MES
MODEL ENGINEERING SOLUTIONS

MES Functional Safety Guidelines

Guidelines

Release v.2.1

Mauerstraße 79
10117 Berlin

T: +49 30 2091 6463-0
F: +49 30 2091 6463-33

info@model-engineers.com
www.model-engineers.com

Contents

1 Introduction

The Model Examiner (MXAM) is designed to support automated analysis and correction of guideline violations for Simulink®, Stateflow®, and TargetLink® models. Guidelines that require checking are frequently those used in the safety-critical environment of software modeling. For this reason it is also advisable to check the safety-relevant properties of Simulink®, Stateflow®, and TargetLink® models.

Potential issues include incorrect data type allocation, missing value initialization, and unsuitable adaptation of value ranges to the module interfaces of the model, all of which can ultimately have a negative effect on model simulation and code generation.

MXAM's MES Functional Safety Guidelines safeguards the modeling process with respect to properties that are crucial to the modeling, simulation, and automated code generation of safety-relevant software.

The MES Functional Safety Guidelines check signal exchange at module interfaces and wherever signals merge and are combined. Furthermore, TargetLink function interfaces are investigated to see whether, for example, the types of incoming and outgoing signals, in other words their function parameters, are correctly allocated to data type, converted, and dimensioned. In addition, the MES Functional Safety Guidelines ensure that the initialization of conditionally executable Simulink subsystems is consistent and correct.

2 Consistency of Interface Signals

All signals that mediate data exchange beyond system boundaries are termed interface signals. In principle, this applies to every Simulink subsystem as well as its inputs and outputs (Inports and Outports). Signals that are located at the interface between different system architectures (i.e. between Simulink, TargetLink, and Stateflow) must fulfill particular requirements, as this is where implicit data conversions can take place or missing signal properties are added.

Taken in this context, consistency means that all the properties of a source signal (data type, scaling, dimension, etc.) must comply with the properties specified at the system input, or at the very least, be compatible with them. Compatible means that conversions are possible without any significant loss of accuracy, inefficient code, or the danger of a value range violation.

To properly check consistency, it is also necessary that all relevant properties are specified at the system input. Missing specifications can hinder the optimization of generated code and make it more difficult to recognize value range violations.

The conversion and combination of interface signals requires consistency of:

- Signal type and dimensionality
- Value range information (min and max values of signals), if available
- Scaling information (Gain/LSB, Offset)

Between:

- Inport and source
- Outport and sink(s)
- Inputs that are combined together

Input and output, dependent on signal calculation (e.g. the value range of the result of a multiplication is the same as the product of the value ranges of its factors)

sdt_sc001 - Strong Data Typing at the TargetLink Function Interface

Copyright: Model Engineering Solutions GmbH

Priority: strongly recommended

Objective: verification/validation, workflow, simulation

Source: MES_FS_1_3

Reference: TargetLink Advanced Practices Guide -> Mapping of Subsystems and Functions -> Scaling-Invariant Functions

MATLAB Versions: all

TargetLink Versions: all

Embedded Coder Versions: irrelevant

Automatically Checkable: fully

Manual Review Required: no

Description:

The data type, scaling and value range of the following blocks must be consistent with the data type, scaling and value range of their respective source blocks:

- All TargetLink Inports that provide arguments of a TargetLink function (subsystems containing a TargetLink Function block)
- All TargetLink Outports that define a return value of a TargetLink function.

A TargetLink port is considered to be an argument or a return value of a function if its "Class" is set to "FCN_ARG", "FCN_REF_ARG" or "FCN_RETURN".

If a TargetLink function is configured as scaling invariant, only the data type of the ports need to be consistent with their respective source, but not the scaling information. To configure a TargetLink function as scaling invariant, open the TargetLink function block and select the "Arguments" page. Then select the scaling-invariant check box and enter the scaling propagation function in the scaling propagation edit field.

For further details, please refer the TargetLink document "TargetLink Advanced Practices Guide -> Mapping of Subsystems and Functions -> Scaling-Invariant Functions".

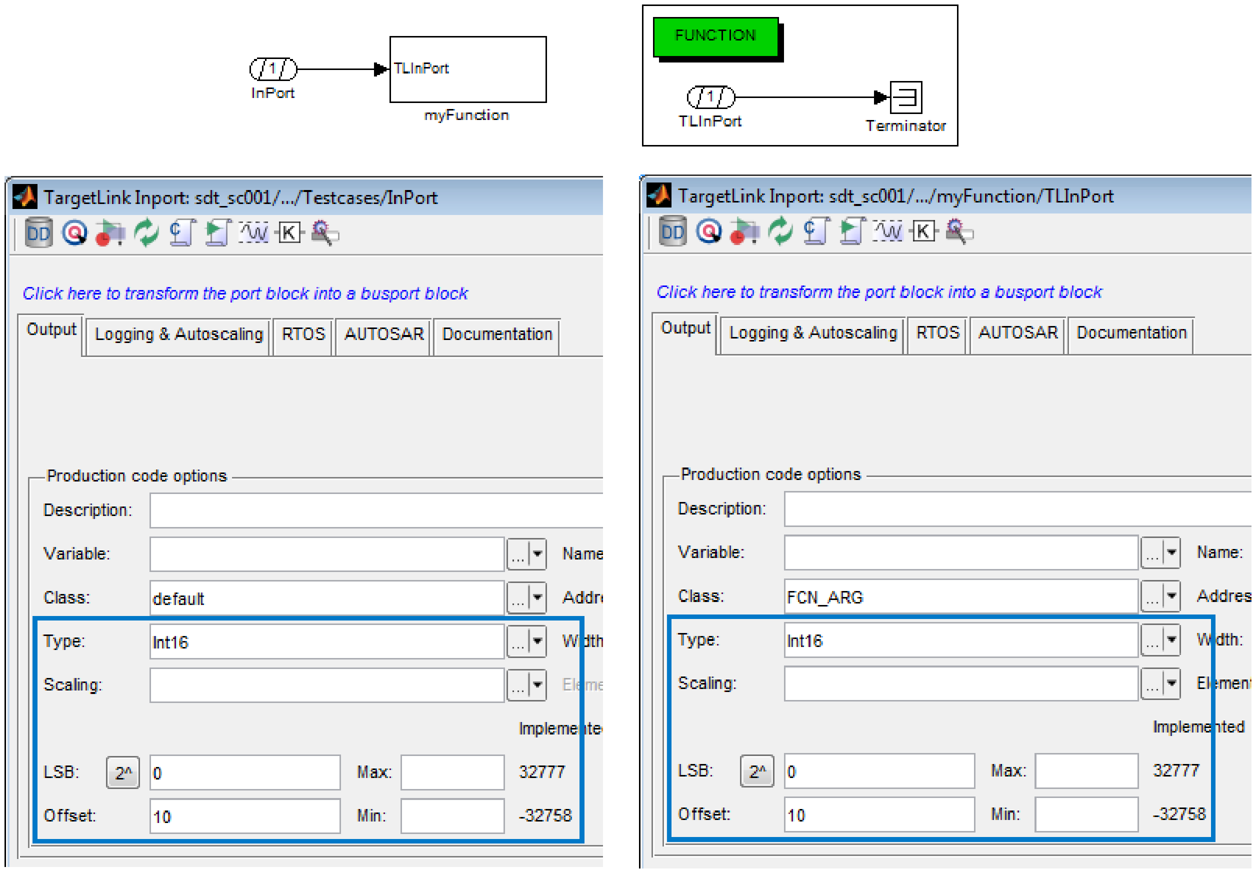
MES Remarks:

This guideline will be marked as obsolete in the coming release. A new version of this guideline is sdt_sc007 and further coming up guidelines.

