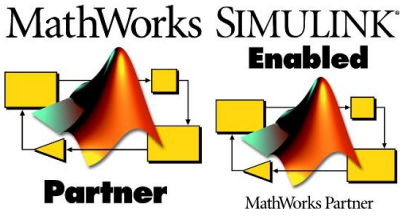
@Source

Model Based Design Development Environment for Simulink®/Stateflow®

Product Specification



December 2023

Podium Technology

Overview

@Source extends MATLAB® and Simulink® by providing a development environment that both increases information content of Simulink® models and productivity.

Information content of the model is increased by providing a block set that enables all parameter and signal properties to be defined at model level, as well as any documentation notes. This enables the entire functionality of a system to be contained within the Simulink® model (i.e. *at source*). This, in turn, makes the Simulink a more useful specification tool, improving information transfer when sharing models with colleagues, customers or suppliers.

Productivity is increased by ensuring information need only be entered once through user friendly dialogues. Tools and utilities are provided that utilise the embedded information to auto generate output files, such as:

- Executable code
- Application or Analysis tool files (any standard or bespoke format, e.g. XML, ASAP)
- Specifications, Documentation or Reports (generating different documents for different audiences, e.g. model documentation for developers or application notes for field engineers).
- Data dictionaries for export.

@Source provides Simulink® blocks that are independent of the target hardware and application tools, therefore the Simulink models focus on the functionality and not on implementation detail. Only at build time does the target specific components come into play. The encompassed information ensures that **@Source** can be configured for any hardware target, operating system, bespoke/legacy environment or any application tool file format (e.g. ASAP – multiple formats can be supported from each build). **@Source** will immediately add value to any of the Simulink Coder (formerly Real-Time Workshop), Embedded Coder or Embedded Targets by outputting fully featured ASAP files and reports.

@Source utilises all the latest developments in Simulink Coder® and Embedded Coder® that enable efficient code to be produced, in code size, memory usage and execution speed, whether this is for production or prototyping.

Key Features:

- **Data Management**

  @Source takes on the duties of managing signals and parameters and their association with the model. Signals and parameters can either be defined in the model and/or with external data dictionaries imported into the model (e.g. legacy variables, BIOS/OS interface or processor register sets). Whichever route is chosen, when the model initialises it is guaranteed to have all the data and properties available to it.

- **Model and Display Units**

  @Source acknowledges the fact that it is better to model in SI units, but the user/developer or application engineers prefers to work with *display* units. Therefore all @Source dialogues offer the set-up of model units and user units and the scaling between them. Standard *quantities* can be defined so that the scaling between, for example, rad/s and rpm is preset to reduce set-up errors.

- **User Friendly Dialogues**

  @Source blocks and utilities proved user friendly dialogues intended to minimise entry errors and improve consistency. For example, when the *quantity* type is selected, e.g. *angular velocity*, then the units selection offers a drop-down dialogue of the available model and display units (rad/s, rpm, rps, krpm etc.)

- **Data Explorer**

  When creating models, it is usually easiest to define the signal and parameter properties by opening up the dialogues at block level. However, when reviewing the model or updating the model, it if often better to take an overview of the model's data. @Source provides an *Explorer* dialogue to enable data to be viewed in a number of ways:

  - **List:** all signals and/or parameters are listed with properties in column.
  - **Menu Groups:** As with the Windows Explorer, a tree view is presented with the user defined menu group hierarchy in the left pane and a list of signals and/or parameters in another pane.
  - **Model:** As above with model hierarchy as the tree view.

  The data explorer offers the ability to highlight multiple signals and/or parameters and change properties all at once.

- **Documentation**

  The facility to document each part of the model is provided with HTML documentation blocks. This facility extends to being able to attach any amount of HTML text with each signal/parameter, which may, for example, contain application notes to be collated into an applications engineers' document.

