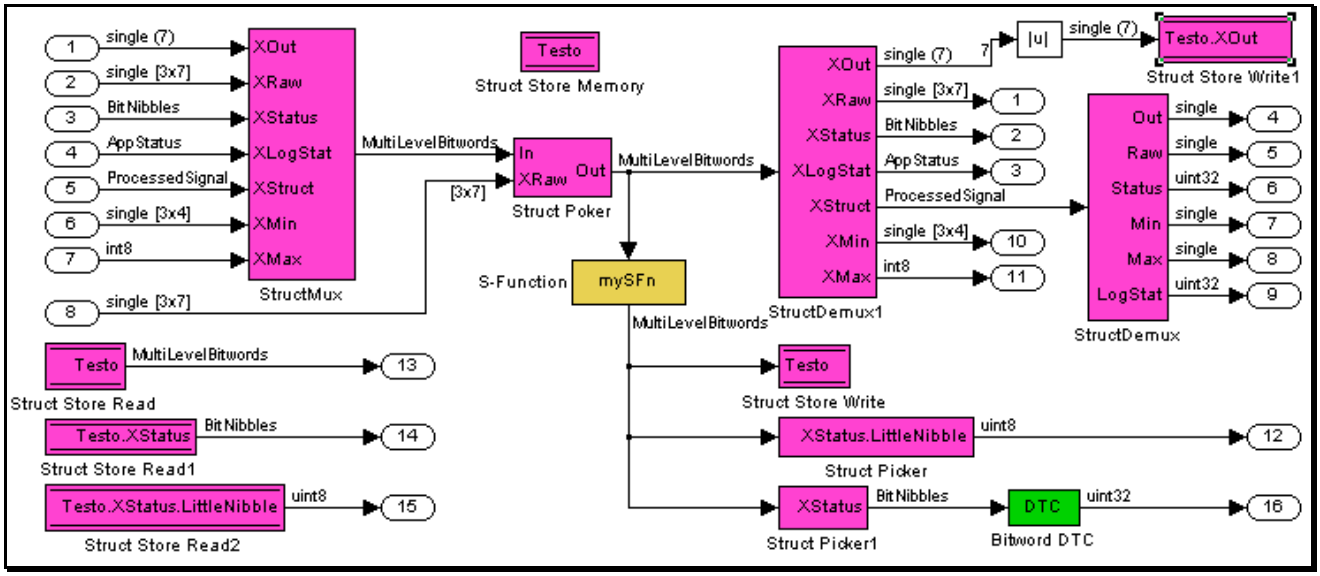
StructureLib

Enabling Use of
Data Structures and Bitwords
In Simulink®

Product Specification



MathWorks Partner

Podium Technology

October 2017

Overview

StructureLib extends Simulink® by providing a means of passing data structures around the model and methods for accessing and setting individual members of the structure.

The need for structures in Simulink usually comes about either for convenience of use when writing your own S-Functions or it becomes forced when structures exist in legacy code that needs to be interfaced to with Simulink modules.

When creating your own S-Functions, it is often more convenient to pass out a structure on one port that to have 20 ports for each of the items to be passed out (which may become 21 in the next version requiring all models to be updated). Once the structure is in the model it is easy to pick off the individual structure members or *demultiplex* the whole structure to its components.

Often legacy code that you are trying to interface Simulink modules to, store their data and parameters in data structures. Rather than rewriting the legacy code to not use structures or writing interface modules, it is now possible to read and write to these structures in Simulink without any need for modification.

