In order to open a 7.3 MAT file in the HDF5 Browser you will need to select All Files from the popup menu in the HDF5 Browser Open File dialog. If you try to open a MAT file that is not version 7.3 as an HDF5 file, the HDF5 library will return an error.

For an example of saving Matlab data in the version 7.3 format, see https://www.mathworks.com/help/matlab/ref/save.html. You can also set a preference in Matlab to make version 7.3 your default format.

Loading General Binary Files

General binary files are binary files created by other programs. If you understand the binary file format, it is possible to load the data into Igor. However, you must understand the binary file format *precisely*. This is usually possible only for relatively simple formats.

There are two ways to load data from general binary files into Igor:

- Using the **FBinRead** operation
- Using the **GBLoadWave** operation

Using FBinRead is somwhat more difficult and more flexible than using GBLoadWave. It is especially useful for loading data stored as structures into Igor. For details, see **FBinRead**. This section focuses on using the GBLoadWave operation.

GBLoadWave loads data from general binary files into Igor waves. "GB" stands for "general binary".

You can invoke the GBLoadWave operation directly or by choosing Data→Load Waves→Load General Binary File which displays the Load General Binary dialog.

You need to know the format of the binary file precisely in order to successfully use GBLoadWave. Therefore, it is of use mostly to load binary files that you have created from your own program. You can also use GBLoadWave to load third party files if you know the file format precisely.

Files GBLoadWave Can Handle

GBLoadWave handles the following types of binary data:

- 8 bit, 16 bit, 32 bit, and 64 bit signed and unsigned integers
- 32 and 64 bit IEEE floating point numbers
- 32 and 64 bit VAX floating point numbers

In addition, GBLoadWave handles high-byte-first (Motorola) and low-byte-first (Intel) type binary numbers.

GBLoadWave currently can not handle IEEE or VAX extended precision values. See **VAX Floating Point** for more information.

GBLoadWave can create waves using any of numeric data types that Igor supports (64-bit and 32-bit IEEE floating point, 64-bit, 32-bit, 16-bit and 8-bit signed and unsigned integers). The data type of the wave does not need to be the same as the data type of the file. For example, if you have a file containing integer A/D readings, you can load that data into a single-precision or double-precision floating point wave.

In general, it is best to load waves as floating point since nearly all Igor operations work faster on floating point. One exception is when you are dealing with images, especially stacks of images. For example, if you have a 512x512x1024 byte image stack in a file, you should load it into a byte wave. This takes one quarter of the memory and disk space of a single-precision floating point wave.

GBLoadWave knows nothing about Igor multi-dimensional waves. It knows about 1D only. The term "array", used in the GBLoadWave dialog, means "1D array". However, after loading data as a 1D wave, you can redimension it as required.