FrameLib: Alpha 0.1 Documentation

Alex Harker

Generated on 6/9/2017 at 17:17

Contents

1 Introduction

1.1 Preliminaries

This document is designed to go with alpha releases of the Max version of FrameLib (which is currently the only extant version). The documentation is currently intended to give enough information to get experienced Max users started and and explain basic concepts, and as such is non-comprehensive.

1.2 What is FrameLib?

FrameLib is a DSP system designed to allow quick modular constructions of frame-based networks. These networks can resolve time at a highly accurate subsample level and can run at different rates, and with different frame sizes across a single network. This allows mixing of different representations (e.g. time and frequency-based representations) in a single network, efficient processing when data sizes are varying and tightly timed DSP. Most of you should already basically know what it is (otherwise you wouldn't be testing), so that will do for now...

1.3 How is this an 'Alpha' Version?

In general the software should be relatively stable, but there are several items that are subject to change in terms of usage, which means **you should not use FrameLib for any large/final version projects yet!** You would do so at your own risk. The is a section later in this documentation that covers items subject to change and specific questions about these items.

1.4 How Can I Help?

Thanks for asking. Please comment on the usability of objects, features, points of confusion. Object ideas are also welcome, but are lower priority. Comments on the end questions are also welcome. Offers of help with documentation (or help patches) are also welcome, although most likely any help patches are a little way off yet. Obviously let me know of any bugs you might find. Developer documentation (for those wishing to write C++ objects) will also come later.

2 Key Concepts

2.1 Schedulers and Timing

Morbi luctus, wisi viverra faucibus pretium, nibh est placerat odio, nec commodo wisi enim eget quam. Quisque libero justo, consectetuer a, feugiat vitae, porttitor eu, libero. Suspendisse sed mauris vitae elit sollicitudin malesuada. Maecenas ultricies eros sit amet ante. Ut venenatis velit. Maecenas sed mi eget dui varius euismod. Phasellus aliquet volutpat odio. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Pellentesque sit amet pede ac sem eleifend consectetuer. Nullam elementum, urna vel imperdiet sodales, elit ipsum pharetra ligula, ac pretium ante justo a nulla. Curabitur tristique arcu eu metus. Vestibulum lectus. Proin mauris. Proin eu nunc eu urna hendrerit faucibus. Aliquam auctor, pede consequat laoreet varius, eros tellus scelerisque quam, pellentesque hendrerit ipsum dolor sed augue. Nulla nec lacus.

2.2 Types of Frames

Morbi luctus, wisi viverra faucibus pretium, nibh est placerat odio, nec commodo wisi enim eget quam. Quisque libero justo, consectetuer a, feugiat vitae, porttitor eu, libero. Suspendisse sed mauris vitae elit sollicitudin malesuada. Maecenas ultricies eros sit amet ante. Ut venenatis velit. Maecenas sed mi eget dui varius euismod. Phasellus aliquet volutpat odio. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Pellentesque sit amet pede ac sem eleifend consectetuer. Nullam elementum, urna vel imperdiet sodales, elit ipsum pharetra ligula, ac pretium ante justo a nulla. Curabitur tristique arcu eu metus. Vestibulum lectus. Proin mauris. Proin eu nunc eu urna hendrerit faucibus. Aliquam auctor, pede consequat laoreet varius, eros tellus scelerisque quam, pellentesque hendrerit ipsum dolor sed augue. Nulla nec lacus.

2.3 Parameters

Morbi luctus, wisi viverra faucibus pretium, nibh est placerat odio, nec commodo wisi enim eget quam. Quisque libero justo, consectetuer a, feugiat vitae, porttitor eu, libero. Suspendisse sed mauris vitae elit sollicitudin malesuada. Maecenas ultricies eros sit amet ante. Ut venenatis velit. Maecenas sed mi eget dui varius euismod. Phasellus aliquet volutpat odio. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Pellentesque sit amet pede ac sem eleifend consectetuer. Nullam elementum, urna vel imperdiet sodales, elit ipsum pharetra ligula, ac pretium ante justo a nulla. Curabitur tristique arcu eu metus. Vestibulum lectus. Proin mauris. Proin eu nunc eu urna hendrerit faucibus. Aliquam auctor, pede consequat laoreet varius, eros tellus scelerisque quam, pellentesque hendrerit ipsum dolor sed augue. Nulla nec lacus.