Rationale:

Inconsistent data types may lead to inefficient code, reduced accuracy or range violations. For function interfaces, strong type consistency of the passed parameters and returned values ensures signal integrity and efficient production code by minimizing the need for type casts and rounding operations. Missing range definitions hinder the detection and control of overflows.

Example:



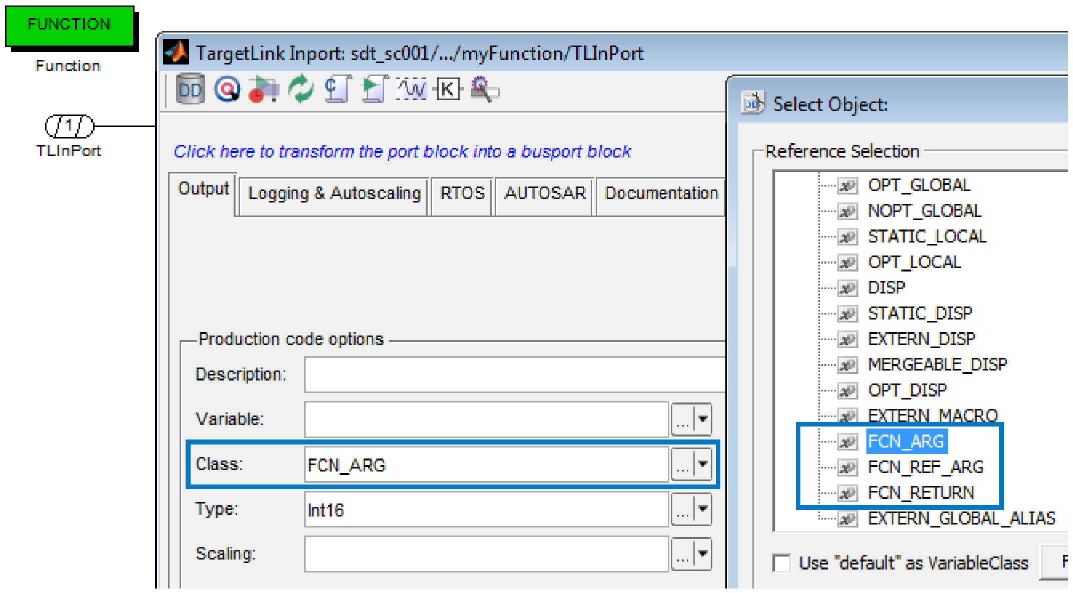
TargetLink Inport: sdt_sc001/.../Testcases/InPort

Description:			
Variable:			
Class:	default		
Type:	Int16		
Scaling:			
LSB:	2^ 0	Max:	32777
Offset:	10	Min:	-32758

TargetLink Inport: sdt_sc001/.../myFunction/TLInPort

Description:			
Variable:			
Class:	FCN_ARG		
Type:	Int16		
Scaling:			
LSB:	2^ 0	Max:	32777
Offset:	10	Min:	-32758

Figure sdt_sc001-1: Equal scaling information, data type and offset of the TargetLink function port and its source


FUNCTION
Function
1
TLInPort
TargetLink Inport: sdt_sc001/.../myFunction/TLInPort
Click here to transform the port block into a busport block
Output
Logging & Autoscaling
RTOS
AUTOSAR
Documentation
Select Object:
Reference Selection

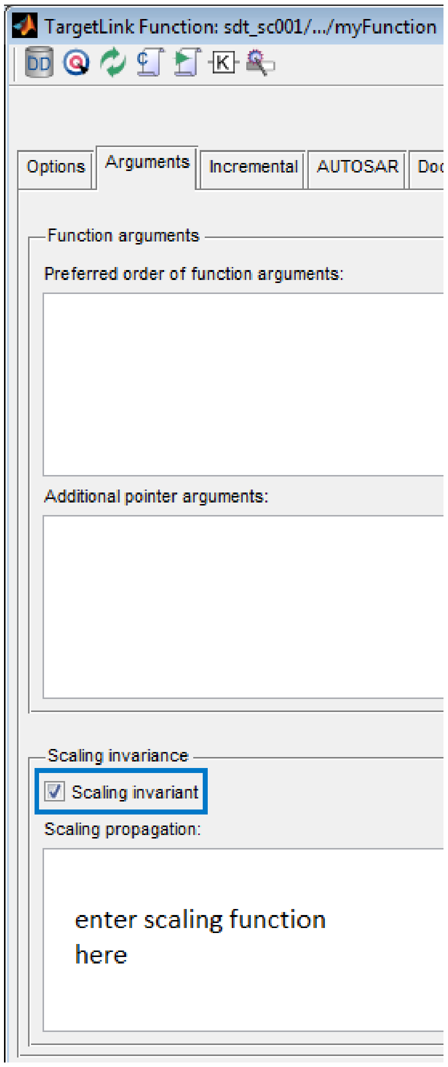
Production code options

Description:	
Variable:	
Class:	FCN_ARG
Type:	Int16
Scaling:	

OPT_GLOBAL
NOPT_GLOBAL
STATIC_LOCAL
OPT_LOCAL
DISP
STATIC_DISP
EXTERN_DISP
MERGEABLE_DISP
OPT_DISP
EXTERN_MACRO
FCN_ARG
FCN_REF_ARG
FCN_RETURN
EXTERN_GLOBAL_ALIAS

Use "default" as VariableClass

Figure sdt_sc001-2: Selection of class "FCN_ARG" for the TargetLink Inport TLInPort


TargetLink Function: sdt_sc001/.../myFunction

Function arguments

Preferred order of function arguments:

Additional pointer arguments:

Scaling invariance

☑ Scaling invariant

Scaling propagation:

enter scaling function here

Figure sdt_sc001-3: Function "myFunction" is declared to be scaling invariant, a valid scaling function must be defined

Model Type: implementation model

ISO 26262-6 Mapping: T1.1c Enforcement of Strong Typing, T8.1g No Implicit Type Conversion

sdt_sc002 - Strong Data Typing at the Stateflow Interface

Copyright: Model Engineering Solutions GmbH

Priority: strongly recommended

Scope: SDT

Objective: code generation, functionality

Source: MES_FS_1_3

MATLAB Versions: all

TargetLink Versions: all

Embedded Coder Versions: irrelevant

Automatically Checkable: fully

Manual Review Required: no

Description:

The data type, scaling and value range of each signal of the Stateflow interface must be well defined and consistent with its source or destination block. To this end, the following criteria must be met.

- The Stateflow chart option "Use Strong Data Typing with Simulink I/O" (Chart -> Properties) must be selected.
- The range of each Stateflow variable must be explicitly defined in the Model Explorer (General -> Limit range). The debug option "Data Range" (Simulation -> Debug -> Debug Chart) assists during a simulation to detect whether the minimum and maximum values you specified for a data item are exceeded. In addition, it is checked whether fixed-point data overflows its base word size.