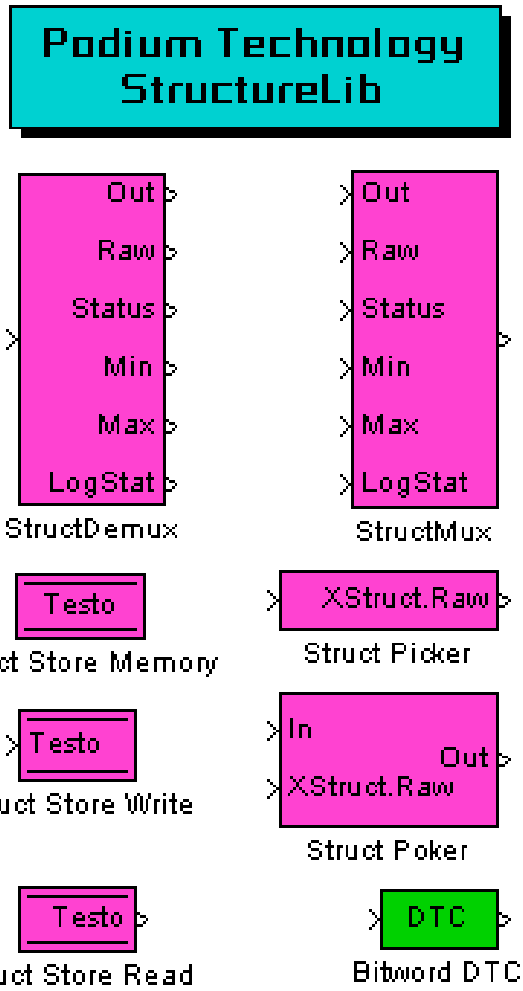
StructureLib supports structures nested to any depth and supports bitword structures (unsigned integer made up of individual bits or collections of bits or *nibbles*). Each member of a structure can be a scalar, vector, matrix or another structure or bitword.

StructureLib implements the key philosophy of a single source definition for the structure, therefore the structure can be defined in M-scripts from which a 'C' Header file is produced with the structure typedefs in. Alternatively, existing 'C' header files can be parsed to produce the necessary M-scripts to describe the structure to Simulink.

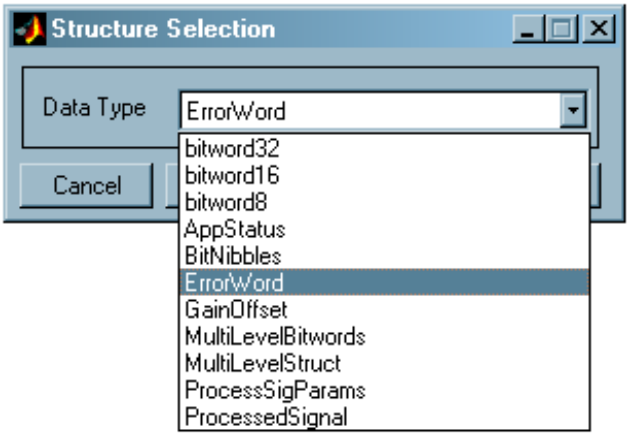
Full block and library documentation is supplied and accessed via the MATLAB® helpdesk program.

Block Set

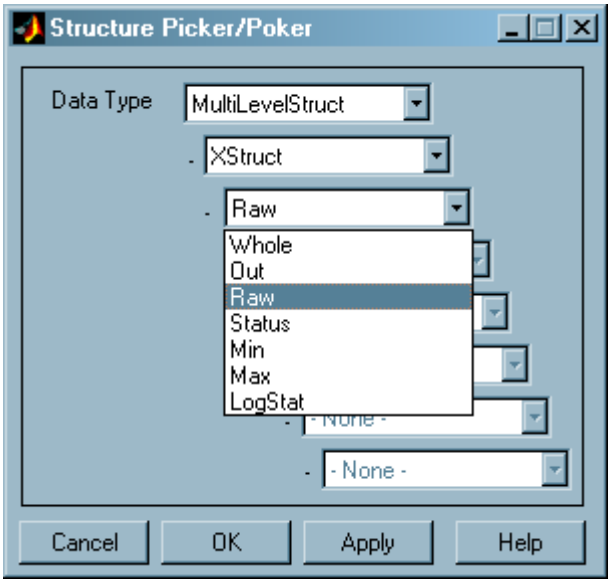
The blocks provided with **StructureLib** are shown on the right, each of the blocks is detailed below:



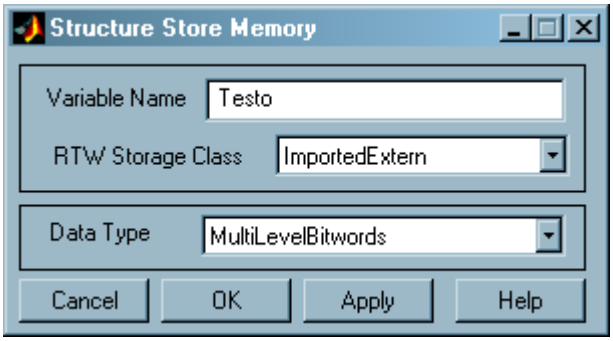
- **Struct Mux & Demux:** These two blocks form the methods for constructing and deconstructing the structures to and from their component parts.



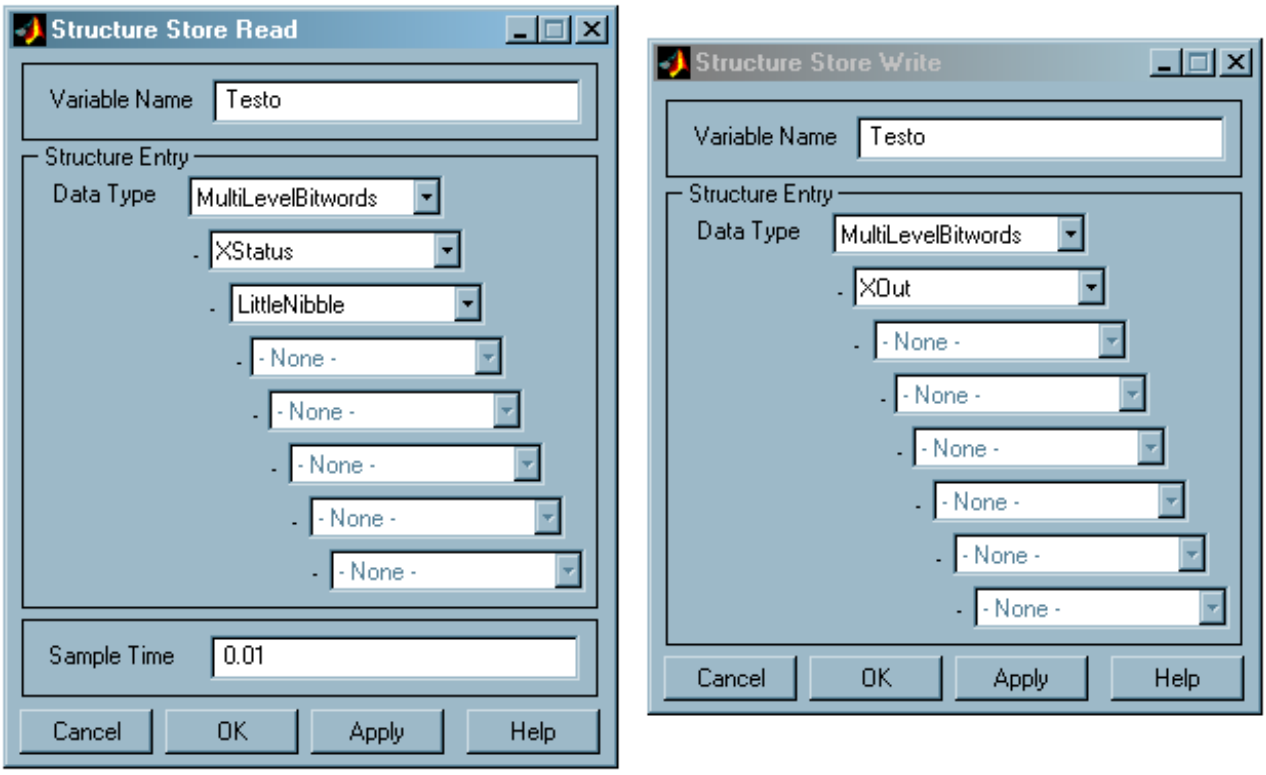
- **StructPicker and Poker:** These enable individual members of a structure to be read from or inserted into structures being passed around the model. The members can be selected at any point in the nesting of structures, therefore a sub-structure can be selected and operated on as a whole, not just the *end points*.



