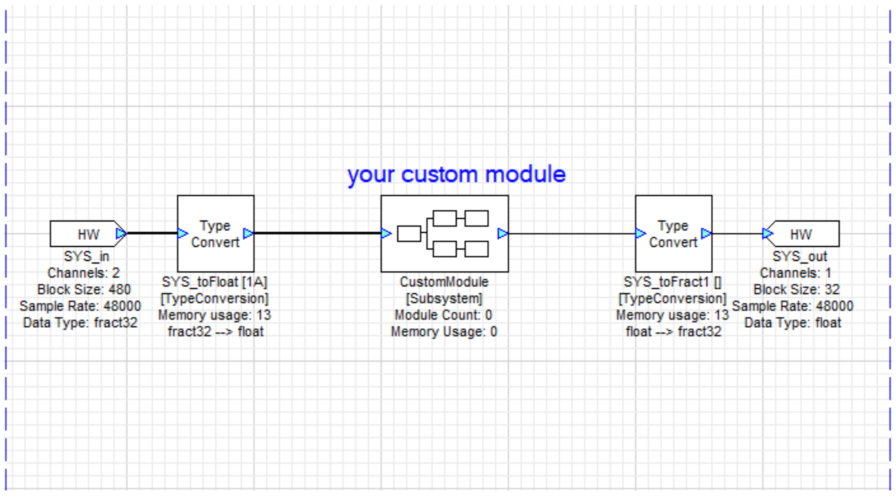
Interpreted Modules



Copyright Information

© 2022 DSP Concepts, Inc., ALL RIGHTS RESERVED. This document may not be reproduced in any form without prior, express written consent from DSP Concepts, Inc.

Printed in the USA.

Disclaimer

DSP Concepts, Inc. reserves the right to change this product without prior notice. Information furnished by DSP Concepts is believed to be accurate and reliable. However, no responsibility is assumed by DSP Concepts for its use; nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under the patent rights of DSP Concepts, Inc.

Change Log

Version	**Date**	**Description**
1.0	22 Dec. 2022	Initial Draft or major changes

Table of Contents

List of Figures

1 About This Guide

The interpreted modules guide contains instructions for enabling a custom module to be imported into Standard and Pro versions of Audio Weaver and found in the 3rd party tab of the module browser.

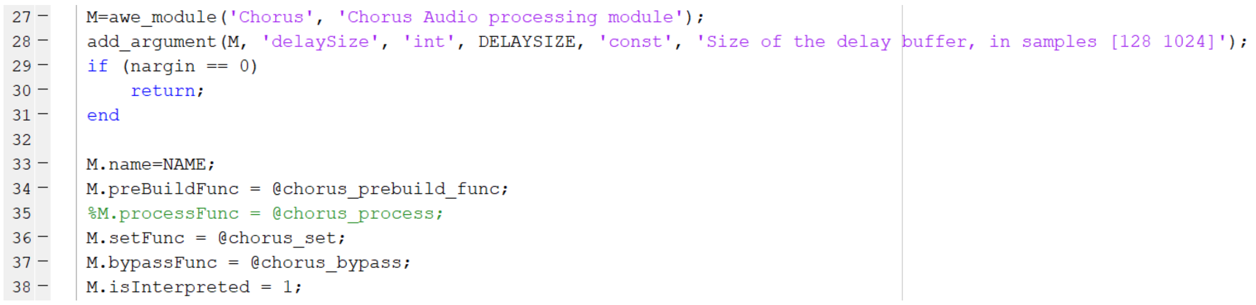
2 Steps for enabling custom module as “interpreted”

2.1 MATLAB

Do the following steps to edit the MATLAB files of the module

1. Add the line "isinterpreted = 1;" to the *<modulename>_module.m* file (found in the “matlab” folder of your custom module) anywhere after the “M = awe_module(…” assignment in the “matlab/” directory of your custom module.

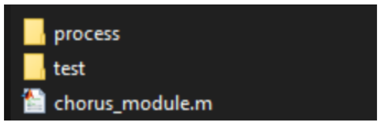
2. Add spaces around equals signs for assignment functions (see lines 34-37 in Figure 1)



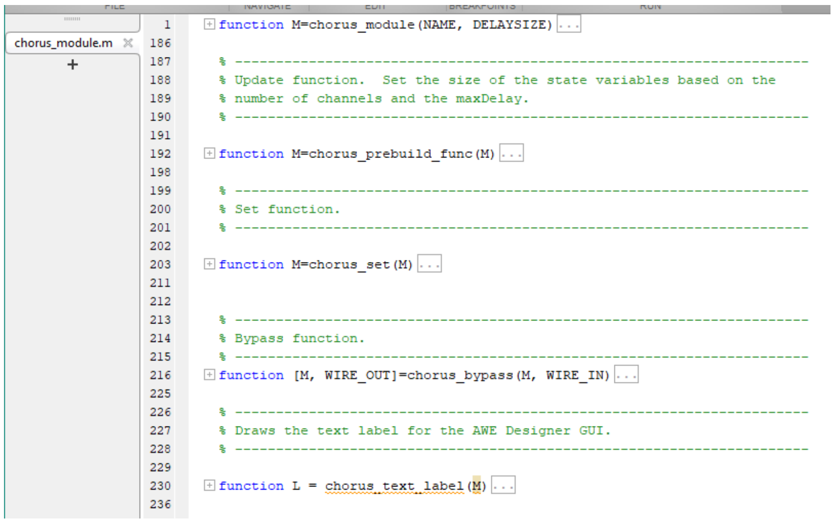
```
27 - M=awe_module('Chorus', 'Chorus Audio processing module');
28 - add_argument(M, 'delaySize', 'int', DELAYSIZE, 'const', 'Size of the delay buffer, in samples [128 1024]');
29 - if (nargin == 0)
30 -     return;
31 - end
32
33 - M.name=NAME;
34 - M.preBuildFunc = @chorus_prebuild_func;
35   %M.processFunc = @chorus_process;
36 - M.setFunc = @chorus_set;
37 - M.bypassFunc = @chorus_bypass;
38 - M.isInterpreted = 1;
```

