Clock
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_setaicvsrcex NI-DAQmx 8.3: **DAQmxSetAIConvSrcEx**

Purpose: Specifies the terminal of the signal to use as the AI Convert Clock

HTBasic Definition:

```
LONG Result=FNHx_setaicvsrcex((LONG Taskid), (Device$), (Data$))

Input:
 Taskid            The task to be used for this function
 Device$           The device to be used for this function
 Data$             The terminal of the signal used as the AI Convert Clock

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_rstaicvsrcex NI-DAQmx 8.3: **DAQmxResetAIConvSrcEx**

Purpose: Resets the terminal of the signal to use as the AI Convert Clock to it's default state

HTBasic Definition:

```
LONG Result=FNHx_rstaicvsrcex((LONG Taskid), (Device$))

Input:
 Taskid            The task to be used for this function
 Device$           The device to be used for this function

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getaicvaceex NI-DAQmx 8.3: **DAQmxGetAIConvActiveEdgeEx**

Purpose: Determines on which edge of the clock pulse an analog-to-digital conversion takes place

HTBasic Definition:

```
LONG Result=FNHx_getaicvaceex((LONG Taskid), (Device$), LONG Data)

Input:
 Taskid            The task to be used for this function
 Device$           The device to be used for this function

Return value:   0 = Function was successful
                    Data: The terminal of the signal used as the AI Convert Clock
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_setaicvaceex NI-DAQmx 8.3: **DAQmxSetAIConvActiveEdgeEx**

Purpose: Specifies on which edge of the clock pulse an analog-to-digital conversion takes place

HTBasic Definition:

```
LONG Result=FNHx_setaicvaceex((LONG Taskid), (Device$), (LONG Data))

Input:
 Taskid            The task to be used for this function
 Device$           The device to be used for this function
 Data              Specifies on which edge of the clock pulse an analog-to-digital
                   conversion takes place

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_rstaicvaceex NI-DAQmx 8.3: **DAQmxResetAIConvActiveEdgeEx**

Purpose: Resets on which edge of the clock pulse an analog-to-digital conversion takes place to it's default state

HTBasic Definition:

```
LONG Result=FNHx_rstaicvaceex((LONG Taskid), (Device$))

Input:
 Taskid            The task to be used for this function
 Device$           The device to be used for this function

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getaicvtbdex NI-DAQmx 8.3: **DAQmxGetAIConvTimebaseDivEx**

Purpose: Determines the number of AI Convert Clock Timebase pulses needed to produce a single AI Convert Clock pulse

HTBasic Definition:

```
LONG Result=FNHx_getaicvtbdex((LONG Taskid), (Device$), LONG Data)

Input:
 Taskid            The task to be used for this function
 Device$           The device to be used for this function

Return value:   0 = Function was successful
                    Data: Number of AI Convert Clock Timebase pulses needed to
                          produce a single AI Convert Clock pulse
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_setaicvtbdex — NI-DAQmx 8.3: DAQmxSetAIConvTimebaseDivEx

Purpose: Specifies the number of AI Convert Clock Timebase pulses needed to produce a single AI Convert Clock pulse

HTBasic Definition:

```
LONG Result=FNHx_setaicvtbdex((LONG Taskid), (Device$), (LONG Data))

Input:
 Taskid            The task to be used for this function
 Device$           The device to be used for this function
 Data              Specifies the number of AI Convert Clock Timebase pulses needed to
                   produce a single AI Convert Clock pulse

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_rstaicvtbdex — NI-DAQmx 8.3: DAQmxResetAIConvTimebaseDivEx

Purpose: Resets on which edge of the clock pulse an analog-to-digital conversion takes place to it's default state

HTBasic Definition:

```
LONG Result=FNHx_rstaicvtbdex((LONG Taskid), (Device$))

Input:
 Taskid            The task to be used for this function
 Device$           The device to be used for this function

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getaicvtbsex — NI-DAQmx 8.3: DAQmxGetAIConvTimebaseSrcEx

Purpose: Determines the terminal of the signal to use as the AI Convert Clock Timebase

HTBasic Definition:

```
LONG Result=FNHx_getaicvtbsex((LONG Taskid), (Device$), LONG Data)

Input:
 Taskid            The task to be used for this function
 Device$           The device to be used for this function

Return value:   0 = Function was successful
                    Data: Determines the terminal of the signal to use as the
                          AI Convert Clock Timebase
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_setaicvtbsex NI-DAQmx 8.3: **DAQmxSetAIConvTimebaseSrcEx**

Purpose: Specifies the terminal of the signal to use as the AI Convert Clock Timebase

HTBasic Definition:

```
LONG Result=FNHx_setaicvtbsex((LONG Taskid), (Device$), (LONG Data))

Input:
 Taskid            The task to be used for this function
 Device$           The device to be used for this function
 Data              Specifies the terminal of the signal to use as the AI Convert Clock
                   Timebase

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_rstaicvtbsex NI-DAQmx 8.3: **DAQmxResetAIConvTimebaseSrcEx**

Purpose: Resets the terminal of the signal to use as the AI Convert Clock Timebase to it's default state

HTBasic Definition:

```
LONG Result=FNHx_rstaicvtbsex((LONG Taskid), (Device$))

Input:
 Taskid            The task to be used for this function
 Device$           The device to be used for this function

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdlfscdunt NI-DAQmx 8.3: **DAQmxGetDelayFromSampClkDelayUnits**

Purpose: Determines the units of delay

HTBasic Definition:

```
LONG Result=FNHx_getdlfscdunt((LONG Taskid), LONG Data)

Input:
 Taskid            The task to be used for this function

Return value:   0 = Function was successful
                    Data: Determines the units of delay
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_setdlfscdunt NI-DAQmx 8.3: **DAQmxSetDelayFromSampClkDelayUnits**

Purpose: Specifies the units of delay

HTBasic Definition:

```
LONG Result=FNHx_setdlfscdunt((LONG Taskid), (LONG Data))

Input:
 Taskid            The task to be used for this function
 Data              Specifies the units of delay

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_rstdlfscdunt NI-DAQmx 8.3: **DAQmxResetDelayFromSampClkDelayUnits**

Purpose: Resets the units of delay to it's default state

HTBasic Definition:

```
LONG Result=FNHx_rstdlfscdunt((LONG Taskid))

Input:
 Taskid            The task to be used for this function

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdlfscduex NI-DAQmx 8.3: **DAQmxGetDelayFromSampClkDelayUnitsEx**

Purpose: Determines the units of delay

HTBasic Definition:

```
LONG Result=FNHx_getdlfscduex((LONG Taskid), (Device$), LONG Data)

Input:
 Taskid            The task to be used for this function
 Device$           The device to be used for this function

Return value:   0 = Function was successful
                    Data: Determines the units of delay
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_setdlfscduex NI-DAQmx 8.3: **DAQmxSetDelayFromSampClkDelayUnitsEx**

Purpose: Specifies the units of delay

HTBasic Definition:

```
LONG Result=FNHx_setdlfscduex((LONG Taskid), (Device$), (LONG Data))

Input:
 Taskid            The task to be used for this function
 Device$           The device to be used for this function
 Data              Specifies the units of delay

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_rstdlfscduex NI-DAQmx 8.3: **DAQmxResetDelayFromSampClkDelayUnitsEx**

Purpose: Resets the units of delay to it's default state

HTBasic Definition:

```
LONG Result=FNHx_rstdlfscduex((LONG Taskid), (Device$))

Input:
 Taskid            The task to be used for this function
 Device$           The device to be used for this function

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdlfscdlay NI-DAQmx 8.3: **DAQmxGetDelayFromSampClkDelay**

Purpose: Get the amount of time to wait after receiving a Sample Clock edge before beginning to acquire the sample

HTBasic Definition:

```
LONG Result=FNHx_getdlfscdlay((LONG Taskid), REAL Data)

Input:
 Taskid            The task to be used for this function

Return value:   0 = Function was successful
                    Data: Amount of time to wait after receiving a Sample Clock edge
                          before beginning to acquire the sample
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_setdlfscdlay NI-DAQmx 8.3: **DAQmxSetDelayFromSampClkDelay**

Purpose: Set the amount of time to wait after receiving a Sample Clock edge before beginning to acquire the sample

HTBasic Definition:

```
LONG Result=FNHx_getdlfscdlay((LONG Taskid), (REAL Data))

Input:
 Taskid            The task to be used for this function
 Data              Amount of time to wait after receiving a Sample Clock edge before
                   beginning to acquire the sample

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_rstdlfscdlay NI-DAQmx 8.3: **DAQmxResetDelayFromSampClkDelay**

Purpose: Resets the units of delay to it's default state

HTBasic Definition:

```
LONG Result=FNHx_rstdlfscdlay((LONG Taskid))

Input:
 Taskid            The task to be used for this function

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdlfscdlex NI-DAQmx 8.3: **DAQmxGetDelayFromSampClkDelayEx**

Purpose: Gets the amount of time to wait after receiving a Sample Clock edge before beginning to acquire the sample

HTBasic Definition:

```
LONG Result=FNHx_getdlfscdlex((LONG Taskid), (Device$), REAL Data)

Input:
 Taskid            The task to be used for this function
 Device$           The device to be used for this function

Return value:   0 = Function was successful
                    Data: Determines the amount of time to wait after receiving a
                          Sample Clock edge before beginning to aquire the sample
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_setdlfscdlex NI-DAQmx 8.3: **DAQmxSetDelayFromSampClkDelayEx**

Purpose: Sets the amount of time to wait after receiving a Sample Clock edge before beginning to acquire the sample

HTBasic Definition:

```
LONG Result=FNHx_setdlfscdlex((LONG Taskid), (Device$), (REAL Data))

Input:
 Taskid            The task to be used for this function
 Device$           The device to be used for this function
 Data              Specifies the amount of time to wait after receiving a Sample Clock
                   edge before beginning to aquire the sample

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_rstdlfscdlex NI-DAQmx 8.3: **DAQmxResetDelayFromSampClkDelayEx**

Purpose: Resets the amount of time to wait after receiving a Sample Clock edge before beginning to acquire the sample to it's default state

HTBasic Definition:

```
LONG Result=FNHx_rstdlfscdlex((LONG Taskid), (Device$))

Input:
 Taskid            The task to be used for this function
 Device$           The device to be used for this function

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdigpptsrc NI-DAQmx 8.3: **DAQmxGetDigPatternPauseTrigSrc**

Purpose: Determines the physical channels to use for pattern matching

HTBasic Definition:

```
LONG Result=FNHx_getdigpptsrc((LONG Taskid), Data$, (LONG Buffersize))

Input:
 Taskid            The task to be used for this function
 Buffersize        Size of Data$ string buffer

Return value:   0 = Function was successful
                    Data$: The physical channels to use for pattern matching
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_setdigpptsrc NI-DAQmx 8.3: **DAQmxSetDigPatternPauseTrigSrc**

Purpose: Specifies the physical channels to use for pattern matching

HTBasic Definition:

```
LONG Result=FNHx_setdigpptsrc((LONG Taskid), (Data$))

Input:
 Taskid            The task to be used for this function
 Data$             The physical channels to use for pattern matching

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_rstdigpptsrc NI-DAQmx 8.3: **DAQmxResetDigPatternPauseTrigSrc**

Purpose: Resets the physical channels to use for pattern matching to it's default state

HTBasic Definition:

```
LONG Result=FNHx_rstdigpptsrc((LONG Taskid))

Input:
 Taskid            The task to be used for this function

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdigpptpat NI-DAQmx 8.3: **DAQmxGetDigPatternPauseTrigPattern**

Purpose: Determines the digital pattern that must be met for the Pause Trigger to occur

HTBasic Definition:

```
LONG Result=FNHx_getdigpptpat((LONG Taskid), Data$, (LONG Buffersize))

Input:
 Taskid            The task to be used for this function
 Buffersize        Size of Data$ string buffer

Return value:   0 = Function was successful
                    Data$: The Digital pattern that must be met for the Pause Trigger
                           to occur
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_setdigpptpat NI-DAQmx 8.3: **DAQmxSetDigPatternPauseTrigPattern**

Purpose: Specifies the digital pattern that must be met for the Pause Trigger to occur

HTBasic Definition:

```
LONG Result=FNHx_setdigpptpat((LONG Taskid), (Data$))

Input:
 Taskid            The task to be used for this function
 Data$             The Digital pattern that must be met for the Pause Trigger to occur

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_rstdigpptpat NI-DAQmx 8.3: **DAQmxResetDigPatternPauseTrigPattern**

Purpose: Resets the digital pattern that must be met for the Pause Trigger to occur to it's default state

HTBasic Definition:

```
LONG Result=FNHx_rstdigpptpat((LONG Taskid))

Input:
 Taskid            The task to be used for this function

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getdigpptwhn NI-DAQmx 8.3: **DAQmxGetDigPatternPauseTrigWhen**

Purpose: Determines if the Pause Trigger occurs when the physical channels specified with Source match or differ from the digital pattern specified with Pattern

HTBasic Definition:

```
LONG Result=FNHx_getdigpptwhn((LONG Taskid), LONG Data)

Input:
 Taskid            The task to be used for this function

Return value:   0 = Function was successful
                    Data: Determines if the Pause Trigger occurs when the physical
                          channels specified with Source match or differ from the
                          digital pattern specified with Pattern
                          to occur
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_setdigpptwhn NI-DAQmx 8.3: **DAQmxSetDigPatternPauseTrigWhen**

Purpose: Specifies if the Pause Trigger occurs when the physical channels specified with Source match or differ from the digital pattern specified with Pattern

HTBasic Definition:

```
LONG Result=FNHx_setdigpptwhn((LONG Taskid), (LONG Data))

Input:
 Taskid            The task to be used for this function
 Data              Specifies if the Pause Trigger occurs when the physical channels
                   specified with Source match or differ from the digital pattern
                   specified with Pattern

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_rstdigpptwhn NI-DAQmx 8.3: **DAQmxResetDigPatternPauseTrigWhen**

Purpose: Resets if the Pause Trigger occurs when the physical channels specified with Source match or differ from the digital pattern specified with Pattern to occur to it's default state

HTBasic Definition:

```
LONG Result=FNHx_rstdigpptwhn((LONG Taskid))

Input:
 Taskid            The task to be used for this function

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getwrovcchex NI-DAQmx 8.5: **DAQmxGetWriteOvercurrentChansExist**

Purpose: Determines if the device(s) detected an overcurrent condition for any channel in the task

HTBasic Definition:

```
LONG Result=FNHx_getwrovcchex((LONG Taskid), LONG Data)

Input:
 Taskid            The task to be used for this function

Return value:   0 = Function was successful
                    Data: Determines if the device(s) detected an overcurrent
                          condition for any channel in the task
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_getwrovcchns NI-DAQmx 8.5: **DAQmxGetWriteOvercurrentChans**

Purpose: Determines the names of any virtual channels in the task for which an overcurrent condition has been detected

HTBasic Definition:

```
LONG Result=FNHx_getwrovcchns((LONG Taskid), Data$, (LONG Buffersize))

Input:
 Taskid            The task to be used for this function
 Buffersize        Size of Data$ string buffer