Note: In Matlab 2015, the Stateflow property setting " Simulation range checking: error" (Simulation-> Configuration Parameters -> Diagnostics-> Data Validity) checks the insufficient data range of a Stateflow data item. The debug option "Data Range" is not valid in Matlab 2015.

If a Stateflow chart resides in a TargetLink system, the data type, scaling, and range

- of each input signal must be consistent with its source block.
- of each output signal must be consistent with its destination block.

In a TargetLink model of TargetLink version less than 2.2, the inheritance mechanism for the size and data type properties at the Simulink/Stateflow interface must not be used.

Rationale:

If the Stateflow chart option "Use Strong Data Typing with Simulink I/O" is not selected, Stateflow performs implicit data type conversions at the Simulink/Stateflow interface. This may lead to data loss for any variable with scope "Input from Simulink" or "Output to Simulink".

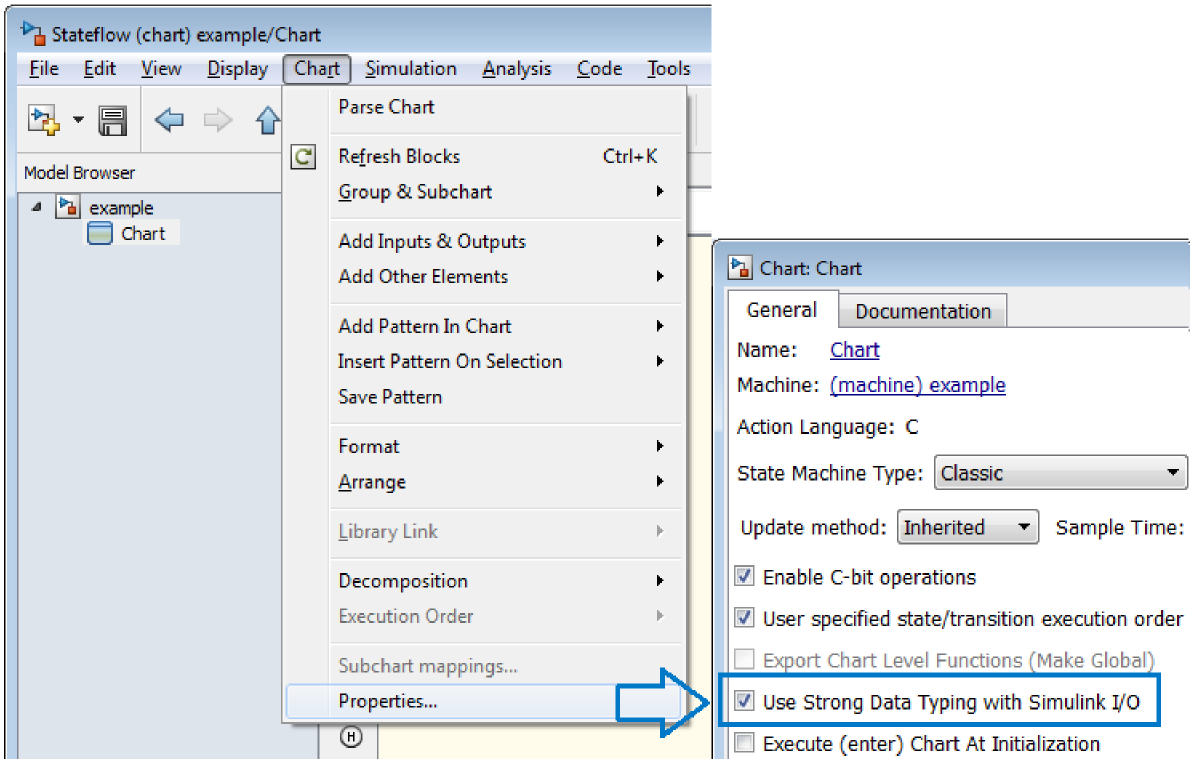
If the range of a Stateflow variable is set as tight as possible, the internal TargetLink Stateflow scaling algorithm can choose an appropriate data type for any intermediate result variable. This leads to operations with small bit sizes. TargetLink uses user defined ranges to optimize the generated code. Missing range definitions hinder the detection and control of overflows.

Inconsistent data types may lead to inefficient code, reduced accuracy or range violations. For function interfaces, strong data type consistency of each passed parameter and each return value ensures signal integrity and efficient production code minimizing the need for type casts and rounding operations.

Between MATLAB versions R2010b and R2012b, the option "Use Strong Data Typing with Simulink I/O" has not been selectable in the GUI. It is set to "true" by default. It is, however, possible to change its value through the API.

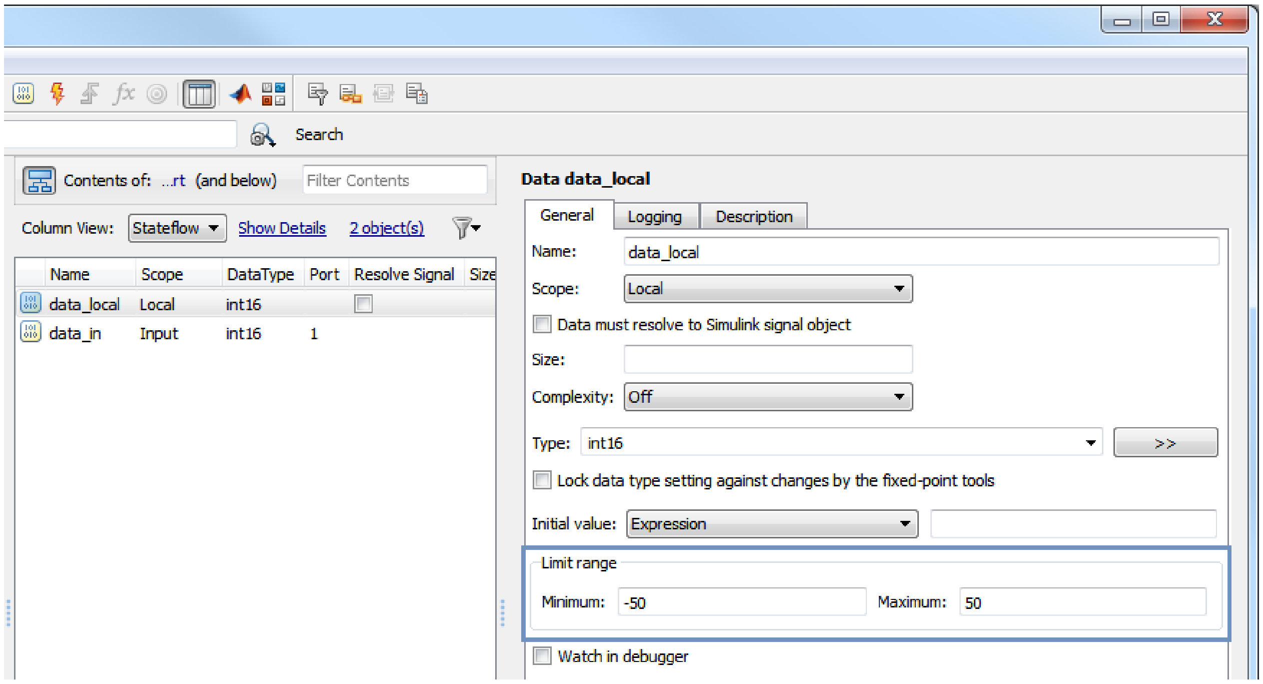
Type/size inheritance was introduced with MATLAB Release R14 and is not supported by TargetLink versions prior to 2.2.