- **Struct Store Memory:** Defines a memory store for a whole structure. The store is given a name and a structure type, it also defines how it should be declared should it be used in a Real-Time Workshop application (either exported or imported).

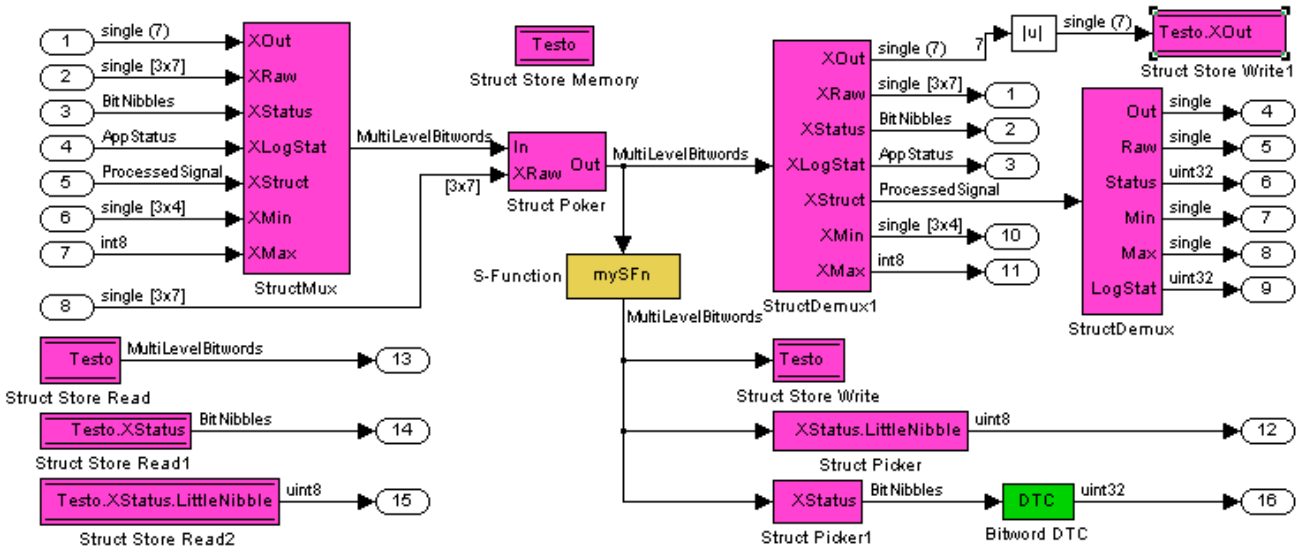


- **Struct Store Read and Write:** Blocks used for reading and writing to the structure memory store. Any point of the structure can be accessed from the whole structure to sub structures to members or bits/nibbles of bitwords.



- **Bitword DTC:** When bitword data types need to be turned into standard Simulink types or fed from standard Simulink types then the *Bitword Data Type Conversion* block is used.

An example of how all the blocks could be used together is shown below:



System Structure Blocks

Structures are defined in one of two ways; either by describing them in an M-script or parsing a 'C' header file to produce the description, i.e. there should only be one root definition to avoid conflicts.

M-Script Method

An example of a definition file is:

D:\Dev\StructureLib\basedefn\TYPEDEF_MultiLevelStruct.m *

```
function thisStruct = MultiLevelStruct()
% returns the structure definition
%
persistent ret

if isempty(ret)

    % Add an element as a key that this is a structure
    ret.IsAtSourceStruct = 1;
    ret.IsBitword = 0;
    ret.NumBits = 0;
    % Name and Description
    ret.Name = 'MultiLevelStruct';
    ret.Desc = 'Multi Level Structure example';
    % Member elements
    ret.Elements = ...
        { ...
            struct('Element', 'XOut'    , 'DataType', 'single'          , 'Dimensions', [1 7]  ), ...
            struct('Element', 'XRaw'    , 'DataType', 'single'          , 'Dimensions', [3 7]  ), ...
            struct('Element', 'XStatus' , 'DataType', 'uint32'          , 'Dimensions', [7]    ), ...
            struct('Element', 'XLogStat', 'DataType', 'uint32'          , 'Dimensions', [1]    ), ...
            struct('Element', 'XStruct' , 'DataType', 'ProcessedSignal' , 'Dimensions', []     ), ...
            struct('Element', 'XMin'    , 'DataType', 'single'          , 'Dimensions', [3 4]  ), ...
            struct('Element', 'XMax'    , 'DataType', 'int8'                                   ), ...
        };
end

thisStruct = ret;
```