Return value:   0 = Function was successful
                    Data$: The names of any virtual channels in the task for which an
                           overcurrent condition has been detected
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_getwropcchex NI-DAQmx 8.5: **DAQmxGetWriteOpenCurrentLoopChansExist**

Purpose: Determines if the device(s) detected an open current loop for any channel in the task

HTBasic Definition:

```
LONG Result=FNHx_getwropcchex((LONG Taskid), LONG Data)

Input:
 Taskid            The task to be used for this function

Return value:   0 = Function was successful
                    Data: Determines if the device(s) detected an open current loop
                          for any channel in the task
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_getwropcchns NI-DAQmx 8.5: **DAQmxGetWriteOpenCurrentLoopChans**

Purpose: Determines the names of any virtual channels in the task for which an open current loop has been detected

HTBasic Definition:

```
LONG Result=FNHx_getwropcchns((LONG Taskid), Data$, (LONG Buffersize))

Input:
 Taskid            The task to be used for this function
 Buffersize        Size of Data$ string buffer

Return value:   0 = Function was successful
                    Data$: The names of any virtual channels in the task for which an
                           open current loop has been detected
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_getwrpsfchex NI-DAQmx 8.5: **DAQmxGetWritePowerSupplyFaultChansExist**

Purpose: Determines if the device(s) detected a power supply fault for any channel in the task

HTBasic Definition:

```
LONG Result=FNHx_getwrpsfchex((LONG Taskid), LONG Data)

Input:
 Taskid            The task to be used for this function

Return value:   0 = Function was successful
                    Data: Determines if the device(s) detected a power supply fault
                          for any channel in the task
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_getwrpsfchns NI-DAQmx 8.5: **DAQmxGetWritePowerSupplyFaultChans**

Purpose: Determines the names of any virtual channels in the task that have a power supply fault

HTBasic Definition:

```
LONG Result=FNHx_getwrpsfchns((LONG Taskid), Data$, (LONG Buffersize))

Input:
 Taskid            The task to be used for this function
 Buffersize        Size of Data$ string buffer

Return value:   0 = Function was successful
                    Data$: The names of any virtual channels in the task that have
                           a power supply fault
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_getwrnxwrila NI-DAQmx 8.3: **DAQmxGetWriteNextWriteIsLast**

Purpose: Determines that the next samples written are the last samples you want to generate

HTBasic Definition:

```
LONG Result=FNHx_getwrnxwrila((LONG Taskid), LONG Data)

Input:
 Taskid            The task to be used for this function

Return value:   0 = Function was successful
                    Data: Determines that the next samples written are the last
                          samples you want to generate
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_setwrnxwrila NI-DAQmx 8.3: **DAQmxSetWriteNextWriteIsLast**

Purpose: Specifies that the next samples written are the last samples you want to generate

HTBasic Definition:

```
LONG Result=FNHx_setwrnxwrila((LONG Taskid), (LONG Data))

Input:
 Taskid            The task to be used for this function

Return value:   0 = Function was successful
                    Data: Specifies that the next samples written are the last
                          samples you want to generate
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_rstwrnxwrila NI-DAQmx 8.3: **DAQmxResetWriteNextWriteIsLast**

Purpose: Resets the behaviour of this function to it's default state

HTBasic Definition:

```
LONG Result=FNHx_setwrnxwrila((LONG Taskid))

Input:
 Taskid            The task to be used for this function

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getphchaitcf NI-DAQmx 8.3: **DAQmxGetPhysicalChanAITermCfgs**

Purpose: Indicates the list of terminal configurations supported by the channel

HTBasic Definition:

```
LONG Result=FNHx_getphchaitcf((Physchan$), LONG Data)

Input:
 Physchan$         The device to be used for this function

Return value:   0 = Function was successful
                    Data: List of terminal configurations by the channel
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getphchaotcf NI-DAQmx 8.3: **DAQmxGetPhysicalChanAOTermCfgs**

Purpose: Indicates the list of terminal configurations supported by the channel

HTBasic Definition:

```
LONG Result=FNHx_getphchaotcf((Physchan$), LONG Data)

Input:
 Physchan$         The device to be used for this function

Return value:   0 = Function was successful
                    Data: List of terminal configurations by the channel
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getphchdipwd NI-DAQmx 8.3: **DAQmxGetPhysicalChanDIPortWidth**

Purpose: Indicates in bits the width of digital input port

HTBasic Definition:

```
LONG Result=FNHx_getphchdipwd((Physchan$), LONG Data)

Input:
 Physchan$         The device to be used for this function

Return value:   0 = Function was successful
                    Data: The bits width of digital input port
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getphchdiscs NI-DAQmx 8.3: **DAQmxGetPhysicalChanDISampClkSupported**

Purpose: Indicates if the sample clock timing type is supported for the digital input physical channel

HTBasic Definition:

```
LONG Result=FNHx_getphchdiscs((Physchan$), LONG Data)

Input:
 Physchan$          The device to be used for this function

Return value:    0 = Function was successful
                     Data: Indicates if the sample clock timing type is supported for
                           the digital input physical channel
                 <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getphchdidsp NI-DAQmx 8.3: **DAQmxGetPhysicalChanDIChangeDetectSupported**

Purpose: Indicates if the change detection timing type is supported for the digital input physical channel

HTBasic Definition:

```
LONG Result=FNHx_getphchdidsp((Physchan$), LONG Data)

Input:
 Physchan$          The device to be used for this function

Return value:    0 = Function was successful
                     Data: Indicates if the change detection timing type is supported
                           for the digital input physical channel
                 <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getphchdopwd NI-DAQmx 8.3: **DAQmxGetPhysicalChanDOPortWidth**

Purpose: Indicates in bits the width of digital output port

HTBasic Definition:

```
LONG Result=FNHx_getphchdopwd((Physchan$), LONG Data)

Input:
 Physchan$          The device to be used for this function

Return value:    0 = Function was successful
                     Data: The bits width of digital output port
                 <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getphchdoscs NI-DAQmx 8.3: **DAQmxGetPhysicalChanDOSampClkSupported**

Purpose: Indicates if the sample clock timing type is supported for the digital output physical channel

HTBasic Definition:

```
LONG Result=FNHx_getphchdoscs((Physchan$), LONG Data)

Input:
 Physchan$          The device to be used for this function

Return value:    0 = Function was successful
                     Data: Indicates if the sample clock timing type is supported for
                           the digital output physical channel
                 <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_getsctresmod NI-DAQmx 8.3: **DAQmxGetSampClkTimingResponseMode**

Purpose: Indicates the terminal of the signal to use as the Sample Clock Timebase

HTBasic Definition:

```
LONG Result=FNHx_getsctresmod((LONG Taskid), LONG Data)

Input:
 Taskid            The task to be used for this function

Return value:   0 = Function was successful
                    Data: Terminal of the signal to use as the Sample Clock Timebase
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_setsctresmod NI-DAQmx 8.3: **DAQmxSetSampClkTimingResponseMode**

Purpose: Specifies the terminal of the signal to use as the Sample Clock Timebase

HTBasic Definition:

```
LONG Result=FNHx_getsctresmod((LONG Taskid), (LONG Data))

Input:
 Taskid            The task to be used for this function
 Data              Terminal of the signal to use as the Sample Clock Timebase

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_rstsctresmod NI-DAQmx 8.3: **DAQmxResetSampClkTimingResponseMode**

Purpose: Resets the terminal of the signal to use as the Sample Clock Timebase to it's default state

HTBasic Definition:

```
LONG Result=FNHx_rstsctresmod((LONG Taskid))

Input:
 Taskid            The task to be used for this function

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.3 and higher.

FNHx_setsctresmod NI-DAQmx 8.5: **DAQmxSetSampClkTimingResponseMode**

Purpose: Specifies the terminal of the signal to use as the Sample Clock Timebase

HTBasic Definition:

```
LONG Result=FNHx_getsctresmod((LONG Taskid), (LONG Data))

Input:
 Taskid            The task to be used for this function
 Data              Terminal of the signal to use as the Sample Clock Timebase

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_rstsctresmod NI-DAQmx 8.5: **DAQmxResetSampClkTimingResponseMode**

Purpose: Resets the terminal of the signal to use as the Sample Clock Timebase to it's default state

HTBasic Definition:

```
LONG Result=FNHx_rstsctresmod((LONG Taskid))

Input:
 Taskid             The task to be used for this function

Return value:    0 = Function was successful
                <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.5 and higher.

FNHx_getaivoacrmu NI-DAQmx 8.6: **DAQmxGetAIVoltageACRMSUnits**

Purpose: Determines the units to use to return voltage RMS measurements from the channel

HTBasic Definition:

```
LONG Result=FNHx_getaivoacrmu((LONG Taskid), (Channel$), LONG Data)

Input:
 Taskid             The task used in this function
 Channel$           The name of the channel you want to work with

Return value:    0 = Function was successful
                     Data    Determines the units for the channel
                <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_setaivoacrmu NI-DAQmx 8.6: **DAQmxSetAIVoltageACRMSUnits**

Purpose: Specifies the units to use to return voltage RMS measurements from the channel

HTBasic Definition:

```
LONG Result=FNHx_setaivoacrmu((LONG Taskid), (Channel$), (LONG Data))

Input:
 Taskid             The task used in this function
 Channel$           The name of the channel you want to set
 Data               Specifies the units for the channel

Return value:    0 = Function was successful
                <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_rstaivoacrmu NI-DAQmx 8.6: **DAQmxResetAIVoltageACRMSUnits**

Purpose: Resets the units to use to return voltage RMS measurements from the channel to default values

HTBasic Definition:

```
LONG Result=FNHx_rstaivoacrmu((LONG Taskid), (Channel$))

Input:
 Taskid             The task used in this function
 Channel$           The name of the channel you want to reset

Return value:    0 = Function was successful
                <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_getaicuacrmu — NI-DAQmx 8.6: DAQmxGetAICurrentACRMSUnits

Purpose: Determines the units to use to return current RMS measurements from the channel

HTBasic Definition:

```
LONG Result=FNHx_getaicuacrmu((LONG Taskid), (Channel$), LONG Data)

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to work with

Return value:   0 = Function was successful
                    Data    Determines the units for the channel
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_setaicuacrmu — NI-DAQmx 8.6: DAQmxSetAICurrentACRMSUnits

Purpose: Specifies the units to use to return current RMS measurements from the channel

HTBasic Definition:

```
LONG Result=FNHx_setaicuacrmu((LONG Taskid), (Channel$), (LONG Data))

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to set
 Data              Specifies the units for the channel

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_rstaicuacrmu — NI-DAQmx 8.6: DAQmxResetAICurrentACRMSUnits

Purpose: Resets the units to use to return current RMS measurements from the channel to default values

HTBasic Definition:

```
LONG Result=FNHx_rstaicuacrmu((LONG Taskid), (Channel$))

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to reset

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_getaoculimit NI-DAQmx 8.6: **DAQmxGetAOVoltageCurrentLimit**

Purpose: Determines the current limit, in amperes, for the voltage channel

HTBasic Definition:

```
LONG Result=FNHx_getaoculimit((LONG Taskid), (Channel$), REAL Data)

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to work with

Return value:   0 = Function was successful
                    Data    Determines the current limit in Ampere for the channel
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_setaoculimit NI-DAQmx 8.6: **DAQmxSetAOVoltageCurrentLimit**

Purpose: Specifies the current limit, in amperes, for the voltage channel

HTBasic Definition:

```
LONG Result=FNHx_setaoculimit((LONG Taskid), (Channel$), (REAL Data))

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to set
 Data              Specifies the current limit in Ampere for the channel

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_rstaoculimit NI-DAQmx 8.6: **DAQmxResetAOVoltageCurrentLimit**

Purpose: Resets the current limit, in amperes, for the voltage channel, to it's default value

HTBasic Definition:

```
LONG Result=FNHx_rstaoculimit((LONG Taskid), (Channel$))

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to reset

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_getaofungtyp NI-DAQmx 8.6: **DAQmxGetAOFuncGenType**

Purpose: Determines the kind of the waveform to generate

HTBasic Definition:

```
LONG Result=FNHx_getaofungtyp((LONG Taskid), (Channel$), LONG Data)

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to work with

Return value:   0 = Function was successful
                    Data    Determines the kind of waveform to generate
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_setaofungtyp NI-DAQmx 8.6: **DAQmxSetAOFuncGenType**

Purpose: Specifies the kind of the waveform to generate

HTBasic Definition:

```
LONG Result=FNHx_setaofungtyp((LONG Taskid), (Channel$), (LONG Data))

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to set
 Data              Specifies the kind of waveform to generate

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_rstaofungtyp NI-DAQmx 8.6: **DAQmxResetAOFuncGenType**

Purpose: Resets the kind of the waveform to generate to it's default state

HTBasic Definition:

```
LONG Result=FNHx_rstaofungtyp((LONG Taskid), (Channel$))

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to reset

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_getaofungfrq NI-DAQmx 8.6: **DAQmxGetAOFuncGenFreq**

Purpose: Determines the frequency of the waveform to generate in hertz

HTBasic Definition:

```
LONG Result=FNHx_getaofungfrq((LONG Taskid), (Channel$), REAL Data)

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to work with

Return value:   0 = Function was successful
                    Data    Determines the frequency of the waveform to generate
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_setaofungfrq NI-DAQmx 8.6: **DAQmxSetAOFuncGenFreq**

Purpose: Specifies the frequency of the waveform to generate in hertz

HTBasic Definition:

```
LONG Result=FNHx_setaofungfrq((LONG Taskid), (Channel$), (REAL Data))

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to set
 Data              Specifies the frequency of the waveform to generate

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_rstaofungfrq NI-DAQmx 8.6: **DAQmxResetAOFuncGenFreq**

Purpose: Resets the frequency of the waveform to generate to it's default value

HTBasic Definition:

```
LONG Result=FNHx_rstaofungfrq((LONG Taskid), (Channel$))

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to reset

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_getaofungamp NI-DAQmx 8.6: **DAQmxGetAOFuncGenAmplitude**

Purpose: Determines the zero-to-peak amplitude of the waveform to generate in volts

HTBasic Definition:

```
LONG Result=FNHx_getaofungamp((LONG Taskid), (Channel$), REAL Data)

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to work with

Return value:   0 = Function was successful
                    Data    Determines the zero-to-peak amplitude of the waveform to
                            generate in Volt
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_setaofungamp NI-DAQmx 8.6: **DAQmxSetAOFuncGenAmplitude**

Purpose: Specifies the zero-to-peak amplitude of the waveform to generate in volts

HTBasic Definition:

```
LONG Result=FNHx_setaofungamp((LONG Taskid), (Channel$), (REAL Data))

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to set
 Data              Specifies the zero-to-peak amplitude of the waveform to generate
                   in Volt

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_rstaofungamp NI-DAQmx 8.6: **DAQmxResetAOFuncGenAmplitude**

Purpose: Resets the zero-to-peak amplitude of the waveform to generate to it's default value

HTBasic Definition:

```
LONG Result=FNHx_rstaofungamp((LONG Taskid), (Channel$))

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to reset

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_getaofungoff NI-DAQmx 8.6: **DAQmxGetAOFuncGenOffset**

Purpose: Determines the voltage offset of the waveform to generate

HTBasic Definition:

```
LONG Result=FNHx_getaofungoff((LONG Taskid), (Channel$), REAL Data)