Example:



Name: Chart
Machine: (machine) example
Action Language: C
State Machine Type: Classic
Update method: Inherited Sample Time:
[✓] Enable C-bit operations
[✓] User specified state/transition execution order
[ ] Export Chart Level Functions (Make Global)
[✓] Use Strong Data Typing with Simulink I/O
[ ] Execute (enter) Chart At Initialization

Figure sdt_sc002-1: Selection of the chart option "Use Strong Data Typing with Simulink I/O"



Search

Contents of: ...rt (and below) Filter Contents

Column View: Stateflow Show Details 2 object(s)

Name	Scope	DataType	Port	Resolve Signal	Size
data_local	Local	int16		☐	
data_in	Input	int16	1		

Data data_local

General | Logging | Description

Name: data_local

Scope: Local

☐ Data must resolve to Simulink signal object

Size:

Complexity: Off

Type: int16

☐ Lock data type setting against changes by the fixed-point tools

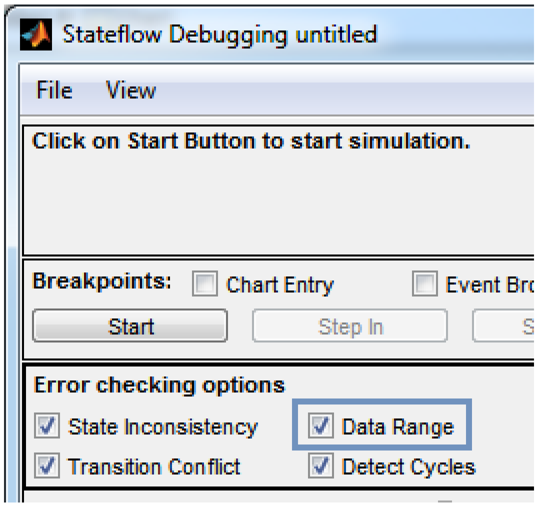
Initial value: Expression

Limit range

Minimum: -50 Maximum: 50

☐ Watch in debugger

Figure sdt_sc002-2: The minimum and maximum values of the Stateflow variable data_local are set



Stateflow Debugging untitled

File View

Click on Start Button to start simulation.

Breakpoints: ☐ Chart Entry ☐ Event Bro

Start | Step In | S

Error checking options

☑ State Inconsistency ☑ Data Range

☑ Transition Conflict ☑ Detect Cycles

Figure sdt_sc002-3: The Stateflow chart debug option "Data Range" is set

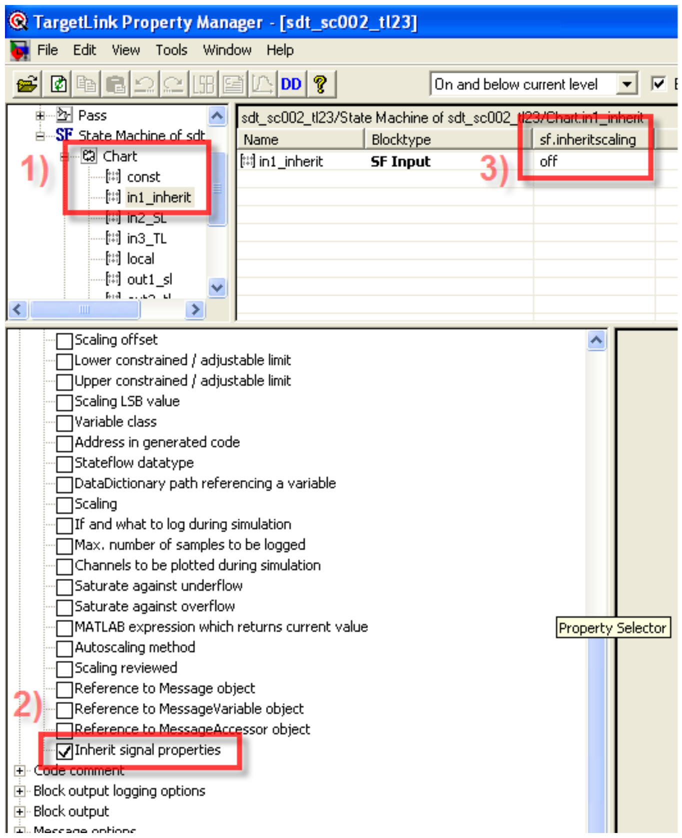
If working with TargetLink versions prior to 2.2.1, the inheritance mechanism for the size and type of Stateflow data needs to be deactivated.

To deactivate this option, open the TargetLink Property Manager (see Figure sdt_sc002-4) and follow the steps below

1) Select the respective chart input data item

2) Set the view option "Inherit signal properties" in section "TargetLink properties of Stateflow objects"

3) Set the property sf.inheritscaling to "off"



sdt_sc002_tl23/State Machine of sdt_sc002_tl23/Chart.in1_inherit

Name	Blocktype	sf.inheritscaling
in1_inherit	SF Input	off

- ☐ Scaling offset
- ☐ Lower constrained / adjustable limit
- ☐ Upper constrained / adjustable limit
- ☐ Scaling LSB value
- ☐ Variable class
- ☐ Address in generated code
- ☐ Stateflow datatype
- ☐ DataDictionary path referencing a variable
- ☐ Scaling
- ☐ If and what to log during simulation
- ☐ Max. number of samples to be logged
- ☐ Channels to be plotted during simulation
- ☐ Saturate against underflow
- ☐ Saturate against overflow
- ☐ MATLAB expression which returns current value
- ☐ Autoscaling method
- ☐ Scaling reviewed
- ☐ Reference to Message object
- ☐ Reference to MessageVariable object
- ☐ Reference to MessageAccessor object
- ✓ Inherit signal properties

Figure sdt_sc002-4: Deactivation of the inheritance of signal properties in the TargetLink Property Manager

Related Guidelines: misra_tl_4_2, dSPACE ds_0058 (2.16.5: Permitted Data Types at the Stateflow Chart Interface), dSPACE ds_0056 (2.16.2: permitted Dimensionality at the Stateflow Chart Interface), dSPACE ds_0072 (2.16.19: Simulink Stateflow Inheritance), dSPACE ds_0112 (12.2: Specification of Range Information for Variables)

Model Type: functional model

ISO 26262-6 Mapping: T1.1c Enforcement of Strong Typing, T8.1g No Implicit Type Conversion

sdt_sc003 - Strong Data Typing of Merge Blocks

Copyright: Model Engineering Solutions GmbH

Priority: recommended

Objective: code generation

Source: MES_FS_1_3

MATLAB Versions: all

TargetLink Versions: all

Embedded Coder Versions: irrelevant

Automatically Checkable: fully

Manual Review Required: no

Description:

The following properties must hold for each Simulink and TargetLink Merge block.

- The "Allow unequal port widths" property must not be checked, and
- data types, scaling and ranges must be identical for all inputs or elements of inputs.