- **Bitwords and Structures**

  @Source allows the user to define bitwords and structures. Bitwords can have their bits or nibbles at a defined bit number. Bitwords can also be accessed as individual bits/nibbles or in its entirety as an unsigned integer. See the **StructureLib** product specification for more details.

- **Flexible Number Formatting**

  @Source dialogues allow the user to define exactly how the value should be displayed, whether it be numeric, hexadecimal or of an enumerated type (values against strings and colours).

- **Operating System / Legacy Code Integration**

  @Source provides a simple method of hooking into and simulating any OS or legacy event, e.g. background, crank trigger, lap trigger etc.

- **Open API**

  @Source comes with a complete blockset and configured ready to use. However, customers may still want to add to or integrate their own library functions or look and feel to the block mask display. @Source provides an open API, with examples on how this is done.

- **3rd Party Interfacing**

  @Source has been designed to interface to third party libraries and block sets so that models developed with these can benefit from the build targets offered by @Source, often with no changes required other than selecting a new RTW target. It is usually possible to mix these libraries meaning that models from different departments, using different development tools, can be compiled into a common target.

- **Consistency Checking**

  Utilities check that where signals and parameters meet (e.g. map output feeding a signal write) that the quantity and units are consistent.

- **Scalable**

  Suitable from small projects to whole systems through the use of Simulink libraries or Simulink model referencing.

