**SimAssist Guideline**

**Video Tutorial:**

<http://list.youku.com/albumlist/show?id=27863410&ascending=1&page=1>

**Getting Started:**

The plug-in tool SimAssist is designed to facilitate modeling using MATLAB/Simulink/Stateflow, it provides various built-in functionalities that relieve the user from repetitive and tedious operations during modeling. The ultimate target of this facility is to get ride of those algorithm-irrelevant operations as far as possible thus help software modeling engineers to focus mainly on how to realizing the control algorithm.

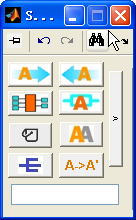
This tool is highly customizable by its infrastructure design.

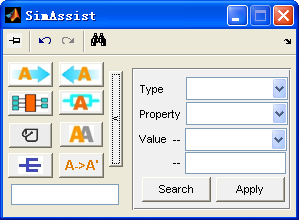
* Functionalities are packed in seperate macro registration files.
* Behavior and features of block types are described in seperate block registration files.
* The registration files are scanned on every start, they are plug-and-play.

In this way the supported functionalities and blocks can be easily tailored or customized without affecting others.

*Note: Basically every functionality is defined as a "Macro", which is essentially a script that perform a seriesof operations programatically.*

The tools is interfaced by a GUI as the following figure (Standard/Expanded):





Of all the functionalities those may be used most frequently are linked with buttons on the GUI, while others can be invoked by executing brief commands in the bottom edit box.

**Detail explanation of usage:**

**Button:**

https://note.youdao.com/yws/public/resource/a417ec0f08651694fb31928168bb010c/xmlnote/0CEE461147EC4605A441538278CCC2F0/10911

* **Propagate Upstream/**

https://note.youdao.com/yws/public/resource/a417ec0f08651694fb31928168bb010c/xmlnote/A36A0CD071614FEBA05A27A0A639D0DC/10910

**Propagate Downstream：**

* + Propagate string from upstream or downstream.
    - Propagate major string upstream or downstream to block
    - When multi-property blocks (e.g., Saturation, Lookup table, etc.) encounterd, format the string automatically. The way to format the string can be easily customized.
    - For BusCreator, this functionality trace downstream of the model to collect all the signals orginated from this block and generate appropriate number of inports as well as adding lines with corresponding name. This feature could be very useful when user have to create bus harness as inputs to a complex bus-input-based model (e.g., for purpose of unit test).
    - For BusSelector, if propagated from downstream, the signals selected will be changed accordingly, regardless of whether the inport is feed with a appropriate bus. If propagated from upstream, all the signals will be recursively selected as output. This can be useful when the input bus is multi-leveled, and you want to select them all out.
    - Propagation to Stateflow chart is also supported, corresponding input/output data objects will be generated in this condition
  + Create counterpart of block
    - If blocks like DataStoreRead/Write, From/Goto are currently selected, the propagate button may be used in inverse direction to generate the counterpart block, i.e., From<->Goto, DataStoreRead<->DataStoreWrite

https://note.youdao.com/yws/public/resource/a417ec0f08651694fb31928168bb010c/xmlnote/DAC0172352004641A5B589F64191EBF6/10912

* **Auto Align**
  + Layout around center block
    - If **only one** block selected, this functionality tries to layout the blocks connected on both sides, exclusively involved branches will also be shifted
  + Layout blocks
    - If more than one blocks selected, internally this functionality tries to align them horizontally with ports, and vertically by block types.
    - Also, externally blocks are grouped as a whole to align with ports
  + Resize on right mouse button click
    - If click right mouse on the button, the functionality tries to resize selected blocks vertically to adapt to its surrounding blocks
  + Straighten line
    - If line selected, the functionality tries to straighten the line together with its connected blocks

https://note.youdao.com/yws/public/resource/a417ec0f08651694fb31928168bb010c/xmlnote/B01DFE94B3744C87A2B45FF9E610B6F9/10913

* **Smart Action**

            This button combines several features into one, perform different actions as per selected block type

* + Annotate block
    - Add annotation to block in predefined pattern, e.g., display datatype, major block parameters, etc.
  + Refine block
    - Standardize block parameter setting according to certain modeling guideline
    - Rename block name in accordance with major block parameter
  + Roll property
    - Roll block major property if selected block have an enumeration major property, e.g.,  Compare, Logic, Math, MinMax, ...

https://note.youdao.com/yws/public/resource/a417ec0f08651694fb31928168bb010c/xmlnote/D25CB0F6AB184BE3907BCC652D948F3F/10921



https://note.youdao.com/yws/public/resource/a417ec0f08651694fb31928168bb010c/xmlnote/ACBB57C91BAD463183E39D5321D2A665/10915

**Format brush**

This button acts just like the format brush button in Microsoft Office, except that it copy parameter settings from source block and then brush it to the destination blocks

* + With single source block selected and click the "empty"

https://note.youdao.com/yws/public/resource/a417ec0f08651694fb31928168bb010c/xmlnote/F77CC36BC1524279AB9B1FB2F0FB285A/10919

 button will set the source, and the button state changes to "filled"

https://note.youdao.com/yws/public/resource/a417ec0f08651694fb31928168bb010c/xmlnote/9B2C7752A4974DC182B7CA1A7A6A9B60/10917

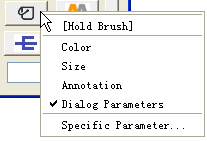
* + Selecte blocks to be brushed, and click the

https://note.youdao.com/yws/public/resource/a417ec0f08651694fb31928168bb010c/xmlnote/5C5D9EB94D0349DF805A750B858F0414/10916

 button again, parameters that both source and destinations have will be copied from source to destination and the button reset to empty state

https://note.youdao.com/yws/public/resource/a417ec0f08651694fb31928168bb010c/xmlnote/88A2932D3A2C43438546D6B812416F69/10920

* + Right click on the button will invoke the menu



* + - "Color", "Size", "Annotation" and "Dialog Properties" are options for user to select the target properties to be brushed. Items that are not checked will not be brushed.
    - Use "Specify Parameter..." option to specify effective source parameters if you don't want all parameters brushed to destination
    - Use "[Hold Brush]" button to keep the brush filled without reset  to empty after brushing (default behavior), so as for multiple usage

https://note.youdao.com/yws/public/resource/a417ec0f08651694fb31928168bb010c/xmlnote/AF6F8F773B534B0397281626D6236185/22540