Depending on the TargetLink version of the model, the following properties must hold for each TargetLink Merge block.

- TargetLink versions prior to 2.2.1: bus inputs are not allowed.
- TargetLink versions 2.2.1 or newer: bus inputs are only allowed, if the block property "Inherit properties" is selected.

Rationale:

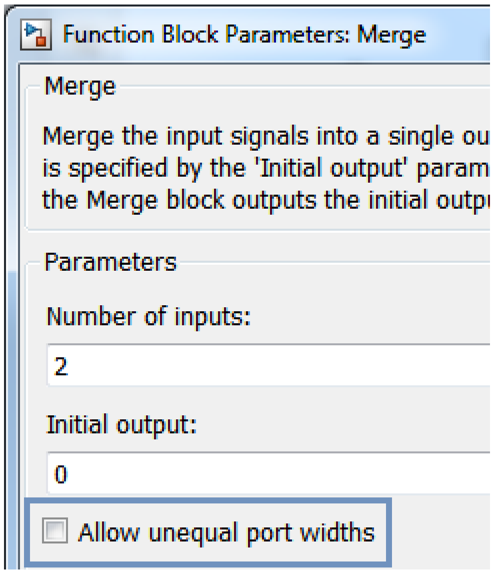
Inconsistent data types may lead to inefficient code, reduced accuracy or range violations.

TargetLink does not support the "Allow unequal port widths" option as it effects code efficiency.

If data types and scaling differ between the inputs of a TargetLink Merge block, the error cannot be detected by compiling/updating the model. If the "Inherit properties" option of a merge block is set, TargetLink requires all inputs to have the same data type and scaling. This can only be tested in code generation. If the property "Inherit properties" option is set to "off", and the data type of all inputs is not the same, the inputs are cast to the data type of the Merge block which may lead to overflow or underflow of the input values.

Bus-capable Merge blocks have been introduced with TargetLink version 2.2.1.

Example:


Function Block Parameters: Merge

Merge

Merge the input signals into a single ou
is specified by the 'Initial output' param
the Merge block outputs the initial outp

Parameters

Number of inputs:

2

Initial output:

0

☐ Allow unequal port widths

Figure sdt_sc003-1: The "Allow unequal port widths" property of a Merge block is not selected.

Counter Example:

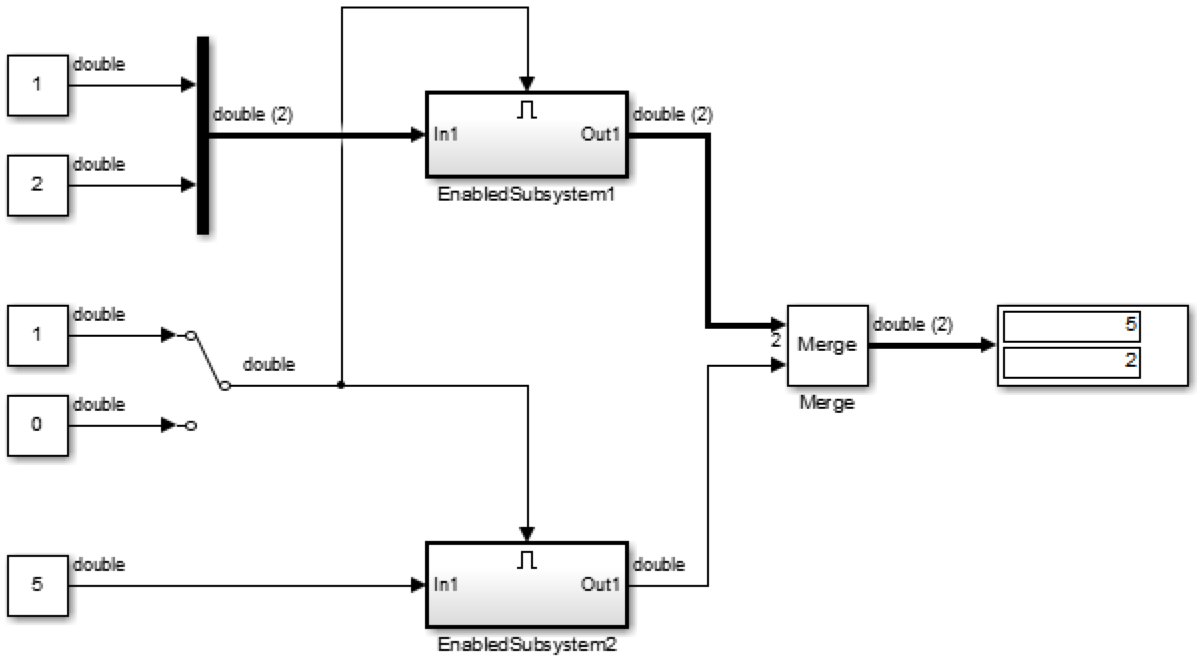


Figure sdt_sc003-2: Incorrect mixture of signals if the Merge block property "Allow unequal port widths" is set.

Related Guidelines: hisl_0015 C

Model Type: functional model

ISO 26262-6 Mapping: T1.1c Enforcement of Strong Typing, T8.1g No Implicit Type Conversion

sdt_sc004 - Strong Data Typing of Arithmetic Blocks

Copyright: Model Engineering Solutions GmbH

Priority: strongly recommended

Scope: SDT

Objective: code generation

Source: MES_FS_1_3

MATLAB Versions: all

TargetLink Versions: all

Embedded Coder Versions: irrelevant

Automatically Checkable: fully

Manual Review Required: no

Description:

Each Arithmetic block with two or more input signals (Sum block, Product block, Dot Product block) must use consistent and adequate data types. Therefore,

4) The option "Require all inputs to have the same data type" must be set.
5) Any scaling and min/max values of the input signals must be consistent.
6) The data range of the output data type must be adequate for holding any result of the arithmetic operation.

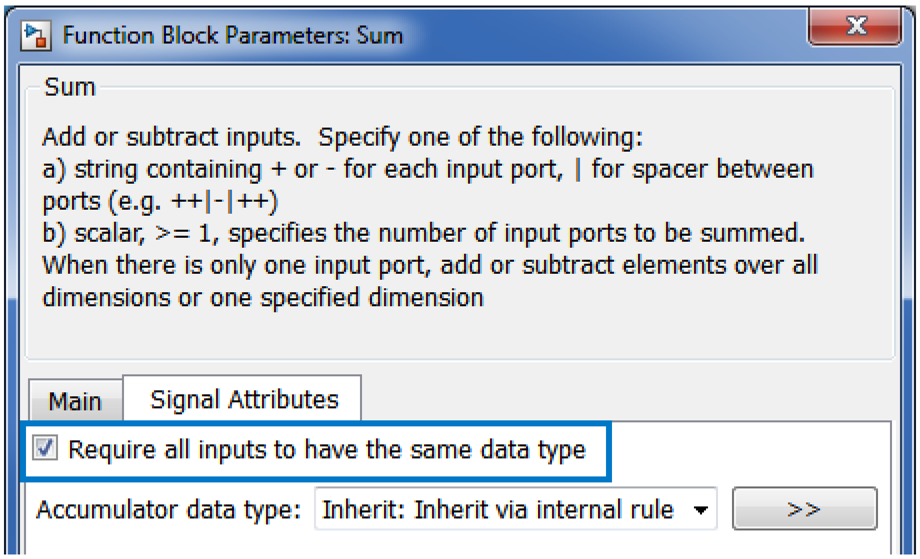
When using TargetLink, provide information about the known data range to constrain the input range, whenever possible.