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to work with

Return value:   0 = Function was successful
                    Data    Determines the voltage offset of the waveform to generate
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_setaofungoff NI-DAQmx 8.6: **DAQmxSetAOFuncGenOffset**

Purpose: Specifies the voltage offset of the waveform to generate

HTBasic Definition:

```
LONG Result=FNHx_setaofungoff((LONG Taskid), (Channel$), (REAL Data))

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to set
 Data              Specifies the voltage offset of the waveform to generate

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_rstaofungoff NI-DAQmx 8.6: **DAQmxResetAOFuncGenOffset**

Purpose: Resets the voltage offset of the waveform to generate to it's default value

HTBasic Definition:

```
LONG Result=FNHx_rstaofungoff((LONG Taskid), (Channel$))

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to reset

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_getaofungsdc NI-DAQmx 8.6: **DAQmxGetAOFuncGenSquareDutyCycle**

Purpose: Determines the square wave duty cycle of the waveform to generate

HTBasic Definition:

```
LONG Result=FNHx_getaofungsdc((LONG Taskid), (Channel$), REAL Data)

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to work with

Return value:   0 = Function was successful
                    Data    Determines the square wave duty cycle of the
                            waveform to generate
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_setaofungsdc NI-DAQmx 8.6: **DAQmxSetAOFuncGenSquareDutyCycle**

Purpose: Specifies the square wave duty cycle of the waveform to generate

HTBasic Definition:

```
LONG Result=FNHx_setaofungsdc((LONG Taskid), (Channel$), (REAL Data))

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to set
 Data              Specifies the square wave duty cycle of the waveform to generate

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_rstaofungsdc NI-DAQmx 8.6: **DAQmxResetAOFuncGenSquareDutyCycle**

Purpose: Resets the square wave duty cycle of the waveform to generate to it's default value

HTBasic Definition:

```
LONG Result=FNHx_rstaofungsdc((LONG Taskid), (Channel$))

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to reset

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_getaofungmtp NI-DAQmx 8.6: **DAQmxGetAOFuncGenModulationType**

Purpose: Determines if the device generates a modulated version of the waveform using the original waveform as a carrier and input from an external terminal as the signal

HTBasic Definition:

```
LONG Result=FNHx_getaofungmtp((LONG Taskid), (Channel$), LONG Data)

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to work with

Return value:   0 = Function was successful
                    Data  Determines the modulated version of the waveform (FM/AM/no)
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_setaofungmtp NI-DAQmx 8.6: **DAQmxSetAOFuncGenModulationType**

Purpose: Specifies if the device generates a modulated version of the waveform using the original waveform as a carrier and input from an external terminal as the signal

HTBasic Definition:

```
LONG Result=FNHx_setaofungmtp((LONG Taskid), (Channel$), (LONG Data))

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to set
 Data              Specifies the modulated version of the waveform (FM/AM/no)

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_rstaofungmtp NI-DAQmx 8.6: **DAQmxResetAOFuncGenModulationType**

Purpose: Resets if the device generates a modulated version of the waveform using the original waveform as a carrier and input from an external terminal as the signal to it's default state

HTBasic Definition:

```
LONG Result=FNHx_rstaofungmtp((LONG Taskid), (Channel$))

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to reset

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_getaofungfmd NI-DAQmx 8.6: **DAQmxGetAOFuncGenFMDeviation**

Purpose: Determines the FM deviation in hertz per volt when Type is DAQmx_Val_FM

HTBasic Definition:

```
LONG Result=FNHx_getaofungfmd((LONG Taskid), (Channel$), REAL Data)

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to work with

Return value:   0 = Function was successful
                    Data    Determines the FM deviation in Hertz per Volt
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_setaofungfmd NI-DAQmx 8.6: **DAQmxSetAOFuncGenFMDeviation**

Purpose: Specifies the FM deviation in hertz per volt when Type is DAQmx_Val_FM

HTBasic Definition:

```
LONG Result=FNHx_setaofungfmd((LONG Taskid), (Channel$), (REAL Data))

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to set
 Data              Specifies the FM deviation in Hertz per Volt

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_rstaofungfmd NI-DAQmx 8.6: **DAQmxResetAOFuncGenFMDeviation**

Purpose: Resets the FM deviation in hertz per volt when Type is DAQmx_Val_FM to it's default value

HTBasic Definition:

```
LONG Result=FNHx_rstaofungfmd((LONG Taskid), (Channel$))

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to reset

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_getsamtimeng — NI-DAQmx 8.6: DAQmxGetSampTimingEngine

Purpose: Determines which timing engine to use for the specified timing type. Refer to device documentation for information on supported timing engines

HTBasic Definition:

```
LONG Result=FNHx_getsamtimeng((LONG Taskid), (Channel$), LONG Data)

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to work with

Return value:   0 = Function was successful
                    Data  Determines which timing engine to use
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_setsamtimeng — NI-DAQmx 8.6: DAQmxSetSampTimingEngine

Purpose: Specifies which timing engine to use for the specified timing type. Refer to device documentation for information on supported timing engines

HTBasic Definition:

```
LONG Result=FNHx_setsamtimeng((LONG Taskid), (Channel$), (LONG Data))

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to set
 Data              Specifies which timing engine to use

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_rstsamtimeng — NI-DAQmx 8.6: DAQmxResetSampTimingEngine

Purpose: Resets which timing engine to use for the specified timing type to it's default state

HTBasic Definition:

```
LONG Result=FNHx_rstsamtimeng((LONG Taskid), (Channel$))

Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to reset

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_getphchaomce NI-DAQmx 8.6: **DAQmxGetPhysicalChanAOManualControlEnable**

Purpose: Determines if you can control the physical channel externally via a manual control located on the device

HTBasic Definition:

```
LONG Result=FNHx_getphchaomce((Channel$), LONG Data)

Input:
 Channel$          The name of the channel you want to work with

Return value:   0 = Function was successful
                    Data  Determines if you can control the physical channel
                          externally via a manual control located on the device
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_setphchaomce NI-DAQmx 8.6: **DAQmxSetPhysicalChanAOManualControlEnable**

Purpose: Specifies if you can control the physical channel externally via a manual control located on the device

HTBasic Definition:

```
LONG Result=FNHx_setphchaomce((Channel$), (LONG Data))

Input:
 Channel$          The name of the channel you want to set
 Data              Determines if you can control the physical channel
                   externally via a manual control located on the device


Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_rstphchaomce NI-DAQmx 8.6: **DAQmxResetPhysicalChanAOManualControlEnable**

Purpose: Resets if you can control the physical channel externally via a manual control located on the device to it's default state

HTBasic Definition:

```
LONG Result=FNHx_rstphchaomce((Channel$))

Input:
 Channel$          The name of the channel you want to reset

Return value:   0 = Function was successful
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_getpcaomcamp NI-DAQmx 8.6: **DAQmxGetPhysicalChanAOManualControlAmplitude**

Purpose: Indicates the current value of the front panel amplitude control for the physical channel in volts

HTBasic Definition:

```
LONG Result=FNHx_getpcaomcamp((Channel$), REAL Data)

Input:
 Channel$          The name of the channel you want to work with

Return value:   0 = Function was successful
                    Data  Indicates the current value of the front panel amplitude
                          control for the physical channel
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_getpcaomcfrq NI-DAQmx 8.6: **DAQmxGetPhysicalChanAOManualControlFreq**

Purpose: Indicates the current value of the front panel frequency control for the physical channel in hertz

HTBasic Definition:

```
LONG Result=FNHx_getpcaomcfrq((Channel$), REAL Data)

Input:
 Channel$          The name of the channel you want to work with

Return value:   0 = Function was successful
                    Data  Indicates the current value of the front panel frequency
                          control for the physical channel in Hertz
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.6 and higher.

FNHx_getdevtermin NI-DAQmx 8.7: **DAQmxGetDevTerminals**

Purpose: Indicates a list of all terminals on the device

HTBasic Definition:

```
LONG Result=FNHx_getdevtermin((Device$), Data$, (LONG Bufsize))

Input:
 Device$           The name of the device you want to work with
 Bufsize           The dimensioned length of the string variable Data$

Return value:   0 = Function was successful
                    Data$ A list of all terminals on the device
              <>0 = Error occured
```

Note: This function is only available if you are using NI-DAQmx 8.7 and higher.

FNHx_setbufattrib — DAQmxSetBufferAttribute

Purpose: Set buffer attributes.

HTBasic Definition:

```
LONG Result= FNHx_setbufattrib((LONG Taskid), (LONG Attrib1), (LONG Attrib2))
```

```
Input:
 Taskid            The task used in this function
 Attrib1           First attribute
 Attrib2           Second attribute

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_settriattrib — DAQmxSetTrigAttribute

Purpose: Set trigger attributes.