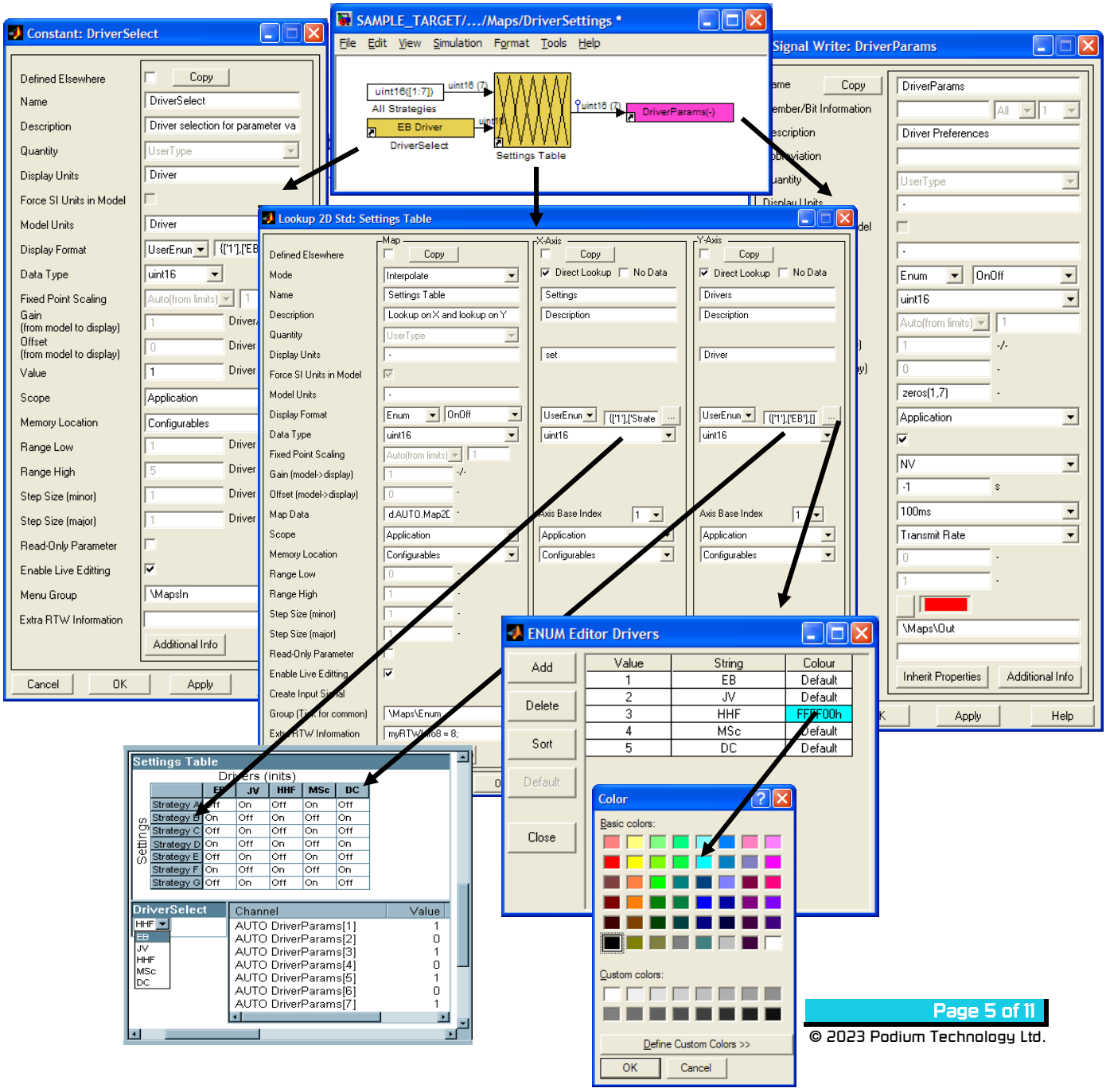
- **On-Line Help**

  Full documentation is available, integrated with the MATLAB HelpDesk.

Defining Properties at Block Level

The diagram below shows an example of how you might use a constant to feed a 2D map (direct lookup mode); repeat the lookup 7 times to get a vector out, one value for each of the 'Settings' axis; finally store the vector for display, logging or reading in another part of the model. The items shown are:

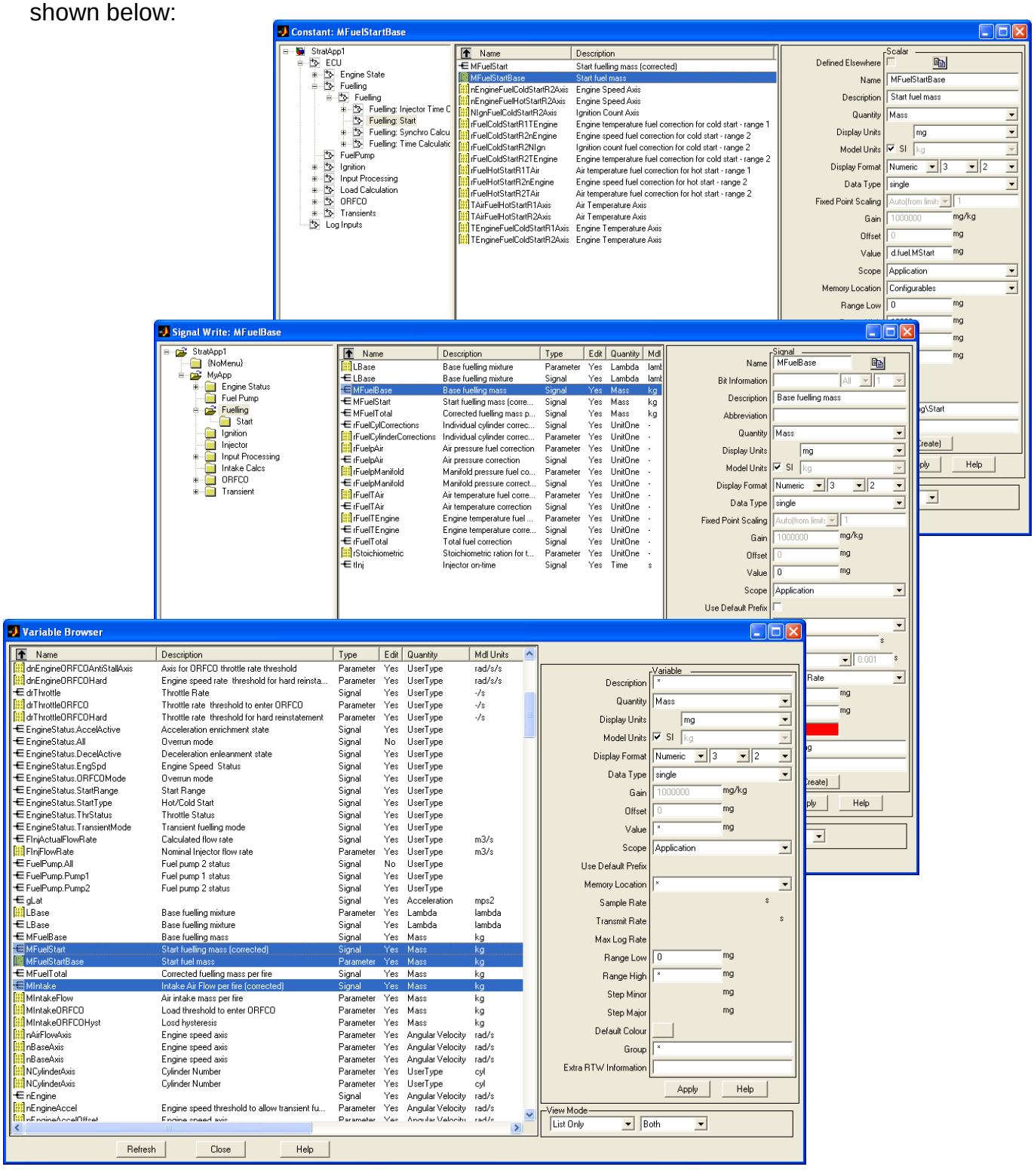
- Simulink model with some @Source blocks
- Constant block dialogue.
- 2D Lookup dialogue: Note the ability to select on each axis whether it is direct lookup or interpolated.
- Signal Write dialogue.
- Enum Editor: This allows text to be associated to values as well as colour (where supported by the application tool). It is possible to predefine your own standard definitions, where used in multiple locations (as shown with the *Signal Write* dialogue, where *OnOff* is the predefined type).
- Example of how this might end up being displayed in an application tool.