Rationale:

Inconsistent data types and scaling may lead to inefficient code and reduced accuracy or range violations.

Output data types with an insufficient range are not necessarily detected by Simulink or TargetLink. Implicit signal saturation or rounding may yield unexpected results.

Example:



Function Block Parameters: Sum

Sum

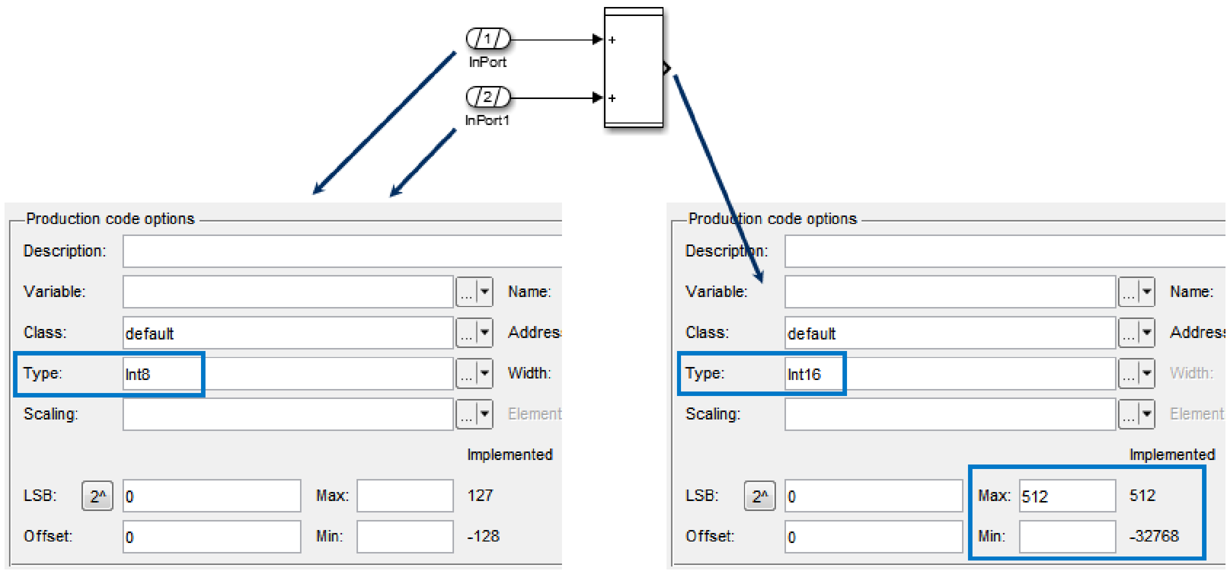
Add or subtract inputs. Specify one of the following:
a) string containing + or - for each input port, | for spacer between ports (e.g. ++|-|++)
b) scalar, >= 1, specifies the number of input ports to be summed.
When there is only one input port, add or subtract elements over all dimensions or one specified dimension

Main | Signal Attributes

[✓] Require all inputs to have the same data type

Accumulator data type: Inherit: Inherit via internal rule

Figure sdt_sc004-1: Option "Require all inputs to have the same data type" of a Sum block is set



Production code options

Description:
Variable:
Class: default
Type: Int8
Scaling:
LSB: 2^ 0
Offset: 0
Max:
Min:
Implemented 127 -128

Production code options

Description:
Variable:
Class: default
Type: Int16
Scaling:
LSB: 2^ 0
Offset: 0
Max: 512
Min:
Implemented 512 -32768

Figure sdt_sc004-2: The data types of the inputs of the Sum block are consistent. The data range of the output data type of the Sum block is adequate to hold the result of operation.

Model Type: functional model

ISO 26262-6 Mapping: T1.1c Enforcement of Strong Typing, T8.1g No Implicit Type Conversion

sdt_sc007 - Definition TargetLink Function Interface

Copyright: Model Engineering Solutions GmbH

Priority: strongly recommended

Objective: verification/validation, workflow, simulation

Source: MES_FS_1_3

MATLAB Versions: all

TargetLink Versions: >2.0

Embedded Coder Versions: irrelevant

Automatically Checkable: fully

Manual Review Required: no

Description:

A proper definition of a TargetLink Function Interface leads to the following statements.

The Inports and Outports on the toplevel of a TargetLink Function subsystem must be TargetLink ports.

A TargetLink port is considered to be an argument or a return value of a function if its "Class" is set to "FCN_ARG", "FCN_REF_ARG" or "FCN_RETURN".

If one of these classes is set the data type, scaling and value range of the following blocks must be consistent:

- All TargetLink Inports that provide arguments of a TargetLink function with their respective source.
- All TargetLink Outports that define a return value of a TargetLink function with their respective destination.

If a TargetLink function is configured as scaling invariant, only the data type of the ports need to be consistent with their respective source, but not the scaling information.

If a TargetLink port is NOT to be considered as an argument or return value of a function, the class should still be selected explicitly, i.e. the class 'default' should not be used.

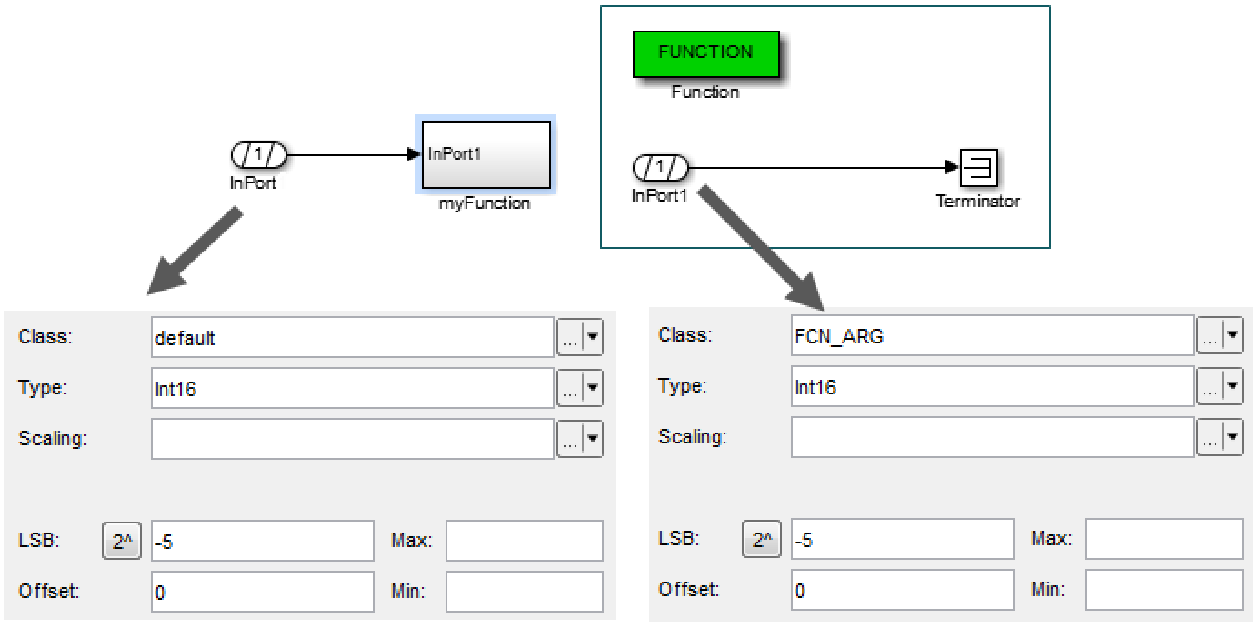
MES Remarks:

The guideline is still in draft status and the last part is not yet implemented. The guideline is derived from guideline sdt_sc001 and describes the TargetLink Function Interface outwards.