HTBasic Definition:

```
LONG Result= FNHx_settriattrib((LONG Taskid), (LONG Attrib1), (LONG Attrib2))
```

```
Input:
 Taskid            The task used in this function
 Attrib1           First attribute
 Attrib2           Second attribute

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_setcopulfreq — DAQmxSetCOPulseFreq

Purpose: Set counter output pulse frequency.

HTBasic Definition:

```
LONG Result= FNHx_setcopulfreq((LONG Taskid), (Channel$), (REAL Data))
```

```
Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to work with
 Data              Pulse frequency

Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_setcopdutcyc — DAQmxSetCOPulseDutyCyc

Purpose: Set counter output duty cycle.

HTBasic Definition:

```
LONG Result= FNHx_setcopdutcyc((LONG Taskid), (Channel$), (REAL Data))
```

```
Input:
 Taskid            The task used in this function
 Channel$          The name of the channel you want to work with
 Data              Duty cycle

Return value:   0 = Function was successful
              <>0 = Error occured
```

Callback services

Introduction: Software Events

Software events provide an asynchronous notification mechanism for a set of DAQ events. Unlike hardware events, software events do not require you to use a thread to wait until data is available. Using event-based programming, you can write an application that continues to perform work while waiting for data without resorting to developing a multi-threaded application.

NI-DAQmx supports the following software events:

Every N Samples Acquired Into Buffer Event—Occurs when the user-defined number of samples is written from the device to the PC buffer. This event works only with devices that support buffered tasks. *Note:* The value you set for this event must be evenly divisible into the buffer size if you are using DMA as your data transfer mechanism. For instance, if the buffer size is 1,000 samples, specifying 102 for this software event generates an error. Specifying 100, however, would not generate an error. If you are using IRQ as the data transfer method, the value does not need to be evenly divisible. With IRQ, however, the Data Transfer Request Condition attribute/property can affect when this software event occurs.

Every N Samples Transferred From Buffer Event—Occurs when the user-defined number of samples is written from the PC buffer to the device. This event works only with devices that support buffered tasks. *Note*: The value you set for this event must be evenly divisible into the buffer size if you are using DMA as your data transfer mechanism. For instance, if the buffer size is 1,000 samples, specifying 102 for this software event generates an error. Specifying 100, however, would not generate an error. If you are using IRQ as the data transfer method, the value does not need to be evenly divisible. With IRQ, however, the Data Transfer Request Condition attribute/property can affect when this software event occurs.

Done Event—Occurs when the task completes execution or when an error causes the task to finish. Recoverable errors that do not cause the task to finish do not cause this event to fire.

Signal Event—Occurs when the specified hardware signal occurs. Supported signals include the counter output event, change detection event, sample complete event, and the sample clock.
The DAQmx library supports callback services.

HTBasic: Software Events creates ON SIGNAL events

All events are supported by the HTBasic library too. For every event handler you create using the functions described later in this chapter you must register an ON SIGNAL event handler in your HTBasic program for every context where the event may occur. In case such an event is generated in the driver HTBasic gets a SIGNAL and calls the defined ON SIGNAL handler where you can e.g. retrieve your measurement data from the buffer or transfer new data into the output buffer or whatever else might be useful.
Several example programs which demonstrates the events can be found in the subdirectory "`HTB Examples\Events`" inside the installation directory of the HTBasic library.

FNHx_regevnsmevnt NI-DAQmx: **DAQmxRegisterEveryNSamplesEvent**

Purpose: Registers a callback function to receive an event when the specified number of samples is written from the device to the buffer or from the buffer to the device. This function only works with devices that support buffered tasks.

HTBasic Definition:

```
LONG Result=FNHx_regevnsmevnt((LONG Taskid), (LONG Eventtype), (LONG Nsamples),
                              (LONG Options), (LONG Htb_signal_id))

Input:
 Taskid           The task used in this function
 Eventtype        The type of event you want to receive
 Nsamples         The number of samples after which each event should occur
 Options          Use this parameter to set certain options (see DAQmx documentation)
 Htb_signal_id    The ID of the HTBasic signal which should be generated (0 .. 15)

Return value:   0 = Function was successful
              <>0 = Error occured
```

Example:

```
 INTEGER Iresult, Debug
 LONG Taskid, Nsamples, Options, Eventtype, Htb_signal_id
 Nsamples=1000                          ! callback every ... samples
 Options=0                              ! no special options
 Htb_signal_id=15                       ! signal 15 for callback
 Eventtype=FNHx_get_constant("DAQmx_Val_Acquired_Into_Buffer",Iresult,Debug)
 !
 Result=FNHx_regevnsmevnt((Taskid),(Eventtype),(Nsamples),(Options),(Htb_signal_id))
 !
 ON SIGNAL Htb_signal_id,15 CALL Cb_everynsample         ! define ON SIGNAL routine
```

FNHx_regsignevent — NI-DAQmx: DAQmxRegisterSignalEvent

Purpose: Registers a callback function to receive an event when the specified hardware event occurs.

HTBasic Definition:

```
LONG Result=FNHx_regsignevent((LONG Taskid), (LONG Signal_id), (LONG Options),
                              (LONG Htb_signal_id))
```

```
Input:
 Taskid            The task used in this function
 Signal_id         The signal for which you want to receive results. Possible values:
                    DAQmx_Val_SampleClock, DAQmx_Val_SampleCompleteEvent
                    DAQmx_Val_ChangeDetectionEvent, DAQmx_Val_CounterOutputEvent
 Options           Use this parameter to set certain options (see DAQmx documentation)
 Htb_signal_id     The ID of the HTBasic signal which should be generated (0 .. 15)
Return value:   0 = Function was successful
              <>0 = Error occured
```

Example:

```
 INTEGER Iresult, Debug
 LONG Taskid, Options, Signal_id, Htb_signal_id
 Signal_id=FNHx_get_constant("DAQmx_Val_SampleCompleteEvent",Iresult,Debug)
 Options=0                              ! no special options
 Htb_signal_id=14                       ! signal 14 for callback
 !
 Result=FNHx_regsignevent((Taskid),(Signal_id),(Options),(Htb_signal_id))
 !
 ON SIGNAL Htb_signal_id,15 CALL Cb_signalevent        ! define ON SIGNAL routine
```

FNHx_regdoneevent — NI-DAQmx: DAQmxRegisterDoneEvent

Purpose: Registers a callback function to receive an event when a task stops due to an error or when a finite acquisition task or finite generation task completes execution.