* + **Multi ->Multi mode:**  If multiple blocks are selected on copy action, the major property of these block will be saved and later brushed on destination blocks. This can be very useful to simultaneously transfer properties between a bunch of blocks when they cannot be connected  (From -> Inport, for example). Note that lines are also supported as long as selected.

https://note.youdao.com/yws/public/resource/a417ec0f08651694fb31928168bb010c/xmlnote/B5A7448ADEC344AEA4FE9A22F634DA81/10914

* **Show/Hide name**
  + Show/Hide name of selected block

https://note.youdao.com/yws/public/resource/a417ec0f08651694fb31928168bb010c/xmlnote/4A08095554EE48D5B7D91692352A00FC/10918

* **Set signal object resolve**
  + Set the lines selected as "must resolve to signal object"

https://note.youdao.com/yws/public/resource/a417ec0f08651694fb31928168bb010c/xmlnote/5E4593E075D54AB5B4584B14F91093D7/10922

* **Standardize naming**
  + With blocks or lines selected, this functionality tries to standardize its abbreviation naming according to the dictionary
  + Dictionary is defined in SACFG\_DICTIONARY.m file, modify this file to comply with customized modeling naming rule
  + Right click on the button will invoke the menu to enter Stateflow mode. In this mode only the objects currently selected in the Stateflow editor will be renamed, otherwise the entire stateflow chart will be renamed throughly

**Batch property setting**



This function block aims to facilitate operation on a bunch of certain blocks, it provides a versatile way to check, modify and synchronize certain properties.

When “Search” executed, the function collects currently selected blocks and lists all the types and associated properties as well as the values in corresponding fields.

When “Apply” executed, the function applies on blocks that currently selected and filtered by type specified in “Type”. The specified “Property” of these blocks will be set to the specified "Value". The text string in the popup menu will be used as specified value if  the above text field is empty, otherwise the string in text field will be used.

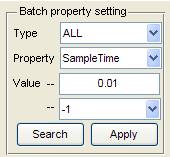
Note that whenever “Apply” is clicked, the function implements only on the currently selected blocks, which allows you to change selection range as needed at any time.

Also note the input Edit allows multiple expression or sequential expression, like "Kate, Jack, Mary" or "Val[1:5]", etc. It would be very convenient to use this feature in case there are multiple blocks with which you want to set their properties in series. Refer to the "Sequential Expression" part for detail usage.

**Example 1:**

Suppose you want to check and synchronize the sample time of all blocks under current system to “0.01”, which could be realized in the following steps:

1. Ctrl+A to select all the objects under current system
2. Click on “Search”, and select “All” in popup menu “Type”, select “SampleTime” from popup menu “Property”
3. If there is “0.01” in popup menu “Value” (which means there is at least one block whose sample time is 0.01), select and click on “Apply”, otherwise manually input “0.01” in the text field and click on “Apply”.



**Example 2:**

Suppose you want to check if all the “Switch” blocks has been set to “u2~=0”, and make change if not:

1. Ctrl+A to select all the objects under current system
2. Click on “Search”, and select “Switch” in popup menu “Type”, select “Criteria” from popup menu “Property”
3. Select “u2~=0” from popup menu “Value”, and apply

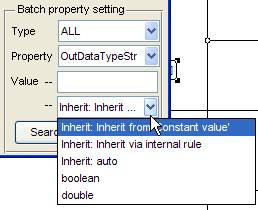


**Example 3:**

Suppose you want to check no “Inherit” data type has been used on some blocks:

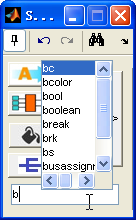
1. Select the blocks to check
2. Select property “OutDataTypeStr”
3. Check value list in popup menu “Value” to see if “Inherit” data type exists

The image below shows that there are several output data types within the selected blocks.



**Command line:**

    Due to the limitation on GUI size, most other functions are invoked in command line. Command patterns can easily be added, removed or changed as necessary, as well as their priorities.



 A prompt list will show according to the letters already entered.

**Command line patterns:**

In most cases, the command line adopts the pattern that a starting keyword ("cmd") and some optional suffix strings.

* General command patterns
  + cmd[option]
    - If suffix option is specified after the starting keyword, the string will be used to set the block major property, for example,
      * cnst5.0 creates a Constant block with name set to 5.0
      * fromabc creates a From block with tag set to "abc"
    - For certain commands the option is intepreted as a second operand, for example,
      * \*5 creates a Multiply block as well as a Constant block with value 5 connected to its second port
      * ==5 creates a "==" RelationOperator block as well as a Constant block with value 5 connected to its second port
  + cmd[N]
    - For certain blocks, if suffix option is specified as an integer, it may be used to control the number of ports, for example,
      * bs6 creates a BusSelector with 6 outports
      * bc6 creates a BusCreator with 6 inports
      * min6 creates a Min block with 6 inports
  + cmd[N][option1][,option2][,...]
    - For certain blocks, suffix integer N is used to specify number of blocks to be created, and option is used to set the block major property, for example
      * "from5" creates five from blocks
      * "from5AAA BBB" creates five blocks and tagged correspondingly (use last option if not given enough number of options)
  + cmd[option1][,option2][,...]
    - For certain blocks with multiple major properties, e.g., Saturation, Lookup table, it is also possible to specify its parameter after the keyword. Multiple parameters can be seperated using whitespace or comma, for example,
      * l2ABC will creates a Lookup2D with Row, Column, and Table set as ABC\_X, ABC\_Y, ABC (note that the naming rules can be customized according to customer requirements)
      * "l2 AX BY CZ" or "l2AX,BY,CZ", they all creates a Lookup2D block with Row, Column, Table set as AX, BY, CZ
      * **"fromAAA BBB CCC" or "fromAAA,BBB,CCC" creates three blocks each tagged respectively**
      * **"cnst 5.0, ABC, -1" creates three Constant blocks each set with respective value**
  + cmd[options]
    - More complex options is possible for some commands, for example,
      * "ss i3 o3 atom" creates an Atomic SubSystem with 3 inports and 3 outports
  + Plain text with no command pattern match
    - Depending on whether there is any block currently selected:
      * if true, the text will be tried prefixed with a backlash "\", which tries to modify the major property of currently selection
      * if false, the command will be used like prefixed with "cnst", which tries to create Constant block(s) using the command string as Value
    - This feature might be quite frequently used to save typing effort, for example,
      * Provided no current selection, if the given command string is parsed without matching, it will be prefixed with "cnst" keyword automatically, for example,
        + "true" creates a Constant block with value "true"
        + "5.0" creates a Constant block with value "5.0"
        + "ABC" creates a Constant block with value "ABC" (suppose "ABC" is not a keyword)
      * Provided current selection exists, if given command string is parsed without matching, it will be prefixed with "\", for example
        + Select five "Inport" blocks and run command "aa,bb,cc,dd,ee", changes their names accordingly
        + Select five "Goto" blocks and run command "var[1:5]", changes their tags to "var1, var2, var3, var4, var5" accordingly
* Plus/Minus method
  + For each block selected, it can handle "+/-" commands if Plus/Minus methods are defined for that block type, for example,
    - +5 on a BusCreator block increaes number of inports by five
    - ++ on a BusSelector block select all signals recursively out from the input bus
    - + i5fc on a SubSystem block adds five inports and an Enable trigger port to it
  + Note that if there is no block selected, the +/- operator will be intepreted as adding a new Sum block
* Other command patterns