Explorer

One of the key benefits of @Source is the *Explorer* that allows the model's data to be reviewed and modified with a number of different views. The explorer allows multiple items to be selected for editing at once. Right-clicking on the item allows you to jump to the defining block, copy the items properties etc.

Some examples of the different views (List, menu group hierarchy and model hierarchy) are shown below:



OS Interface/Function Call Trigger Blocks

The majority of the code will be time triggered and the model will be called in a multi-tasking configuration, a call for each of the time rates in the model, depending on the target configuration. In addition, to give more access to typical operating system functions, a number of function call trigger blocks are supplied:

- **Message call**: Called when a message is available to process.
- **System Event**: Called before or after a system event executes, the system events that are available is dependent on the target, typically these would be:
  - Background
  - Crankshaft interrupt
  - Delete errors
  - NV Corrupt
  - NV Reset
  - Reset Min/Max logged values
  - Data set changed

  The list of system events is definable by the user and provides a mechanism for adding user specific events (such as character received on serial stream, ADC complete etc.)
- **Critical function call:** Enables functionality to be called without external interrupt.
- **Semaphore triggered call**: Triggered when a semaphore is set via a *Set Semaphore* block.
- **Time process call**: Background task that executes on a time basis (without the priority of an interrupt).

All the above will work in simulation with utilities available to send messages during simulation or to trigger system events, such as a lap trigger.