HTBasic Definition:

```
LONG Result=FNHx_regdoneevent((LONG Taskid), (LONG Options), (LONG Htb_signal_id))
```

```
Input:
 Taskid            The task used in this function
 Options           Use this parameter to set certain options (see DAQmx documentation)
 Htb_signal_id     The ID of the HTBasic signal which should be generated (0 .. 15)
Return value:   0 = Function was successful
              <>0 = Error occured
```

Example:

```
 INTEGER Iresult, Debug
 LONG Taskid, Options, Htb_signal_id
 Options=0                              ! no special options
 Htb_signal_id=13                       ! signal 14 for callback
 !
 Result=FNHx_regdoneevent((Taskid),(Options),(Htb_signal_id))
 !
 ON SIGNAL Htb_signal_id,15 CALL Cb_Doneevent        ! define ON SIGNAL routine
```

Watchdog functions

FNHx_crwdogtimtsk NI-DAQmx: **DAQmxCreateWatchdogTimerTask**

Purpose: Creates and configures a task that controls the watchdog timer of a device

HTBasic Definition:

```
LONG Result=FNHx_crwdogtimtsk((Devicename$), (Taskname$),(LONG Taskid), (REAL T_out),
                              (Lines$), (LONG Expstate))
```

```
Input:
 Devicename$       The name of the device, as configured in MAX, to which this operation
                   applies
 Taskname$         The name of the task
 T_out             The amount of time, in seconds, until the watchdog timer expires
 Lines$            The digital line or port to modify. You cannot modify dedicated
                   digital input lines. You can specify a list or range of physical
                   channels
 Expstate          The state to which to set the digital physical channel when the
                   watchdog timer expires

Return value:   0 = Function was successful
                    Taskid now contains Task ID of newly created task
              <>0 = Error occured
```

Please note: This function differs from the original DAQmx function since you can only create a watchdog task for one digital line or line range with one function call. In case you need to change more than one line or line range you must create several separate watchdog timers.

FNHx_ctrlwdogtsk NI-DAQmx: **DAQmxControlWatchdogTask**

Purpose: Controls the watchdog task according to the action you specify

HTBasic Definition:

```
LONG Result=FNHx_ctrlwdogtsk((LONG Taskid), (LONG Action))
```

```
Input:
 Taskid            The task used in this function
 Action            Specifies how to control the watchdog task

Return value:   0 = Function was successful
              <>0 = Error occured
```

Storage

FNHx_savetask NI-DAQmx: **DAQmxSaveTask**

Purpose: Saves the specified task and any local channels it contains to MAX

HTBasic Definition:

```
LONG Result=FNHx_savetask((LONG Taskid), (Saveas$), (Author$), (LONG Options))

Input:
 Taskid            The task to save
 Saveas$           Name to save the task as. If you pass an empty string ("") or NULL,
                   the name currently assigned to the task will be used
 Author$           Name to store with the task

Return value:   0 = Function was successful
              <>0 = Error occured
```

Example:

```
 LONG Result, Taskid, Options, Options_1, Options_2
 DIM Saveas$[255], Author$[255]
 INTEGER Iresult, Debug
 Saveas$="MySavedTaskName"
 Author$="MyName"
 Options_1=FNHx_get_constant("DAQmx_Val_Save_Overwrite",Iresult,Debug)
 Options_2=FNHx_get_constant("DAQmx_Val_Save_AllowInteractiveEditing",Iresult,Debug)
 Options=Options_1+Options_2
 Result=FNHx_savetask((Taskid), (Saveas$), (Author$), (Options))
 !
 IF Result=0 THEN
    !... subsequent commands...
 END IF
```

FNHx_delsavedtask NI-DAQmx: **DAQmxDeleteSavedTask**

Purpose: Deletes the specified task from MAX

HTBasic Definition:

```
LONG Result=FNHx_delsavedtask((Taskname$))

Input:
 Taskname$         Name of the saved task to be deletede

Return value:   0 = Function was successful
              <>0 = Error occured
```

Example:

```
 LONG Result
 DIM Saveas$[255]
 Saveas$="MySavedTaskName"
 Result=FNHx_delsavedtask((Saveas$))
 !
 IF Result=0 THEN
    !... subsequent commands...
 END IF
```

HTBasic specific functions

FNHx_manage_subs

Purpose: Load / unload and init / de-init HTBasic library functions

HTBasic Definition:

```
LONG Result=FNHx_manage_subs(Func$, OPTIONAL INTEGER Verbose, Debug)

Input:
 Func$              Function code
                    "LOAD"   = Load and initialize DAQmx_HTB library
                    "UNLOAD" = Unload and de-initialize DAQmx_HTB library
 Verbose            Verbose messages (0=none, 1=all, 2=Errors only)
Debug

Return value:   0 = Function was successful
               <>0 = Error occured
```

Example:

```
 INTEGER Verbose
 IF NOT FNHx_manage_subs("LOAD",Verbose) THEN STOP   ! load and init DAQmx library
```

FNHx_get_constant

Purpose: Returns the value of a numeric constant from `NIDAQmx.h` from the NI-DAQmx Library

HTBasic Definition:

```
DEF FNHx_get_constant(Constant$,INTEGER Result, OPTIONAL INTEGER Debug)

Input:
 Constant$          Name of DAQmx constant as defined in file "NIDAQmx.h"
 Debug              Shows debug infos (constant found / not found etc.)
                     Debug = 3 : Show all debug messages

Return value:   0 = Function was successful
               <>0 = Error occured

If the function was successful it returns the constant as INTEGER , LONG, or REAL
(according to the definition of the constant)
```

Example:

```
 INTEGER Iresult, Debug
 LONG Units
 Debug=3                                ! Shows comprehensive debug messages (test mode)
 Units=FNHx_get_constant("DAQmx_Val_Volts",Iresult,Debug)         ! get LONG constant
 IF Iresult=0 THEN
    !... weitere Befehle ...
 END IF
```

FNHx_get_constant$

Purpose: Returns the value of a constant from `NIDAQmx.h` from the NI-DAQmx Library as a string

HTBasic Definition:

```
Const$=FNHx_get_constant$(Constant$,INTEGER Result, Rtype$, OPTIONAL INTEGER Debug)

Input:
 Constant$          Name of DAQmx constant as defined in file "NIDAQmx.h"
 Debug              Shows debug infos (constant found / not found etc.)
                     Debug = 3 : Show all debug messages

Return value:   0 = Function was successful
                    Result = 1: Function was sucessful   0: Function failed
               <>0 = Error occured

If the function was successful it returns the constant as a string and the variable type
in Rtype$ ("REAL", "SHORTHEX", "LONGHEX", "STRING", "LONG")

Use this function in order to retrieve string constants, e.g. for constants for switch
topologies.
```

SUB Hx_show_errstat

Purpose: Shows comprehensive information about current error status

HTBasic Definition:
```
SUB Hx_show_errstat(LONG Result, OPTIONAL Fname$,INTEGER Verbose)
```

```
Input:
 Result            Result of last DAQmx function HTBasic function
 Fname$            Name of function (optional)
 Verbose           Verbose messages (0 = none, 1 = all, 2 = errors only)
Return value:   0 = Function was successful
              <>0 = Error occured
```

FNHx_getdaqmxvers$

Purpose: Returns version number of NI-DAQmx driver DLL "nicaiu.dll"

HTBasic Definition:
```
Daqvers$=FNHx_getdaqmxvers$
```

```
Return value:   If function was successful it returns version number, e.g. "8.0"
                If function failed the returned string is ""
```

FNHx_getdllvers$

Purpose: Returns version number of DAQmx_htb wrapper DLL "DAQmx_htb.dll"

HTBasic Definition:
```
Dllvers$=FNHx_getdllvers$
```

```
Return value:   If function was successful it returns version number, e.g. "1.01.085"
                If function failed the returned string is ""
```

FNHx_getdevname$

Purpose: Extracts device name from device string or device path (strips leading "/" if necessary)

HTBasic Definition:
```
Dname$=FNHx_getdevname$(Devname$)
```

```
Input:
 Devname$          Name of device or path, e.g. "Dev1" or "/Dev1/ao0"
Return value:   If function was successful it returns device name, e.g. "Dev1"
                If function failed the returned string is ""
```

FNHx_lib_version$

Purpose: Returns version number of HTBasic DAQmx library "DAQmx_htb.lib"

HTBasic Definition:
```
Libvers$=FNHx_lib_version$
```

```
Return value:   If function was successful it returns version number, e.g. "1.03.116"
                If function failed the returned string is ""
```

FNHx_init

Purpose: Initializes HTBasic function library
Note: This function will be automagically called from FNHx_manage_subs if you load the function library, so normally it is *not* necessary to call this function.