        There are also other less common command patterns, as are explained later in the Macros section, e.g.,

* + \ABC direct change of major property
  + ABC=>BCD string replace
  + $ system commands
* Hotkeys
  + For convenience come commands are associated with hotkey when the command line editbox is active.
    - Ctrl key is equivalent to "line" command
    - Alt+Arrow is equivalent to "wider", "narrower", "longer", "shorter" commands, they can be useful in adjusting size of multiple blocks at the same time
* Sequential expression
  + Sequential expression might be very useful when you want to create/change a series of properties that follows certain sequential rule
  + Usage pattern: series of expression enclosed in bracket pair like [expr1, expr2, ... ...]
  + The following sequence expression is allowed ("#" indicates number(s), and "A", "a", "Z", "z" indicates alphabet letters)
    - #:#, #:#:#
    - A:#:Z, a:#:z
    - "STR1, STR2, STR3", "STR1 STR2 STR3"
    - mix of the above patterns seperated with whitespace or comma
  + If mulitiple sequential expression exists, the shorter ones will be repeated to align with longest one, see the example.
  + Examples of the sequence expression part:
    - "PRE[1:10]TAIL" translates to sequence like {PRE1TAIL, PRE2TAIL, ...}
    - "PRE[1:2:10,AA,BB,CC]TAIL" translates to sequence like {PRE1TAIL, PRE3TAIL, ..., PRE9TAIL, ..., PREAATAIL, PREBBTAIL, PRECCTAIL}
    - "PRE[b:e]TAIL translates to sequence like {PREbTAIL, PREcTAIL, PREdTAIL, PREeTAIL}
    - "PRE[a:3:z]" translates to sequence like {PREa, PREd, PREg, ...}
    - "PRE[AA,BB,CC,A:3:Z]TAIL" translates to sequence like {PREAATAIL, PREBBTAIL, PRECCTAIL, PREATAIL, PREDTAIL, ...}
    - "PRE[1:5]MID[a:k]END" translates to sequence like {PRE1MIDaEND, PRE2MIDbEND, PRE3MIDcEND, PRE4MIDdEND, PRE5MIDeEND, PRE1MIDfEND, PRE2MIDgEND, PRE3MIDhEND, PRE4MIDiEND, PRE5MIDjEND, PRE1MIDkEND}
  + Examples of command usage
    - "ipTest[1:10]"creates ten inport named as "Test1, Test2, ..."
    - Select a bunch of SubSystems, say you want to name them using their schedule time, and "sch\_[10, 20, 50, 100]ms", changes their names to "sch\_10ms", "sch\_20ms", ...
    - Suppose you have several signals in From blocks to merge, you can utilize sequential expression to name them in one line, like "sig\_[idle, pwrup, pwrdn, run, afterrun]" tags the blocks like "sig\_idle", "sig\_pwrup", ...
* Insignificance of whitespace and order of options
  + In most cases, as long as there is no ambiguous, whitespace is insignificant in idnetifying command segments. Order of options (if any) is also insignificant. This relieves the user from remebering too many rules, for example:
    - "ipabc" is equivalent to "ip abc", they all creates an Inport block named as "abc"
    - "ssi5o6en" is equivalent to "ss en i5 o6", they all creates an Enable SubSystem with five inports and six outports
    - "and4" is equivalent to "and 4", they all creates a Logical AND block with four inputs
* Multiple commands in single run
  + Use "&" symbol to join several commands together in one session, for example,
    - "from & goto" is equivalent to execute "from" and then execute "goto"
  + Use "\*N" suffix with command to execute the command N times
    - This can be very useful to create several blocks in a batch
    - "fromabc\*5" is equivalent to execute "fromabc" five times, except the block will be automatically offset to avoid overlap
* Use clipboard for multiple inputs
  + The keyword "#cb", "#clip", "clipboard" in command will be replaced by actual string currently in clipboard (processed appropriately)
    - This can be extremely useful when modeling from documented requirement
    - "ip#cb" creates 5 inport and named correspondingly according to the clipboard content (assuming that clipboard with content "aaa bbb ccc ddd eee")

**Scenario specific behaviors of command line:**

There are some scenario specific behaviors that most blocks adopt:

* Auto number of inport/outport
  + For blocks like BusCreator, BusSelector, Merge, MinMax, etc. if command given with no option suffix, the number of inputs or outputs will be determined according to current selection of block or lines.
  + More specifically, number of inputs is determined according to number of unconnected outports and lines selected
  + Number of outputs is determined according to number of unconnected inports currently selected.
* Auto grounds/terminates with block
  + For sink-like blocks, e.g., Outport, Goto, etc., if command given with no option suffix, the currently selected lines or unconnected outports will be terminated with this type of block and signal name automatically propagated
  + For source-like blocks, e.g., Inport, From, etc., if command given with no option suffix, the currently selected unconnected inports and no source lines will be grounded with this type of block and signal name automatically propagated
  + For blocks that can be connected on both sides, if side option specified ("[" and "]", refer to the Global Session Option part), it will act as a sink or source block
  + The block terminates on lines can be positioned using mouse (use +/-ms option to enable/disable this feature)
* Auto insert block to line
  + For blocks that can be connected on both sides, if command given with no option suffix, the currently selected lines will be automatically inserted with this type of block, e.g., DataTypeConversion, RateTransition, etc.
* Auto line connection
  + When new blocks added with blocks or lines already selected, this tool will automatically add line to connect the already selected block/line with the newly added block
* Mouse position to assist location
  + Specify postion when adding new block
  + Specify space when adding new block
  + Use +ms/-ms option to alter the default behavior

**Global session options:**

There are some global options that can affect the command behaviour. These options are pre-processed and removed from the command string before it is actually executed, thus it can be put anywhere of the command string.