Rationale:

Inconsistent data types may lead to inefficient code, reduced accuracy or range violations. For function interfaces, strong type consistency of the passed parameters and returned values ensures signal integrity and efficient production code by minimizing the need for type casts and rounding operations. Missing range definitions hinder the detection and control of overflows.

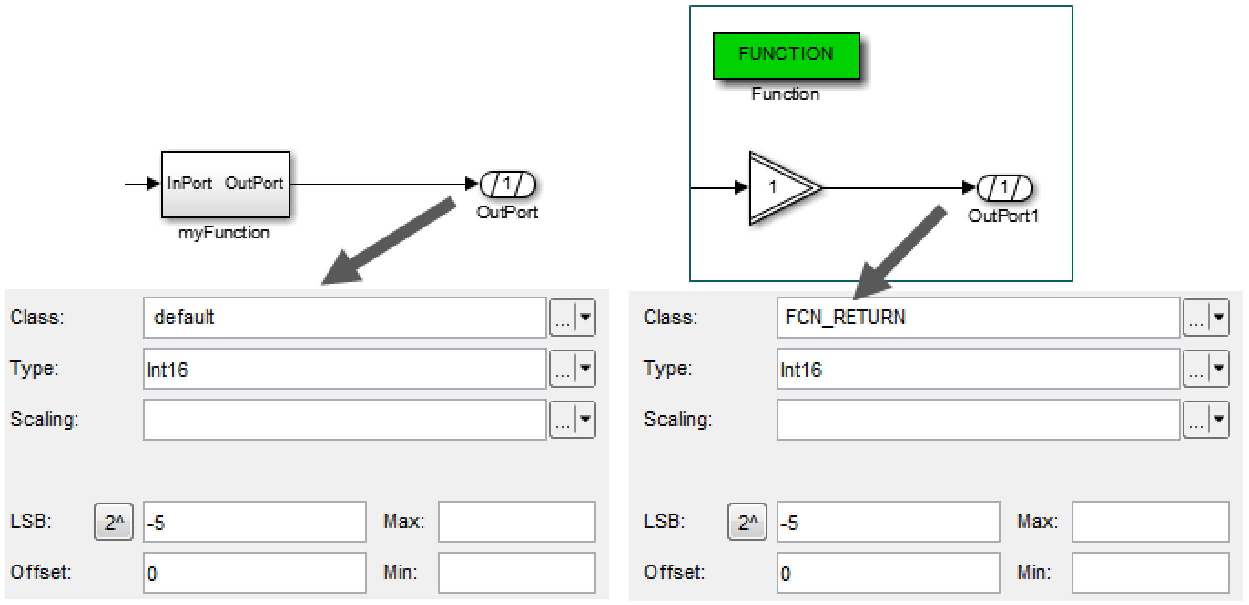
Example:



Class:	default		
Type:	Int16		
Scaling:			
LSB:	2^ -5	Max:	
Offset:	0	Min:	

Class:	FCN_ARG		
Type:	Int16		
Scaling:			
LSB:	2^ -5	Max:	
Offset:	0	Min:	

Figure sdt_sc007-1: Equal scaling information, data type and LSB of the TargetLink function Inport and its source



Class:	default		
Type:	Int16		
Scaling:			
LSB:	2^ -5	Max:	
Offset:	0	Min:	

Class:	FCN_RETURN		
Type:	Int16		
Scaling:			
LSB:	2^ -5	Max:	
Offset:	0	Min:	

Figure sdt_sc001-2: Equal scaling information, data type and LSB of the TargetLink function Outport and its destination

Model Type: implementation model

ISO 26262-6 Mapping: T1.1c Enforcement of Strong Typing, T8.1g No Implicit Type Conversion

3 Adequacy of Signal Properties

It is important to ensure that signals are always adequate for the respective block type. Signals with incorrect data types, scaling and dimensions can result in undefined states, value range violations, or losses in accuracy. Furthermore, the associated type conversions can result in implementation-dependent side effects.

Signals with the correct data type and scaling, on the other hand, increase the comprehensibility, transparency, and adaptability of a model.

3.1 Control signal requirements

- Switch signals (switch block, conditionally executable subsystems) must be of a Boolean type, or of a discrete numeric type with an adequate value range [0 | >0].
- Routing signals (e.g. index signals from Multiport switches, variables in Stateflow transition conditions) may only display a discrete data type. Their value range must match the range of the index used to route the signal. If, for example, a control signal of a Multiport switch routes three data lines, its' value range must be restricted to [1..3], or, if zero-based indexing is used, to [0..2]. Moreover, the value range must cover the entire index range. Thus, a Boolean control signal would be inadequate, because it is not able to route the third data line.

sdt_sa003 - Control Input of Switch Blocks

Copyright: Model Engineering Solutions GmbH

Priority: strongly recommended

Objective: code generation

Source: MES FS 1.3

MATLAB Versions: all

TargetLink Versions: all

Embedded Coder Versions: all

Automatically Checkable: fully

Manual Review Required: no

Description:

The data type of the control signal of each Switch block must be Boolean.

Rationale:

Restricting the data type of the control signal of a Switch block to Boolean avoids undefined output conditions which ensures compliance with MISRA C rule 30 (variables must be initialized before being used).

Model Type: functional model

ISO 26262-6 Mapping: T1.1c Enforcement of Strong Typing, T8.1g No Implicit Type Conversion

MISRA-C: 30

sdt_sa004 - Control Input of Multiport Switch Blocks

Copyright: Model Engineering Solutions GmbH

Priority: strongly recommended

Scope: SDT

Objective: code generation, functionality, maintainability

MATLAB Versions: all

TargetLink Versions: all

Embedded Coder Versions: all

Automatically Checkable: fully

Manual Review Required: no

Description:

The control input of each Multiport Switch block must be of an unsigned integer type. Further, the control input must have a range that is limited to [1 .. number of data ports].

Rationale:

Restricting the data type of the control input of a Multiport Switch block to an unsigned integer type avoids rounding and resolution effects on the switch behavior. Restricting the range of the control input avoids undefined output conditions.