HTBasic Definition:
```
LONG Result=FNHx_init
```

```
Return value:   1 = Function was successful
                0 = Error occured
```

FNHx_initlib

Purpose: Initializes the DAQmx wrapper DLL "DAQmx_htb.dll" (clears memory etc.)
Note: This function will be automagically called from FNHx_manage_subs / FNHx_init if you load the function library, so normally it is *not* necessary to call this function.

HTBasic Definition:
```
LONG Result=FNHx_initlib
```

```
Return value:   0 = Function was successful
                1 = Error occured
```

SUB Hx_enter_kbd

Purpose: Enhanced ENTER KBD function. Uses windows dialog box if Basic Plus is loaded

HTBasic Definition:
```
SUB Hx_enter_kbd(Text$)
```

```
Input:
 Text$              Text to display

Return value:   none
```

FNHx_get_dev_info

Purpose: Retrieves a number of interesting device parameters and returns it in a string array and (optional) in a file

HTBasic Definition:
```
LONG Result=FNHx_get_dev_info(Devname$, Dev_info$, OPTIONAL INTEGER Verbose, Write2File)
```

```
Input:
 Devname$           Name of device or path, e.g. "Dev1" or "/Dev1/ao0"
 Verbose            Verbose messages (0 = none, 1 = all, 2 = errors only)
 Write2File         1: Write device info to file "daqmx_devinfo.txt" in program directory
                    0: (or omit variable) Do not write device info to file

Return value:   1 = Function was successful
                    If function was successful it returns comprehensive device info in
                    variable Dev_info$. Make sure that this variable is dimensioned big
                    enough to hold info (at least 400 characters).
                0 = Error occured
                    If function failed the returned string is "" or truncated string if
                    Dev_info$ is too small to hold complete device info.
```

FNHx_gettnamwpref — NI-DAQmx: GetTerminalNameWithDevPrefix

Purpose: Retrieves terminal name with prefix information for the given task

HTBasic Definition:
```
LONG Result=FNHx_gettnamwpref((LONG Taskid), Terminalname$, Triggername$)
```

```
Input:
 Taskid             The task used in this function

Return value:   0 = Function was successful
                      Triggername$ contains the trigger name
                      Terminalname$ contains Terminal name
              <>0 = Error occured
```

FNHx_numofdevices

Purpose: Retrieves the number of installed devices in NI MAX (Measurement & Automation Explorer)

HTBasic Definition:
```
LONG Result=FNHx_numofdevices(OPTIONAL Devnames$(*))
```

```
Return value:  >0 = (Function was successful): Number of devices
                    Devnames$ contains the list of device names available in MAX
                0 = Error occured
```

FNHx_getsfpath$

Purpose: Returns the path to special windows directories.

HTBasic Definition:

```
Sfpath$=FNHx_getsfpath$(Foldercsid$,Subdir$,INTEGER Dflag,OPTIONAL INTEGER Verbose)
```

Input: Foldercsid$ = Path CSIDLS

Foldercsid$ value	English Windows	(German translation)
CSIDL_DESKTOP	Desktop	(Arbeitsplatz)
CSIDL_INTERNET	Internet	
CSIDL_PROGRAMS	Start menu: Programs	(Startmenü: Programme)
CSIDL_CONTROLS	Control Panel	(Systemsteuerung)
CSIDL_PRINTERS	Printer	(Drucker)
CSIDL_PERSONAL	My Documents	(Eigene Dateien)
CSIDL_FAVORITES	IE: Favorites	(Favoriten)
CSIDL_STARTUP	Autostart	
CSIDL_RECENT	Recent documents	(Zuletzt benutzte Dokumente)
CSIDL_SENDTO	SendTo	(Senden an)
CSIDL_BITBUCKET	Recycle Bin	(Papierkorb)
CSIDL_STARTMENU	Startmenu	(Startmenü)
CSIDL_MYMUSIC	My Music	(Eigene Musik)
CSIDL_MYVIDEO	My Video	(Eigene Videos)
CSIDL_DESKTOPDIRECTORY	Desktop Directory	(Desktopverzeichnis)
CSIDL_DRIVES	My Computer	(Mein Computer)
CSIDL_NETWORK	Network	(Netzwerk)
CSIDL_NETHOOD	Network Neighbourhood	(Netzwerkumgebung)
CSIDL_FONTS	Windows\Fonts	
CSIDL_TEMPLATES	Templates	(Vorlagen)
CSIDL_COMMON_STARTMENU	"All Users" - Startmenu	(Startmenü)
CSIDL_COMMON_PROGRAMS	"All Users" - Programs	(Programme)
CSIDL_COMMON_STARTUP	"All Users" - Autostart	
CSIDL_COMMON_DESKTOPDIRECTORY	"All Users" - Desktop	
CSIDL_APPDATA	Application Data	(Anwendungsdaten)
CSIDL_PRINTHOOD	Printhood	(Druckumgebung)
CSIDL_LOCAL_APPDATA	Local Settings\Application Data (Lokale Einstellungen\Anwendungsdaten)	
CSIDL_COMMON_FAVORITES	"All Users" - Favorites	(Favoriten)
CSIDL_INTERNET_CACHE	IE: Temorary Internet files	(Temporäre Internetdateien)
CSIDL_COOKIES	IE: Cookies	
CSIDL_HISTORY	IE: History	(Verlauf)
CSIDL_COMMON_APPDATA	"All Users" - Application Data	(Anwendungsdaten)
CSIDL_WINDOWS	Windows	
CSIDL_SYSTEM	Windows\System32	
CSIDL_PROGRAM_FILES	C:\Program Files	(C:\Programme)
CSIDL_MYPICTURES	My Pictures	(Eigene Bilder)
CSIDL_PROFILE	Profile (Anwenderprofil	(Benutzername))
CSIDL_SYSTEMX86	Windows\System32	
CSIDL_PROGRAM_FILES_COMMON	Common Files	(Gemeinsame Dateien)
CSIDL_COMMON_TEMPLATES	"All Users" - Templates	(Vorlagen)
CSIDL_COMMON_DOCUMENTS	"All Users" - Documents	(Dokumente)
CSIDL_COMMON_ADMINTOOLS	"All Users" - AdminTools	(Verwaltung)
CSIDL_ADMINTOOLS	Startmenu\Program Files\AdminTools (Startmenü\Programme\Verwaltung)	

Subdir$ = Subdirectory to be added to directory name
Dflag = Flags.
Currently defined: 1 = Create subdir if it does not exist

Return value: **Sfpath$**: Returns the path or "" if the path does not exist

Index

B

C

D

E

F

G

H

I

L

M

N

O

R

S

T

W

Tech Soft GmbH
Test & Measurement
Karmeliterweg 114
D - 13465 Berlin, Germany

ph +49 - 30 - 40608-0
fax +49 - 30 - 40608-220
e-mail: HTB@TechSoft.de

www.TechSoft.de