* :*DATATYPE* : Specifies data type to be set
  + The DATATYPE field can be one of the following expression, single, sgl, double, dbl, boolean, bool, uint8, u8, uint16, u16, uint32, u32, int8, i8, int16, i16, int32, i32
  + For example,
    - "cnst ABC:sgl" creates a Constant block and sets its output data type to single
    - "dt:bool" creates a DataTypeConversion block and sets its output data type to boolean
  + Note that this option status will be preserved for later session, i.e., you don't have to explicitly add this option if you want to use the same data type for next command run
  + Suffix the colon with nothing resets the data type to "Inherit...", in other words, do nothing
* [ ] : Side specifier
  + It happens that the user want to specify that the command shall take effect only on inport side or outport side, which can be realized by adding "[" symbol for inport only or "]" symbol for outport only, for example,
  + "clean[" performs the clean action only on its inport side
  + "dt]" add DataTypeConverter only to the outport side of selected block
* +c/-c : Controls whether to set random color, by default OFF
  + For example, "from +c" creates a From block and set random foreground color
  + Note that this option status will be preserved for later session
* +a/-a : Controls whether to append annotation, by default OFF
  + For example, "cnst ABC+a" creates and annotates a Constant block
  + Note that this option status will be preserved for later session
* +dt/-dt : Controls enable or disable automatically setting data type, by default ON
  + For example, "cnst ABC-dt" creates a Constant block but disables the data type setting behaviour
  + Note that this option status will be preserved for later session
* +ms/-ms: Controls enable or disable using mouse position as reference when adding blocks, by default on
  + For example, "goto-ms" adds block to selected line using default layout parameter defined for the block (in registration file) instead of mouse position as vertical reference
  + For example, "ip-ms" adds block to selected empty input port using default layout parameter defined for the block (in registration file) instead of mouse position as horizontal space reference
  + Note that this option status will be preserved for later session

**Console command operator ($):**

The "$" operator is defined as console command, it can be used to operate on the SimAssist elementary operations, such as  list/add/remove or manage the blocks and macros supported, as well as import and export the database. So far the following commands are supported (functions yet to be refined and expanded):

* $+ , $add
  + Add new block type definition based on the current block selected
  + User can customize its pattern and behavior (routines) on the dialog box
* $-macro
  + List registered macros and user can select to remove them from database
* $-block
  + List registered blocks and user can select to remove them from database
* $cmdlist
  + List all the command patterns of macros in current database
* $saveas
  + Save the current database to mat file

**Detailed description of block specific behaviours:**

* Abs

|  |  |
| --- | --- |
| Keyword | abs |
| Examples | abs abs5 |
| Auto ground on selected ports/lines | Yes |
| Auto terminate on selected ports/lines | Yes |
| Insert in line | Yes |
| Multiple blocks at one time | absN, abs\*N |
| String at inport | pass through |
| String at outport | pass through |

* BusAssignment

|  |  |
| --- | --- |
| Keyword | busassignment |
| Examples | busassignment |
| Propagate from upstream | Set corrsponding bus signal |
| Propagate from downstream | Trace forward to collect all bus signals and set as inport assigned signals |
| String at inport | Inport 1: pass through from 1st outport Other inports: corresponding bus signal |
| String at outport | Pass through from 1st inport |
| Plus operation | Not yet supported |
| Minus operation | Not yet supported |
| Clean operation | Not yet supported |

* BusCreator

|  |  |
| --- | --- |
| Keyword | bc, buscreator |
| Examples | bc bc5 |
| Auto number of ports | Context dependent inport number |
| Propagate from downstream | Trace forward to collect all bus signals and set as input signals, a named line with no source may be generated at that port if necessary |
| String at inport | Corresponding downstream bus signals (indexed by port number) |
| String at outport | Block name |
| Plus operation | "+" add 1 inport "++" number of increase is determined according to unconnected ports currently selected "+N" increase number of inports by N |
| Minus operation | "--" reduce 1 inport "-N" reduce number of inports by N |
| Clean operation | Arrange and remove unconnected inports |

* BusSelector

|  |  |
| --- | --- |
| Keyword | bs, busselector |
| Examples | bs bs5 |
| Auto number of ports | Context dependent outport number |
| Propagate from upstream | Select all signals in the bus |
| Propagate from downstream | Set selected outport signal to corresponding downstream signals |
| String at inport | Block Name |
| String at outport | Corresponding selected signal |
| Plus operation | "+" add 1 outport "++" recursively select all signals from the incoming bus "+N" increase number of outports by N |
| Minus operation | "--" reduce 1 outport "-N" reduce number of outports by N |
| Clean operation | Arrange and remove unconnected outports |

* Concatenate

|  |  |
| --- | --- |
| Keyword | vecconcat, concat |
| Examples | concat concat5 |
| Auto number of ports | Context dependent inport number |
| Propagate from downstream | Trace forward to collect all bus signals and set as input signals, a named line with no source may be generated at that port if necessary |
| String at inport | Corresponding downstream bus signals (indexed by port number) |
| String at outport | Block name |
| Annotation | Current concatenate mode (Vector or Matrix), and dimension information |
| Plus operation | "+" add 1 inport "++" number of increase is determined according to unconnected ports currently selected "+N" increase number of inports by N |
| Minus operation | "--" reduce 1 inport "-N" reduce number of inports by N |

* Constant

|  |  |
| --- | --- |
| Keyword | cnst, const, constant |
| Examples | cnst 5.0 cnst5.0:sgl cnst5.0,cnst1,cnst2 true (if "true" doesn't match any existing keyword) VariableName:sgl (if "VariableName" doesn't match any existing keyword) |
| Auto ground on selected ports/lines | Yes |
| Multiple blocks at one time | Yes |
| Brother blocks | Constant blocks within the same subsystem that same in 'Value' |
| Propagate from downstream | set property 'Value' |
| String at outport | Property 'Value' |
| Annotation | Show datatype of the block |
| Refine operation | Rename the block to be consistent with 'Value' |

**Note: The keyword may be omitted if the intended 'Value' doesn't match any keyword**

* DataStoreMemory

|  |  |
| --- | --- |
| Keyword | dsm |
| Examples | dsm dsm5 dsmABCD dsmAAA,BBB,CCC |
| Major property/Property sequence | DataStoreName |
| Multiple blocks at one time | Yes |
| Brother blocks | DataStoreRead, DataStoreWrite |
| Annotation | Show datatype of the block |
| Refine operation | Rename the block to be consistent with 'DataStoreName' |
| Other scenario dependent behavior | Create corresponding block if brother blocks currently selected |

