

Getting Started

This topic explains how to begin using NI-HWS with your application development environment (ADE), lists files to include in your application, and mentions special considerations for each ADE.

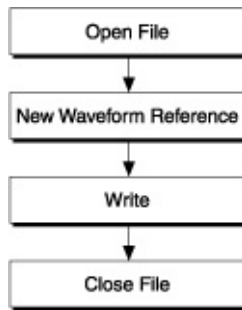
To successfully build your application, you need to install one of the following ADEs along with NI-HWS:

- [LabVIEW](#)
- [LabWindows™/CVI™](#)
- [C/C++](#)

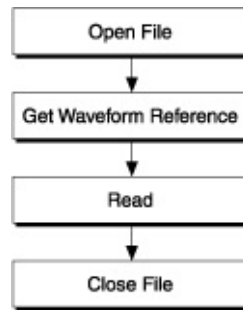
Basic Programming Flows

The following diagrams show the basic programming flows for applications using NI-HWS.

Basic Write Programming Flow



Basic Read Programming Flow



The high-level Store and Retrieve functions and VIs combine all the above steps.

NI-HWS Key Concepts

[Groups](#), [waveforms](#), and [scaling](#) are concepts central to the NI-HWS API.

Groups

Groups are like folders, giving NI-HWS a one-level-deep storage hierarchy. A single HWS file may contain multiple groups. Each group may contain multiple [waveforms](#). A group attribute applies to all the waveforms contained in the group.

When only one group is in a file, the group does not need to be named. If a group is named and is the only one in the file, the name is also not required to access any of the waveforms stored in that group. When two or more groups are in the file, a group name is required.

Waveforms

A waveform is stored in an HWS file as an array of analog or digital data. The elements in an analog data array can include the following types:

- Analog waveform data type (WDT)
- Double-precision floating-point number (F64)
- 8-bit signed integer (I8)
- 16-bit signed integer (I16)
- 32-bit signed integer (I32)

The elements in a digital data array can include the following types:

- Digital waveform data type (WDT)
- 8-bit unsigned integer (U8)
- 16-bit unsigned integer (U16)
- 32-bit unsigned integer (U32)

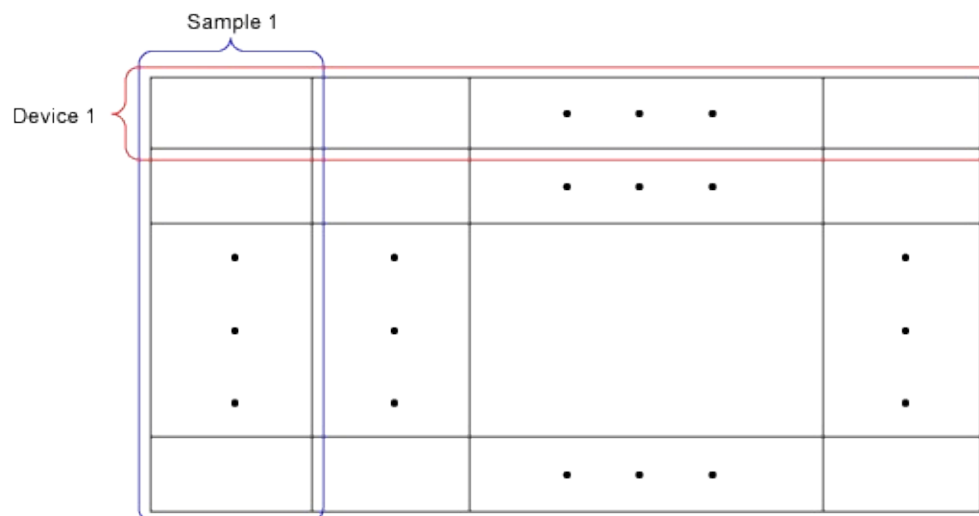


Note The Waveform Data Type (WDT) (supported in LabVIEW versions 7.0 and later and in the C API for NI-HWS 1.4 and later) is a convenient way to manage data storage.

You can also save digital data in a two-dimensional array of unsigned integers. In this representation, you can think of your waveform as a table where the columns represent samples, and the rows represent devices, as shown in the following figure.



Note Rows are contiguous in memory.



Waveforms are stored in groups with one or more waveforms per group.

A waveform attribute is intended to apply only to its waveforms, not to the group.

A waveform reference is required to read or write any waveform data or attributes. Waveform names are used to obtain waveform references. However, if only one waveform is in the group, the waveform name is not needed. When two or more waveforms are in a group, the waveform name is required.

Refer to [Data Conversion](#) for more information about changing one waveform type to another.

Data Conversion

NI-HWS can convert some waveform data from the data type in which it was stored to a different data type when the data is read or retrieved. The table below shows the supported type conversions. A "Yes" in a box means that data stored as one data type can be retrieved as the other data type; a dash means the conversion is not supported.