Figure 1 - see line 38

3. Separate out functions contained in the *<modulename>_module.m* file to individual *.m* files.

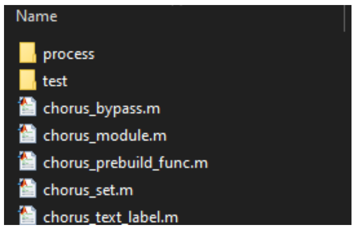


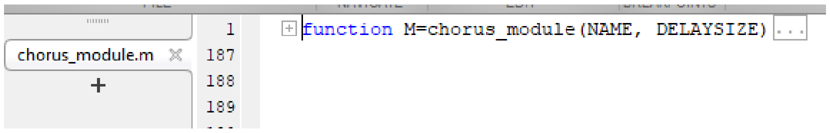
```
process
test
chorus_module.m
```



```
function M=chorus_module(NAME, DELAYSIZE) ...

  % -----------------------------------------------------------------------
  % Update function.  Set the size of the state variables based on the
  % number of channels and the maxDelay.
  % -----------------------------------------------------------------------

function M=chorus_prebuild_func(M) ...

  % -----------------------------------------------------------------------
  % Set function.
  % -----------------------------------------------------------------------

function M=chorus_set(M) ...


  % -----------------------------------------------------------------------
  % Bypass function.
  % -----------------------------------------------------------------------
function [M, WIRE_OUT]=chorus_bypass(M, WIRE_IN) ...

  % -----------------------------------------------------------------------
  % Draws the text label for the AWE Designer GUI.
  % -----------------------------------------------------------------------

function L = chorus_text_label(M) ...
```

Figure 2 - Before separating functions into unique .m files (Above, Right)

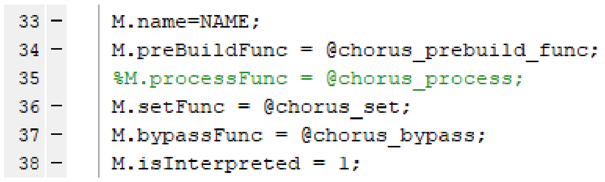




```
function M=chorus_module(NAME, DELAYSIZE) ...
```

Figure 3- After separating functions into unique .m files (Above)

4. Confirm there is a corresponding .m file for any function being defined, otherwise comment out the definition. For example, this module was failing because there was no corresponding "chorus_process.m" file. Commenting out line 35 fixed the issue.



```
33 -   M.name=NAME;
34 -   M.preBuildFunc = @chorus_prebuild_func;
35     %M.processFunc = @chorus_process;
36 -   M.setFunc = @chorus_set;
37 -   M.bypassFunc = @chorus_bypass;
38 -   M.isInterpreted = 1;
```

Figure 4 - Comment out any unused function definitions

5. Move any module function up to the same folder as the *<modulename>_module.m*

6. Run "make_<modulename>_pack(1)" to create the updated .c and .h files.

2.2 Building the module

Build the module, and before launching Designer, move or copy the created .dll file into the same directory containing *AWE_Server.exe* (typically located in *<install directory>/Bin/win32-vc142-rel/)*

2.3 Set Module Path in Designer

In Designer Standard, go *File-> Set Module Path*, then click *Add Folder,* select your custom module folder, and click *Select Folder*. Your module should now show in the modules tab and load into Designer.

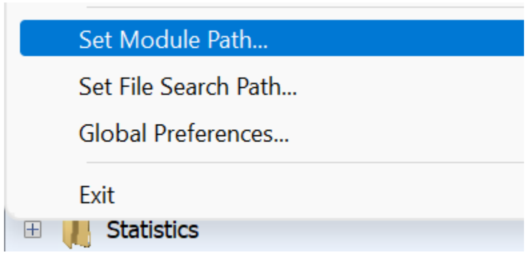
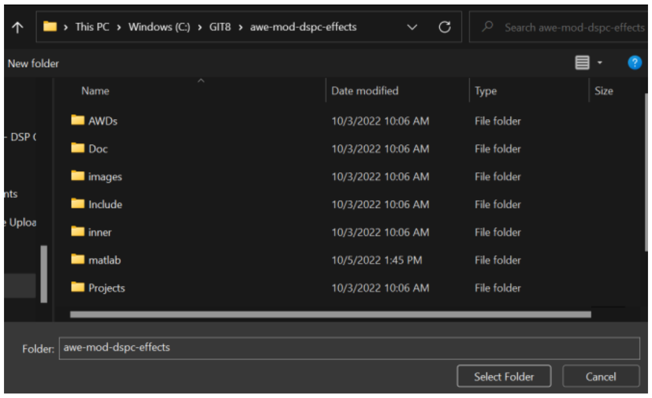


Figure 5- Add Module Path


This PC › Windows (C:) › GIT8 › awe-mod-dspc-effects
Search awe-mod-dspc-effects
New folder
Select Folder
Cancel

Name	Date modified	Type	Size
AWDs	10/3/2022 10:06 AM	File folder	
Doc	10/3/2022 10:06 AM	File folder	
images	10/3/2022 10:06 AM	File folder	
Include	10/3/2022 10:06 AM	File folder	
inner	10/3/2022 10:06 AM	File folder	
matlab	10/5/2022 1:45 PM	File folder	
Projects	10/3/2022 10:06 AM	File folder	

Folder: awe-mod-dspc-effects

Your module should now show in the modules tab and load into Designer.