* DataStoreRead

|  |  |
| --- | --- |
| Keyword | dsr |
| Examples | dsr dsr5 dsrABCD dsrAAA,BBB,CCC |
| Auto ground on selected ports/lines | Yes |
| Major property/Property sequence | DataStoreName |
| Multiple blocks at one time | Yes |
| Brother blocks | DataStoreMemory, DataStoreWrite |
| Propagate from downstream | DataStoreName |
| String at outport | DataStoreName |
| Refine operation | Rename the block to be consistent with 'DataStoreName' |
| Other scenario dependent behavior | Create corresponding block if brother blocks currently selected |

* DataStoreWrite

|  |  |
| --- | --- |
| Keyword | dsw |
| Examples | dsw dsw5 dswABCD dswAAA,BBB,CCC |
| Auto terminate on selected ports/lines | Yes |
| Major property/Property sequence | DataStoreName |
| Multiple blocks at one time | Yes |
| Brother blocks | DataStoreMemory, DataStoreRead |
| Propagate from upstream | DataStoreName |
| String at inport | DataStoreName |
| Refine operation | Rename the block to be consistent with 'DataStoreName' |
| Other scenario dependent behavior | Create corresponding block if brother blocks currently selected |

* DataTypeConversion

|  |  |
| --- | --- |
| Keyword | dt, datatype |
| Examples | dt:sgl dt:u8 |
| Auto ground on selected ports/lines | Yes, with "[" option |
| Auto terminate on selected ports/lines | Yes, with "]" option |
| Insert in line | Yes |
| Multiple blocks at one time | dt\*N |
| String at inport | pass through |
| String at outport | pass through |

* DeadZone

|  |  |
| --- | --- |
| Keyword | dzone, dz, deadzone |
| Examples | dz dz ABC dzLo  Hi |
| Auto ground on selected ports/lines | Yes, with "[" option |
| Auto terminate on selected ports/lines | Yes, with "]" option |
| Insert in line | Yes |
| Multiple blocks at one time | \*N |
| Propagate from upstream | Set lower and upper bounds with suffix '\_Lb' and '\_Ub' |
| Propagate from downstream | Set lower and upper bounds with suffix '\_Lb' and '\_Ub' |
| String at inport | Suffix downstream string with 'Raw' |
| String at outport | pass through |
| Annotation | Range of deadzone |

* Demux

|  |  |
| --- | --- |
| Keyword | demux |
| Examples | demux demux5 |
| Auto number of ports | Yes |
| String at inport | Block name |
| String at outport | Upstream name with suffix '\_DN', where N indicates the outport number, i.e., dimension |
| Plus operation | Outport number |
| Minus operation | Outport number |

* Display

|  |  |
| --- | --- |
| Keyword | disp, display |
| Examples | disp disp5 |
| Auto ground on selected ports/lines | Yes |
| Multiple blocks at one time | \*N |

* Enable

|  |  |
| --- | --- |
| Keyword | en, enable |
| Examples | en |

* From

|  |  |
| --- | --- |
| Keyword | from |
| Examples | from fromabc from+c from5 from5abc fromAAA BBB CCC |
| Auto ground on selected ports/lines | Yes |
| Major property/Property sequence | GotoTag |
| Multiple blocks at one time | Yes |
| Brother blocks | Goto, GotoVisibility |
| Propagate from upstream | Generate corresponding 'Goto' block |
| Propagate from downstream | GotoTag |
| String at outport | GotoTag |
| Refine operation | Rename the block name to be consistent with 'GotoTag' |

* FromWorkspace

|  |  |
| --- | --- |
| Keyword | fromws, fromworkspace |
| Examples | fromws fromwsabc fromws5 fromwsAAA,BBB,CCC |
| Auto ground on selected ports/lines | Yes |
| Major property/Property sequence | VariableName |
| Multiple blocks at one time | Yes |
| Brother blocks | ToWorkspace |
| Propagate from upstream | Generate corresponding 'ToWorkspace' block |
| Propagate from downstream | VariableName |
| String at outport | VariableName |
| Refine operation | Rename the block name to be consistent with 'VariableName' |

* Gain

|  |  |
| --- | --- |
| Keyword | gain |
| Examples | gain gain2 gainabc gainAAA,BBB,CCC |
| Auto ground on selected ports/lines | Yes |
| Auto terminate on selected ports/lines | Yes |
| Insert in line | Yes |
| Major property/Property sequence | Gain |
| String at inport | pass through |
| String at outport | pass through |

* Goto

|  |  |
| --- | --- |
| Keyword | goto |
| Examples | goto goto5 gotoabc gotoAAA,BBB,CCC goto5abc+c |
| Auto terminate on selected ports/lines | Yes |
| Major property/Property sequence | GotoTag |
| Multiple blocks at one time | Yes |
| Brother blocks | From, GotoVisibility |
| Propagate from upstream | GotoTag |
| Propagate from downstream | Create corresponding 'From' block |
| String at inport | GotoTag |
| Annotation | Display scope of the block |
| Refine operation | Rename the block name to be consistent with 'GotoTag' |

* Ground

|  |  |
| --- | --- |
| Keyword | gnd, ground |
| Examples | gnd gnd5 |
| Auto ground on selected ports/lines | Yes |
| Multiple blocks at one time | Yes |

* Inport

|  |  |
| --- | --- |
| Keyword | ip, ipt, inport |
| Examples | ip ip5 ipabc ip5abc ipAAA,BBB,CCC |
| Auto ground on selected ports/lines | Yes |
| Major property/Property sequence | Name Port |
| Multiple blocks at one time | Yes |
| Propagate from upstream | Name |
| Propagate from downstream | Name |
| String at inport | Name |
| String at outport | Name |

* Interpolation\_n-D

|  |  |
| --- | --- |
| Keyword | itp, interpnd |
| Examples | itp itp3abc: block with Dimension set as 3 and Table set as "abc" |
| Auto ground on selected ports/lines | Yes |
| Multiple blocks at one time | Yes |
| Propagate from downstream | Table |
| String at outport | Table |
| Annotation | Show Table name |

* Logic

|  |  |
| --- | --- |
| Keyword | and, or, not, xor, nor, nand |
| Examples | and5 |
| Auto number of ports | Yes |
| String at inport | String representation based on port signals |
| String at outport | String representation based on port signals |
| Roll property operation | Roll across operators |
| Plus operation | Add one more inport |
| Minus operation | Remove inport by one |
| Clean operation | Clean unconnected inports and reorder them to neaten up layout |