F64 and DBL are both used by NI-HWS to mean double-precision floating-point numbers.

Analog Data Types

Stored Type	Retrieved Type				
	Analog WDT	Analog I8	Analog I16	Analog I32	Analog F64
Analog WDT	Yes	—	—	—	Yes
Analog F64	Yes	—	—	—	Yes
Analog I8	Yes	Yes	Yes	Yes	Yes
Analog I16	Yes	—	Yes	Yes	Yes
Analog I32	Yes	—	—	Yes	Yes

Digital Data Types

Stored Type	Retrieved Type						
	Digital WDT	Digital 1D U8	Digital 2D U8	Digital 1D U16	Digital 2D U16	Digital 1D U32	Digital 2D U32
Digital WDT	Yes	—	—	—	—	—	—
Digital 1D U8	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Digital 2D U8	Yes	—	Yes	—	Yes	—	Yes
Digital 1D U16	Yes	Yes*	Yes*	Yes	Yes	Yes	Yes
Digital 2D U16	Yes	—	Yes*	—	Yes	—	Yes
Digital 1D U32	Yes	Yes*	Yes*	Yes*	Yes*	Yes	Yes
Digital 2D U32	Yes	—	Yes*	—	Yes*	—	Yes

*For digital waveforms, smaller data types can read larger data types with the following restrictions: the data must be mapped and all mapped bits must be within the smaller data types range.

Waveforms are stored in [groups](#) with one or more waveforms per group. A waveform attribute is intended to apply only to its waveforms, not to the group.

A waveform reference is required to read or write any waveform data or attributes. Waveform names are used to obtain waveform references. However, if only one waveform is in the group, the waveform name is not needed. When two or more waveforms are in a group, the waveform name is required.

Scaling

NI-HWS stores polynomial scaling coefficients that are intended to be applied to analog waveform data by your application or your hardware. The **offset** and **gain** parameters in the high-level store and retrieve functions and VIs are identical to the first two elements in the polynomial **coefficient scaling** array in the Get and Set Scaling Coefficients functions and VIs.



Notes When analog data is stored in an integer format, but read back with a floating point Read function or VI, NI-HWS scales the data before returning it.

You cannot apply scaling to digital waveforms.

Using NI-HWS in LabVIEW

This topic assumes that you are using the LabVIEW ADE and that you are familiar with the ADE.

To develop an NI-HWS application in LabVIEW, follow these general steps:

1. Open an existing or new LabVIEW VI.
2. From the Functions Palette, locate the NI-HWS VIs inside the NI-FGEN, NI-HSDIO, NI-SCOPE or Instrument Drivers palettes.
3. Select the VIs that you want to use and drop them on the block diagram to build your application.

Using NI-HWS in LabWindows/CVI

This topic assumes that you are using the LabWindows/CVI ADE to manage your code development and that you are familiar with the ADE.

To develop an NI-HWS application in LabWindows/CVI, follow these general steps:

1. Open an existing or new project file.
2. Load the NI-HWS function panel (nihws.fp) from <CVI>\instr by selecting **Instrument»Load**.
3. Use the function panel to navigate the function hierarchy and generate function calls with the proper syntax and variable values.

Using NI-HWS in C/C++

This topic assumes that you are using a C/C++ ADE to manage your code development and that you are familiar with the ADE.

To develop an NI-HWS application in C/C++, follow these general steps:

1. Open an existing or new C/C++ project.
2. Make sure that you include the NI-HWS header file (niHWS.h) as follows in your source code files: `#include "niHWS.h"`
3. Specify the directory that contains the NI-HWS header file under **C/C++»Preprocessor»Additional include directories**. The NI-HWS header files are located in the `<NI-HWS>\Include` directory.
4. Add the NI-HWS import library (nihws.lib) to the project under **Link»General»Object/Library Modules**. The NI-HWS import library files are located in the `<NI-HWS>\Lib` directory.
5. Add NI-HWS function calls to your application.
6. Build your application.



Tip By default, C passes parameters by value. Remember to pass pointers to variables when you need to pass by address.

Using Attributes with NI-HWS

NI-HWS contains high-level functions and VIs that set most of the waveform storage attributes.

Some attributes, such as the units or the label for a waveform axis, are not accessible through the high-level functions and VIs. The values for these attributes must be set using a Set Attribute function.

Accessing Attributes

In LabVIEW, you can access attributes with the Get and Set Attribute VIs. A pair of Get and Set VIs exists for both the [Group](#) attributes and the [Waveform](#) attributes.

In C, attributes are accessed with the Get and Set Attribute functions.

Refer to the [NI-HWS VI Reference](#) or the [NI-HWS C Function Reference](#) for a complete listing of available attributes.

Programming

Expand this topic for information about the NI-HWS programming flow, for VI and C function reference information, and for error and status codes.

[VI Function Reference](#)

[C Function Reference](#)