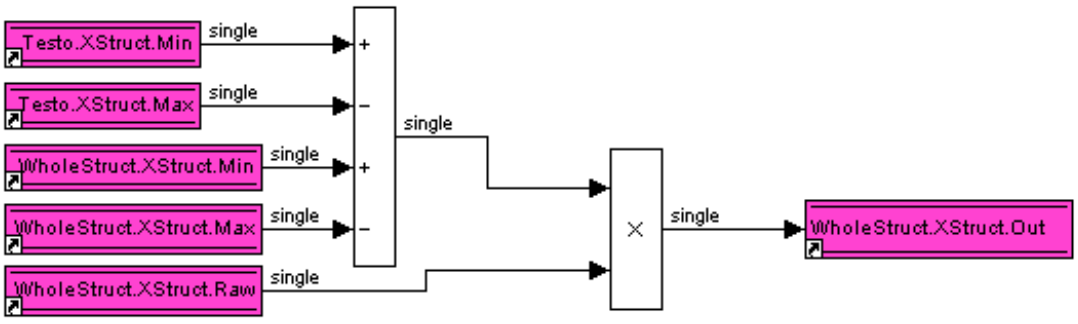
Each of the structure members or elements just needs to be given a name and a data type, this can optionally be supplemented with dimensions and number of bits, if it is a bitword type.

Header File Parser Method

The above description files can be created automatically from 'C' header files using a parsing utility that is provided with **StructureLib**.

Real-Time Workshop Support

All the blocks in the library are also supplied with TLC files to produce efficient in-line 'C' code during a build process with Real-Time Workshop. For example, the following model extract:



Procuces the following code:

```
WholeStruct.XStruct.Out = ((Testo.XStruct.Min - Testo.XStruct.Max +
  WholeStruct.XStruct.Min - WholeStruct.XStruct.Max) * WholeStruct.XStruct.Raw);
```

System Requirements:

StructureLib has been tested with MATLAB®[i] R2011b onwards (32-bit and 64-bit), no special toolboxes are required in order for it to be used with Simulink®.

StructureLib will produce 'C' code for any Real-Time Workshop® target.

Podium Technology is part of The MathWorks Connections Program to ensure that **StructureLib** is available when new releases are made by The MathWorks.

Related Products:

Podium Technology produces **@Source** from which **StructureLib** was derived. **@Source** is the ideal tool for generating a target independent model that can be applied to standard or customer target hardware platforms. Signal and parameter properties are stored in the model so that application tool files can be produced in any format (e.g. ASAM-MCD2) and documentation to be produced with the MATLAB/Simulink Report Generator. See our web site or contact us for further information.

Design Service:

Podium Technology can has experience in a wide range of areas, with particular focus on the MATLAB/Simulink, control and embedded arenas:

- Rapid generation of Real-Time Workshop targets for customer hardware platforms using **@Source.**
- Hardware specific block sets for Simulink and Real-Time Workshop.
- Developing control systems in 'C' or Simulink
- MATLAB/Simulink Report Generator component generation.
- Porting legacy models to **@Source**.

Contact us:

Address	**Podium Technology Ltd.**[ii] Sprytown Lifton Devon PL16 0AY
Telephone	+44 1566 784508
E-Mail	enquiries@podiumtechnology.co.uk
WWW	www.podiumtechnology.co.uk

[i] MATLAB, Simulink, Real-Time Workshop and Embedded Coder are registered trademarks of The MathWorks, Inc.

[ii] Podium Technology Ltd. is registered in England and Wales No. 4536934