* Lookup

|  |  |
| --- | --- |
| Keyword | l1, lookup, lu1d |
| Examples | l1 l1abc l1abcX,abcY l1abcX abcY |
| Auto number of ports |  |
| Auto ground on selected ports/lines | Yes |
| Auto terminate on selected ports/lines | Yes |
| Insert in line | Yes |
| Multiple blocks at one time | Yes |
| Propagate from upstream | Propagate to X axis variable with suffix (InputValues property) |
| Propagate from downstream | Table |
| String at inport | InputValues property |
| String at outport | Table |
| Annotation | Display X, Y variable |
| Refine operation | Rename X axis with "\_X" suffix on Table field |

* Lookup2D

|  |  |
| --- | --- |
| Keyword | l2, lookup2d, lu2d |
| Examples | l2 l2abc l2abcX,abcY,abc |
| Auto ground on selected ports/lines | Yes |
| Multiple blocks at one time | Yes |
| Propagate from downstream | Table |
| String at inport | Corresponding axis string |
| String at outport | Table |
| Annotation | Display X, Y, Z variable |
| Refine operation | Rename X, Y axis with "\_X", "\_Y" suffix on Table field |

* Math

|  |  |
| --- | --- |
| Keyword | math,  exp, log, log10, square, sqrt, reciprocal, pow, rem, mod |
| Examples | exp pow\*2 |
| Auto ground on selected ports/lines | Yes |
| Auto terminate on selected ports/lines | Yes |
| Insert in line | Yes |
| Roll property operation | Roll across operators |

* Merge

|  |  |
| --- | --- |
| Keyword | mg, merge |
| Examples | mg mg5 |
| Auto number of ports | Yes |
| Multiple blocks at one time | \*N |
| String at inport | Rename by outport string with suffix \_In[N] |
| String at outport | Use string from inport 1 |
| Plus operation | "+" add 1 inport "++" number of increase is determined according to unconnected ports currently selected "+N" increase number of inports by N |
| Minus operation | "--" reduce 1 inport "-N" reduce number of inports by N |
| Clean operation | Arrange and remove unconnected inports |

* MinMax

|  |  |
| --- | --- |
| Keyword | mn, mx, min, max |
| Examples | mn max5 |
| Auto number of ports | Yes |
| Multiple blocks at one time | \*N |
| Plus operation | "+" add 1 inport "++" number of increase is determined according to unconnected ports currently selected "+N" increase number of inports by N |
| Minus operation | "--" reduce 1 inport "-N" reduce number of inports by N |
| Clean operation | Arrange and remove unconnected inports |

* MotoHawk Calibration

|  |  |
| --- | --- |
| Keyword | mh cal |
| Examples | mhcal mhcalABC4.0 mhcal1e-1ABC |
| Auto ground on selected ports/lines | Yes |
| Multiple blocks at one time | \*N |
| Propagate from downstream | "nam" and "val" |
| String at outport | val |

* MotoHawk Probe

|  |  |
| --- | --- |
| Keyword | mh prb |
| Examples | mhprb mhprbabc |
| Auto terminate on selected ports/lines | Yes |
| Multiple blocks at one time | Yes |
| Propagate from upstream | nam |
| String at inport | nam |

* Mux

|  |  |
| --- | --- |
| Keyword | mux |
| Examples | mux mux5 |
| Auto number of ports | Context dependent inport number |
| String at inport | Downstream signal suffixed by dimensioni "\_D[N]" |
| Plus operation | "+" add 1 inport "++" number of increase is determined according to unconnected ports currently selected "+N" increase number of inports by N |
| Minus operation | "--" reduce 1 inport "-N" reduce number of inports by N |

* Outport

|  |  |
| --- | --- |
| Keyword | op, opt, outport |
| Examples | op5 opabc op5abc opAAA,BBB,CCC |
| Major Property/Property sequence | Name Port |
| Auto terminate on selected ports/lines | Yes |
| Multiple blocks at one time | Yes |
| Propagate from upstream | Name |
| Propagate from downstream | Name |
| String at inport | Name |

* PreLookup

|  |  |
| --- | --- |
| Keyword | pl, prelookup |
| Examples | pl5 pl5abc |
| Auto terminate on selected ports/lines | Yes |
| Major property | BreakpointsData |
| Multiple blocks at one time | Yes |
| Propagate from upstream | BreakpointsData |
| String at outport | "BreakpointsData" suffix with "\_k", "\_f" |
| Annotation | Display BreakpointsData and IndexSearchMethod |

* Product

|  |  |
| --- | --- |
| Keyword | \*, / |
| Examples | \*\*\* \*/\* /abc \*5.0:sgl |
| 2nd Operand | Yes |

* RateTransition

|  |  |
| --- | --- |
| Keyword | rt, rate, ratetrans |
| Examples | rt rt5 |
| Auto ground on selected ports/lines | Yes |
| Auto terminate on selected ports/lines | Yes |
| Insert in line | Yes |
| Multiple blocks at one time | Yes |
| Propagate from upstream | Pass through |
| Propagate from downstream | Pass through |
| String at inport | Pass through |
| String at outport | Pass through |

* RelationalOperator

|  |  |
| --- | --- |
| Keyword | ==, ~=, <, <=, >=, > |
| Examples | >= ==ABC ~=5:u8 |
| 2nd Operand | Yes |
| Roll property operation | Yes |

* Rounding

|  |  |
| --- | --- |
| Keyword | floor, round, ceil, fix |
| Examples | floor ceil3 |
| Auto ground on selected ports/lines | Yes |
| Auto terminate on selected ports/lines | Yes |
| Insert in line | Yes |
| Multiple blocks at one time | Yes |
| Propagate from upstream | Pass through |
| Propagate from downstream | Pass through |
| String at inport | Pass through |
| String at outport | Pass through |
| Roll property operation | Yes |

* Saturate

|  |  |
| --- | --- |
| Keyword | sat, saturate |
| Examples | sat satA satHi,Lo |
| Auto ground on selected ports/lines | Yes |
| Auto terminate on selected ports/lines | Yes |
| Insert in line | Yes |
| Propagate from upstream | Set upper/lower limit with suffix "\_Hi", "\_Lo" |
| Propagate from downstream | Set upper/lower limit with suffix "\_Hi", "\_Lo" |
| String at inport | Downstream string suffix with "Raw" |
| Annotation | Show upper and lower limit |