Additional Blocks

A number of additional blocks are supplied to complete the system, these are:

- **Message response**: Used in a message call subsystem to send a response to the message being processed.
- **Message send**: Sends a message to another part of the system or communications bus.
- **Version block:** Block to set the build version with options to add version description and auto-incrementing build numbers.
- **Unit conversion block**: Utilises the *Physical Quantities* information to simplify and make clear the conversion between units (e.g. from rads/s to rpm).
- **Event generation**: Generate a time stamped event
- **Bit-Wise Operations**: Bit-wise logical operations (AND, OR, NOT, XOR, Shift-Left, Shift-Right).
- **Printf**: Implements a fully functioning *C-Style* printf block, during simulation this will print to the command window, in the target this will code as a printf statement, where supported by the target.
- **Sprintf**: As printf except outputting characters for further processing in the model.
- **Dash Display**: Blocks to allow different priority messages to be displayed
- **Joystick**: Block to enable PC joysticks/steering wheels to be used in Simulink simulations or PC targeted build environments.
- **CCP/XCP Calibration Protocol**: Blocks to add support for the CAN Calibration Protocol (CCP) or Generic Calibration Protocol (XCP) over Ethernet.
- **Byte Swap:** Block to byte swap between Intel and Motorola byte ordering.

Documentation Generation

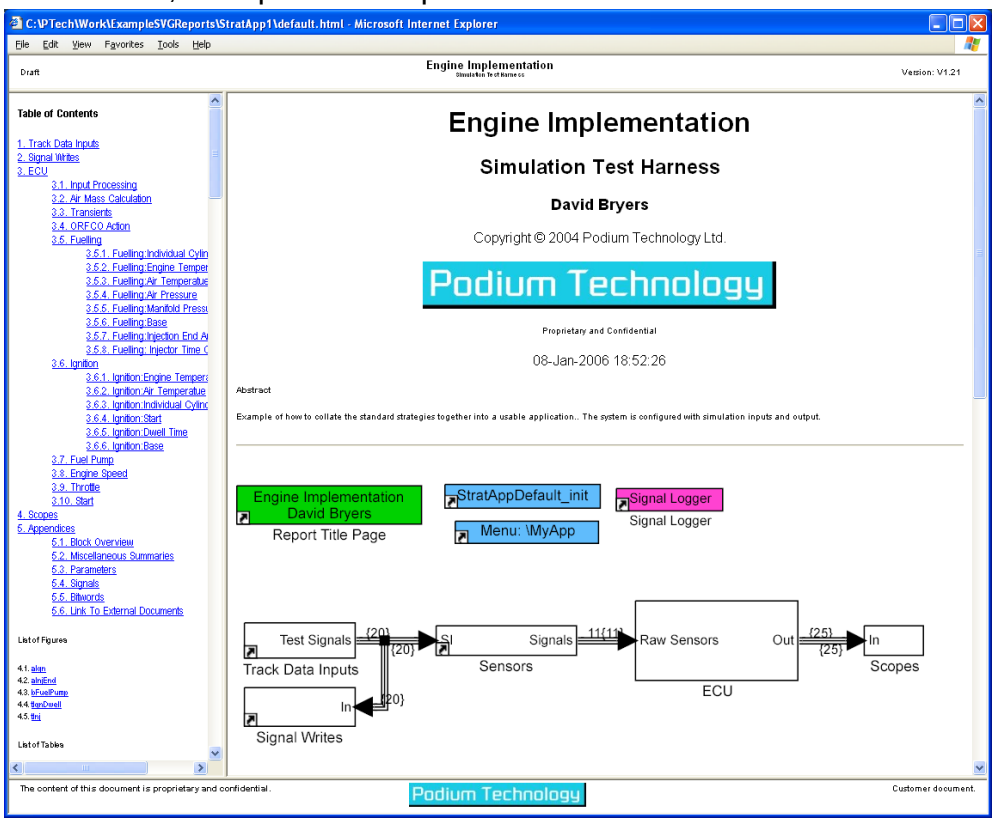
As the model is structured in a logical hierarchy and all information regarding the signals and parameters has been defined, a number of reports can readily be generated. **Doc@Source** provides a number of Simulink Report Generator components that enables you to build up your own report layout that can be fully hyperlinked to make navigation easy. **Doc@Source** is supplied with **@Source**. Unique to **Doc@Source** is its ability to produce Simulink diagrams and Stateflow charts in Scalar Vector Graphics (SVG) format, enabling the full detail of the diagrams to be viewed, zoomed and panned in a browser.