2.4 Multi-channel Expansion

Morbi luctus, wisi viverra faucibus pretium, nibh est placerat odio, nec commodo wisi enim eget quam. Quisque libero justo, consectetuer a, feugiat vitae, porttitor eu, libero. Suspendisse sed mauris vitae elit sollicitudin malesuada. Maecenas ultricies eros sit amet ante. Ut venenatis velit. Maecenas sed mi eget dui varius euismod. Phasellus aliquet volutpat odio. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Pellentesque sit amet pede ac sem eleifend consectetuer. Nullam elementum, urna vel imperdiet sodales, elit ipsum pharetra ligula, ac pretium ante justo a nulla. Curabitur tristique arcu eu metus. Vestibulum lectus. Proin mauris. Proin eu nunc eu urna hendrerit faucibus. Aliquam auctor, pede consequat laoreet varius, eros tellus scelerisque quam, pellentesque hendrerit ipsum dolor sed augue. Nulla nec lacus.

3 Getting Started (Max Specifics)

3.1 Connecting Inputs and Outputs

Only one object can be connected to a frame input. There is simply no one sensible way to combine two sets of frames in a generic manner. When framelib objects are connected directly this is enforced (you are not physically able to connect two objects). When you are using abstraction or subpatching this is not possible, but you will receive an error when dsp recompiles if you've connected two objects to a single input. One output can feed any number of inputs.

3.2 Setting Parameters

Parameters can be set in one of two ways, either at frame rate (if the object has a Parameter Update input), or at object instantiation. At object instantiation the syntax is currently:
either:

#parameter_name value(s) ...

or:

/parameter_name value(s) ...

within the object box. The @ sign is not used to avoid confusion with max attributes (which show in the inspector and can be set in the max thread using messages, which framelib parameters cannot). In addition parameters can be assigned to a given argument number in which case they can be set from object arguments.

To set parameters at frame rate based on values in vectors, use the *fl.setparam∼* object.

3.3 Setting Fixed Inputs

FrameLib objects in Max support input of fixed numeric values at object instantiation that are used for any subsequent calculations, rather than values updated at frame rate. This is accessed in one of two ways. Most binary operators (e.g. *fl.times∼* or *fl.divide∼*) allow you to directly type the value of either input in as one or more arguments to the object (this mimics standard max object behaviour). However, as most objects use arguments to represent parameters (for ease), other objects require a special syntax for setting the value of a specific input. This is currently:
either:

/input_number/ value(s) ...

or:

⟨input_number⟩ value(s) ...

or:

[input_number] value(s) ...

or:

∼input_number value(s) ...

Inputs are numbered from 1 (not zero!). You can enter vectors of any length.

3.4 Control from Max

As everything in max runs at frame rate you cannot just hook up max messages to control inputs. You must go through the *fl.frommax∼* object. This object allows two modes of operation. In the first you can translate single numerical values or lists of numeric values into framelib vectors. In this case you simply pass max messages into the object. The last value(s) received will be sent as a vector when the object is triggered by a frame input.

In the second mode of operation a variable number of inputs allow for max messages to be translated into parameter frames. In this mode each input corresponds to a named parameter, and messages send to each input will be stored until the next frame is triggered. At this point **all** stored parameter changes are sent to the output in a single concatenated frame, and cleared from the object's memory (unlike numeric vectors which will be repeated on subsequent trigger inputs).

3.5 Getting Object Help

All objects support the **info** message. This currently prints reference type information to the max window. This is the only way to get help on individual objects, but should be compact but comprehensive.

Without any arguments message will print a description followed by info on inputs, outputs and parameters including ranges/enum types etc.). The operation/numbering of arguments is covered in the info text.

You can also use any combination of the following to customise the info given:

- description (include the description)
- inputs (include info on the inputs)
- outputs (include info on the outputs)
- io (include info on inputs and outputs)
- parameters (include info on the objects parameters)
- quick (give brief versions of all info)

4 Alpha Aspects of FrameLib

- The object set is subject to change.
- All object features are subject to change.
- All object parameter names are subject to change.
  - here input on preferred formats (with or without underscores/capitals etc. would be helpful).
- Notation for parameters / inputs are subject to change.
  - here input on preferences is appreciated.
- The text displayed by the info object is subject to change.
  - here input on clarity / wording is appreciated.
  - input on what should be considered quick / comprehensive would be useful.