* Scope

|  |  |
| --- | --- |
| Keyword | scope, scopes |
| Examples | scope scopes scope5 scopes5 |
| Auto number of ports | Yes, when used with "scope" pattern |
| Auto terminate on selected ports/lines | Yes, when used with "scopes" pattern |
| Multiple blocks at one time | Yes, when used with "scopes" pattern |
| Plus operation | "+" add 1 inport "++" number of increase is determined according to unconnected ports currently selected "+N" increase number of inports by N |
| Minus operation | "--" reduce 1 inport "-N" reduce number of inports by N |
| Clean operation | Arrange and remove unconnected inports |

* SignalConversion

|  |  |
| --- | --- |
| Keyword | sigconv, signalconv |
| Examples | sigconv sigconv5 |
| Auto ground on selected ports/lines | Yes |
| Auto terminate on selected ports/lines | Yes |
| Insert in line | Yes |
| Multiple blocks at one time | Yes |
| Propagate from upstream | Pass through |
| Propagate from downstream | Pass through |
| String at inport | Pass through |
| String at outport | Pass through |

* SignalSpecification

|  |  |
| --- | --- |
| Keyword | sigspec, signalspec |
| Examples | sigspec sigspec5 sigspec:u8 |
| Auto ground on selected ports/lines | Yes |
| Auto terminate on selected ports/lines | Yes |
| Insert in line | Yes |
| Multiple blocks at one time | Yes |
| Propagate from upstream | Pass through |
| Propagate from downstream | Pass through |
| String at inport | Pass through |
| String at outport | Pass through |

* CAN Pack/Unpack

|  |  |
| --- | --- |
| String at inport | Corresponding CAN signal name |
| String at outport | Corresponding CAN signal name |

* Sqrt

|  |  |
| --- | --- |
| Keyword | sqrt |
| Examples | sqrt sqrt5 |
| Auto ground on selected ports/lines | Yes |
| Auto terminate on selected ports/lines | Yes |
| Insert in line | Yes |

**Note that this block is version dependent**

* Stateflow

|  |  |
| --- | --- |
| Keyword | sf, stateflow |
| Examples | sf sfi5o6 sf i5 o6 e2 fc chartname sfin5out6event2fc |
| Auto number of ports | Yes |
| Multiple blocks at one time | \*N |
| Propagate from upstream | Propagate to Stateflow Data name at corresponding inport |
| Propagate from downstream | Propagate to Stateflow Data/Event name at corresponding outport |
| String at inport | Name of corresponding Stateflow Data |
| String at outport | Name of corresponding Stateflow Data/Event |
| Plus operation | +iN: add inport by N +oN: add outport by N +eN: add event output by N +fc: add function call trigger on the chart +iN is equivalent to +inN +oN is equivalent to +outN +eN is equivalent to +eventN |
| Minus operation | vice versa: -iN, -oN, -eN, -fc |
| Clean operation | Arrange and remove unconnected inports and outports |

* SubSystem

|  |  |
| --- | --- |
| Keyword | ss, subsystem |
| Examples | ss ssi5o6 ss i5 o6 fc en sysname sfin5out6fcensysname |
| Auto number of ports | Yes |
| Multiple blocks at one time | \*N |
| Propagate from upstream | Propagate to Inport name at corresponding inport |
| Propagate from downstream | Propagate to Outport name at corresponding outport |
| String at inport | Name of corresponding Inport block |
| String at outport | Name of corresponding Outport block |
| Plus operation | +iN: add inport by N +oN: add outport by N +fc: add function call trigger on the chart +iN is equivalent to +inN +oN is equivalent to +outN |
| Minus operation | vice versa: -iN, -oN, -eN, -fc |
| Clean operation | Arrange and remove unconnected inports and outports |

* Sum

|  |  |
| --- | --- |
| Keyword | +, - |
| Examples | +++ +-+ -abc +5.0:sgl |
| 2nd Operand | Yes |

* Switch

|  |  |
| --- | --- |
| Keyword | sw, switch |
| Examples | sw sw1 sw3 |
| Auto terminate on selected ports/lines | When used with "sw" pattern |
| Insert in line | Yes when used with "sw1" or "sw3" pattern, the number indicates which port to connect |
| Multiple blocks at one time | \*N |
| String at inport | Pass through |
| String at outport | Pass through the 3rd inport |
| Refine operation | Set Criteria to "u2 ~= 0" |

* Terminator

|  |  |
| --- | --- |
| Keyword | term, terminator |
| Examples | term term5 |
| Auto terminate on selected ports/lines | Yes |
| Multiple blocks at one time | Yes |

* ToWorkspace

|  |  |
| --- | --- |
| Keyword | tows, toworkspace |
| Examples | tows towsabc tows5 |
| Auto terminate on selected ports/lines | Yes |
| Multiple blocks at one time | Yes |
| Refine operation | Rename block |

* TriggerPort

|  |  |
| --- | --- |
| Keyword | trig, trigger, fc |
| Examples | trig fc trigfc (equivalent to "fc") trigr (rising) trigf (falling) trig e (either) trigeName |

* Trigonometry

|  |  |
| --- | --- |
| Keyword | sin,cos,tan,asin,acos,atan,atan2,sinh,cosh,tanh,asinh,acosh,atanh,sincos |
| Examples | sin5 |
| Auto ground on selected ports/lines | Yes |
| Auto terminate on selected ports/lines | Yes |
| Insert in line | Yes |
| Multiple blocks at one time | Yes |
| Major property | Operator |
| Roll property operation | Roll through major property |

* UnitDelay

|  |  |
| --- | --- |
| Keyword | ud, dly, unitdelay |
| Examples | dly dlyabc dlyAAA,BBB,CCC dly5 |
| Auto ground on selected ports/lines | Yes |
| Auto terminate on selected ports/lines | Yes |
| Insert in line | Yes |
| Multiple blocks at one time | \*N |
| Major property | X0 |
| String at inport | Pass through |
| String at outport | Pass through |
| Roll property operation | Roll through major property |
| Annotation | Display X0 |

* General MotoHawk blocks
* SiLTest blocks

**Detailed description of script macros:**

* Annotation macro

|  |  |
| --- | --- |
| Keyword | anno |
| Examples | anno anno- annoAnnotation for the block anno+ extra annotation anno+\nextra annotation on another line |
| Explanation | If given with no option string, call the annotation method of the block if available Else if given with option string, use it as the block annotation "anno-" removes annotation from selected blocks "anno+ extra annotation" appends the preceding string to the current annotation Note that Simulink block annotation supports escape symbol "\n" for a new line |

* Connect line macro

|  |  |
| --- | --- |
| Keyword | line |
| HotKey | Ctrl |
| Examples | line |
| Explanation | Connect line automatically among the unconnected ports and lines that is currently selected |