Examples of some of the tables that can be generated are:

- Summary of all signals with hyperlinks to a detailed list of properties and information.
- Summary of all parameters that have been defined with their default data and where they have been used, hyperlinks to a detailed list of properties and information.
- List of inputs and outputs at all levels of the hierarchy, i.e. reading or writing to a higher level of scope.
- Details table for each signal/parameter, listing all properties and listing where defined and used.
- Summaries of individual block properties, e.g. to list all the events that may be generated in the model or messages processed.

To further the cause of making the Simulink model the complete specification, **Doc@Source** also provides an HTML documentation block. This block opens up a fully featured HTML editor that lets formatted text be entered, this text is saved with the model and can be inserted in the generated reports where required. The benefit of multiple blocks around the model is that any changes to a sub-system need only be described in a small block of text and, at the next release, an up to date specification user document can be produced. This bypasses an engineer's natural fear of handling and producing large documents.

A Detailed product specification for **Doc@Source** is available on our web-site.



Configuring @Source to Your Target System

At the core of **@Source**, is its ability to output the functionality defined by the model, in a way required for the target system. Some typical considerations that need to be made on a target-by-target basis are:

- How the model is split before compilation - it is not possible to build a multi-processor system in one step due to the way block I/O structures are generated by Simulink, therefore a model has to be split processor-by-processor, then built (an automatic process).
- Grouping of variables - either by processor, application, task etc.
- Grouping variable with similar attributes into one structure.
- Locating variable of different attributes and scope in specific memory locations.
- Ordering the variable alphabetically, so that each time the model in built, or execution order changed, the variables will be in a consistent order (provided some have not been deleted or added).
- Naming conventions. Naming conventions can be implemented that can take into account:
    - Scope (e.g. prefixing the variable with the application name they are source from, AUTO_Delay)
    - Quantity (e.g. prefixing 'time' variables with a 't', AUTO_tDelay)
    - Data type (e.g. prefixing variables with a letter to denote data type, AUTO_wtDelay or wAUTO_tDelay).
- How variable are read from different scopes, e.g. from an OS call to import from a communication bus, double-reads from DPRAM (to avoid clashes) or direct from memory.
- How to handle NV-RAM, e.g. by locating as a special address or grouping together and periodically writing to EEPROM. Also how to cater for NV corruptions.
- How the different model time rates can be called.
- How bitwords and bits can best be used, e.g. is there bit-addressable memory that can be used, does the compiler support bit structures etc?

The above is not an exhaustive list of the target variations that **@Source** can deal with; some targets are more straightforward than others to define. Once a target definition has been created you can be sure in the knowledge that the same, unmodified, model can then be generated for the other **@Source** targets and any future targets. Hence, when the code needs to be run on different hardware, all that is required is support for that hardware from **@Source**, considerably less effort that revising the model. This method also extends to the application tools, it is easy to migrate from one application tool to another or even support multiple application tools just by selecting different output formats for **@Source** to generate.

@Source is delivered with a number or target hardware options and output file formats, should these not suit your system, Podium Technology can deliver a system to your requirements. Alternatively, with some knowledge of the MATLAB/Simulink® environment,

the customer, through the hook functions provided and the example of the predefined outputs, can configure **@Source**.

System Requirements:

@Source can be used with MATLAB®[i] versions R2017b to R2023b. The following tools are also required:

- Simulink®
- Simulink Coder® (previously Real-Time Workshop®)
- Embedded Coder® (for some targets)

@Source is compatible with Stateflow®, Fixed Point block-set and MATLAB/Simulink Report Generator.

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

Design Service:

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

- Hardware specific block sets for Simulink and Simulink Coder.
- Developing control systems in 'C' or Simulink
- MATLAB/Simulink Report Generator component generation.
- Models for *Hardware-in-the-loop* testers, including driver simulation, data replay etc.
- Utilities for translating 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, Stateflow, Simulink Coder, 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