* Connect line macro

|  |  |
| --- | --- |
| Keyword | autoline |
| Examples | autoline |
| Explanation | Connect between selected ports and lines with same name matched |

* Break link macro

|  |  |
| --- | --- |
| Keyword | brk, break |
| Examples | brk |
| Explanation | Break library link of the currently selected blocks |

* Adjust block size macro

|  |  |
| --- | --- |
| Keyword | short(er), long(er), narrow(er), wide(r) |
| HotKey | Alt + ArrowKey: Left (narrower), Right(wider), Up(longer), Down(shorter) |
| Examples | short long2.5 narrow wide |
| Explanation | Change block size of the currently selected blocks. Use extra option as scale factor if given, otherwise use default value 1.1 |

* Clean macro

|  |  |
| --- | --- |
| Keyword | clean |
| Examples | clean |
| Explanation | This command involves three actions: 1. Clean up unconnected lines that currently selected 2. Clean up unconnected blocks that currently selected 3. Call clean method of the currently selected blocks if defined, for example reorder the lines, remove unconnected ports, etc. |

* Add color macro

|  |  |
| --- | --- |
| Keyword | color, bcolor, fcolor |
| Examples | color (random forground color for each block) fcolor (equivalent to "color") bcolorgreen (set background color of selected blocks to green) colorsame/color=/color== (set forground color of selected blocks with the same random color) |
| Explanation | If given no option, set forground color (color/fcolor) or background color (bcolor) to random color for each selected block If given option "same", "=" or "==", set selected blocks with the same random color If given option as color name of RGB specification, set selected blocks with the specified color |

* Add data compare macro

|  |  |
| --- | --- |
| Keyword | datacompare |
| Examples | datacompare |
| Explanation | Given selected subsystem, add a 2 element Mux block together with a Inport block connected at 2nd input. This macro is intended for use with ExpData2Sim tools to compare outcome of simulated data with vehicle test data |

* Set data type macro

|  |  |
| --- | --- |
| Keyword | single, sgl, double, dbl, boolean, bool, uint8, u8, uint16, u16, uint32, u32, int8, i8, int16, i16, int32, i32 |
| Examples | sgl |
| Explanation | Set the block output data type to the specified one |

* Line to from/goto macro

|  |  |
| --- | --- |
| Keyword | line2fg |
| Examples | line2fg |
| Explanation | Convert each selected line to From/Goto pair |

* Magnetic connection macro

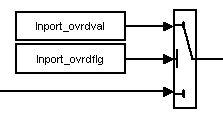
|  |  |
| --- | --- |
| Keyword | magnetic |
| Examples | magnetic |
| Explanation | With subsystem selected, search across the current system by name to find the matching Inport/Outport/From/Goto and make connection between them. This function intends to relieve the user from the condition that the current layer has been such a mess that hard to find a specific block. It is very covenient to be used in combination with "line2fg" that first establish line connections and then convert the lines to From/Goto pair. |

* Set property macro

|  |  |
| --- | --- |
| Keyword | \, \\, … |
| Examples | \abc \\abc \\\-1 \abc,bcd,cde,eee |
| Explanation | With number of N backlashes prefixed, set the Nth parameter of the selected block to the option value. If not specified, use the order of "DialogParameters" as default. For example, for a "UnitDelay" block, "\ABC" sets X0 of the block to "ABC", "\\\-1" sets the Sample Time to "-1" If multiple values given and multiple blocks selected, the values will be set to block one by one, this can be very useful when modeling from external documentation requirement |

* Insert override pattern macro

|  |  |
| --- | --- |
| Keyword | ovrd |
| HotKey |  |
| Examples | ovrd |
| Explanation | With line selected, insert override pattern on the line. |



* String replace

|  |  |
| --- | --- |
| Keyword | => |
| Examples | ABC=>BCD ^=>VABC\_ $=>\_ovrdflg (\w+)Voltage=>$1Volt |
| Explanation | Replace string on the currenly selected blocks, supports MATLAB regular expression. Note 1:  ^ indicates start of string, $ indicates end of string, i.e., "^=>" can be used to add prefix to string, while "$=>" can be used to add suffix to string Note 2:  With Stateflow selected, all objects inside of it will be replaced Note 3:  With SubSystem selected, the blocks inside of it will not be replaced. Use SimReplace tool ifyou want to do so |

* Stateflow Replace

|  |  |
| --- | --- |
| Keyword | >> |
| Examples | ABC>>BCD ^>>VABC\_ $>>\_ovrdflg (\w+)Voltage>>$1Volt |
| Explanation | Replace string only on the currenly selected Stateflow objects. Note that only use this command when Stateflow Editor is currently active. |

* unname

|  |  |
| --- | --- |
| Keyword | unname |
| Examples | unname |
| Explanation | Remove name of selected line |

* MotoHawk to Simulink macro

|  |  |
| --- | --- |
| Keyword | m2s |
| Examples | m2s (equivalent to "m2sall") m2s cal (MotoHawk Calibration to Simulink Constant) m2s dr (MotoHawk DataRead to Simulink Constant) m2s pl (Prelookup) m2s i1 (Interpolation 1D) m2s i2 (Interpolation 2D) m2s1d (Lookup 1D) m2s 2d (Lookup 2D) m2s all (equivalent to "pl, i1, i2, 1d, 2d, cal, dr") m2s table (equivalent to "1d, 2d") m2s scalar (equivalent to "cal, dr") m2s pretable (equivalent to "pl, i1, i2")m2s p1 (Interpolation 1D to Simulink Lookup) m2s p2 (Interpolation 2D to Simulink Lookup2D) |
| Explanation | Convert selected MotoHawk blocks to Simulink blocks |

* Simulink to MotoHawk macro

|  |  |
| --- | --- |
| Keyword | s2m |
| Examples | s2m (equivalent to "s2mall") s2m cal (Simulink Constant to MotoHawk Calibration) s2m pl (Prelookup) s2m i1 (Interpolation 1D) s2m i2 (Interpolation 2D) s2m1d (Lookup 1D) s2m 2d (Lookup 2D) s2m probe (Remove probe and convert to Signal Resolve on line) s2m all (equivalent to "pl, i1, i2, 1d, 2d, cal, probe") s2m table (equivalent to "1d, 2d") s2m scalar (equivalent to "cal") s2m pretable (equivalent to "pl, i1, i2") |
| Explanation | Convert selected Simulink blocks to corrsponding MotoHawk blocks |