Related Guidelines: dSPACE ds_0107 (8.13: Use of Data Types for Multiport Switch Blocks)

Model Type: functional model

ISO 26262-6 Mapping: T1.1c Enforcement of Strong Typing, T8.1g No Implicit Type Conversion

3.2 Numerical and logical signal requirements

- Value ranges should be defined as a signal property if they are known at the time of design. The availability of value ranges enables a precise scaling of data. Moreover, their definition safeguards the integrity of data and simplifies the analysis of dataflow as well as system dynamics.
- Scaling should always be adapted to the respective value range.
- Continual signals may not feed into discrete operators. If this cannot be avoided, an explicit type conversion should be carried out. This restriction ensures clear semantics of discrete operations (e.g., what is the result of 0.4 AND -0.01?) and avoids implementation-specific dependencies, e.g. due to rounding.
- The scaling (accuracy) of the signal must be taken into account in the case of continual relational and arithmetic operations (e.g. with divisions or zero comparisons). For example, avoid dividing by low-resolution signals if their values can get close to zero. In this case, the signal may have to be rescaled. When comparing continual signals, their scaling have to be adjusted.

sdt_sa001 - Strong Data Typing at the Input of Logical Operator Blocks

Copyright: Model Engineering Solutions GmbH

Priority: recommended

Objective: code generation, functionality

Source: MES_FS_1_3

MATLAB Versions: all

TargetLink Versions: all

Embedded Coder Versions: not applicable

Automatically Checkable: fully

Manual Review Required: no

Description:

Each input signal of a Logical block must define a Boolean data type. Therefore,

- for each Combinatorial Logic block and Logical Operator block, the block option "Require all inputs and output to have the same data type" must be set, and
- for each TargetLink Logical Operator block, every input signal must be defined with a Boolean base data type.

Rationale:

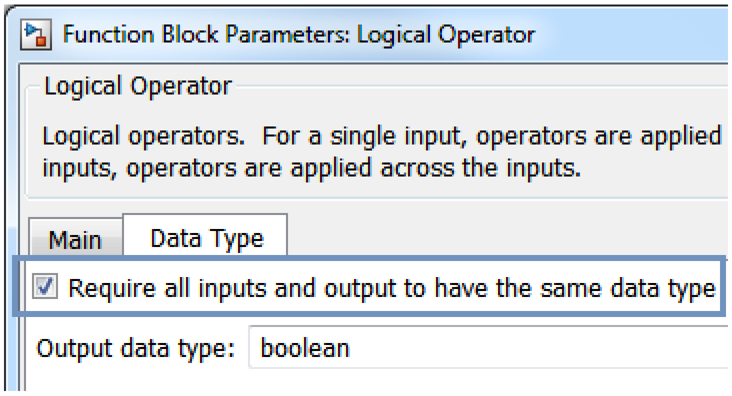
Inconsistent data types may lead to inefficient code, reduced accuracy or range violations.

Restricting logical blocks to process and output Boolean data types only, helps to avoid design errors, enhances readability and enforces strict semantics of the processed signals.

Each input signal value of a logical block that is unequal to zero is interpreted as true, which

- may lead to unexpected results (0.1 and -0.1 = true?, see Figure sdt_sa001-1), and
- makes the code vulnerable to rounding and resolution effects.

Example:


Function Block Parameters: Logical Operator

Logical Operator

Logical operators. For a single input, operators are applied
inputs, operators are applied across the inputs.

☑ Require all inputs and output to have the same data type

Output data type: boolean

Figure sdt_sa001-1: The option "Require all inputs and output to have the same data type" is selected for a Simulink Logical Operator block

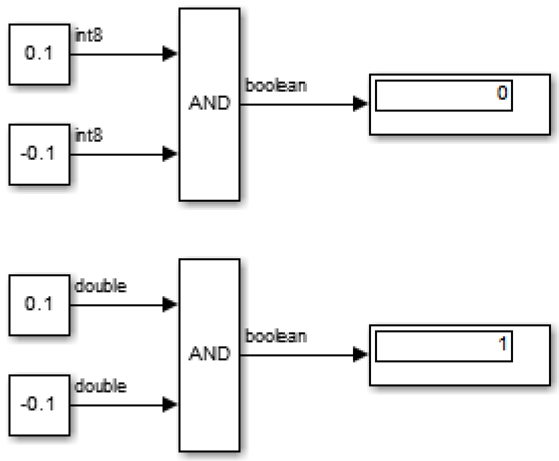


Figure sdt_sa001-2: Type dependency of the result of a logical operation. The result of the lower AND block is true, whereas the result of the upper AND block (with the same input values) is false.

Model Type: functional model

ISO 26262-6 Mapping: T1.1c Enforcement of Strong Typing, T8.1g No Implicit Type Conversion

sdt_sa002 - Output Type of Logical and Relational Operator Blocks

Copyright: Model Engineering Solutions GmbH

Priority: strongly recommended

Scope: SDT

Objective: code generation, functionality

Source: MES FS 1.3

MATLAB Versions: all

TargetLink Versions: all

Embedded Coder Versions: irrelevant

Automatically Checkable: fully

Manual Review Required: no

Description:

The "Output data type" property of each Relational Operator block and each Logical Operator block must be either set to "Boolean" or "Logical".

If "Logical" is selected, the configuration parameter "Implement logic signals as Boolean data (vs. double)" (see the "Optimization" section) of the model must also be checked.

Rationale:

For means of code efficiency, any data type other than Boolean or Logical is not allowed for the output signal of a Relational Operator block and a Logical Operator block. Moreover, with this setting, Simulink only allows Boolean input signals to Logical Operator blocks, which further increases code efficiency and validity.

Example:

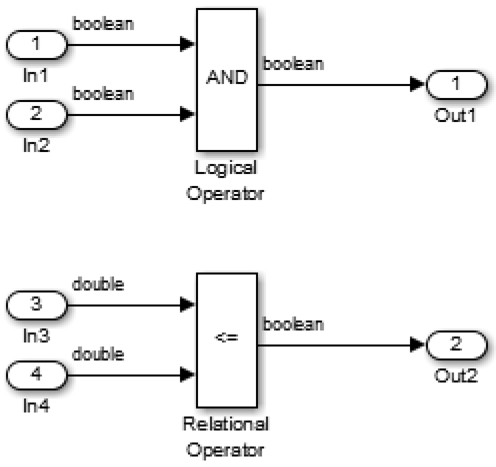
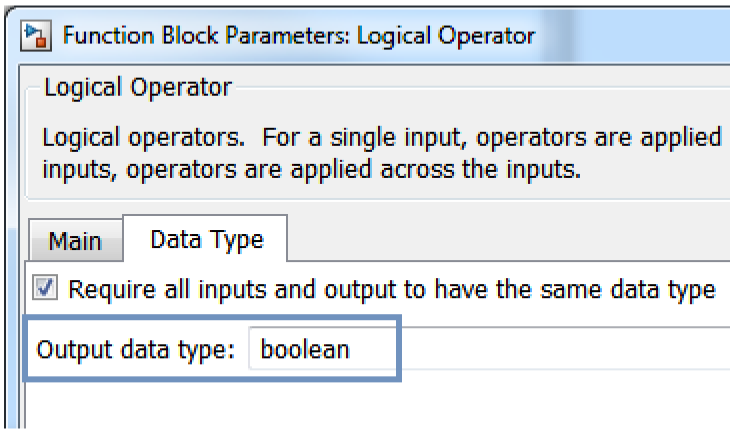


Figure sdt_sa002-1:The data type of the output of the Logical Operator block is set to Boolean (see Figure sdt_sa002-2) and the data type of the output of the Relational Operator block is set to Logical (see Figure sdt_sa002-3).


Function Block Parameters: Logical Operator

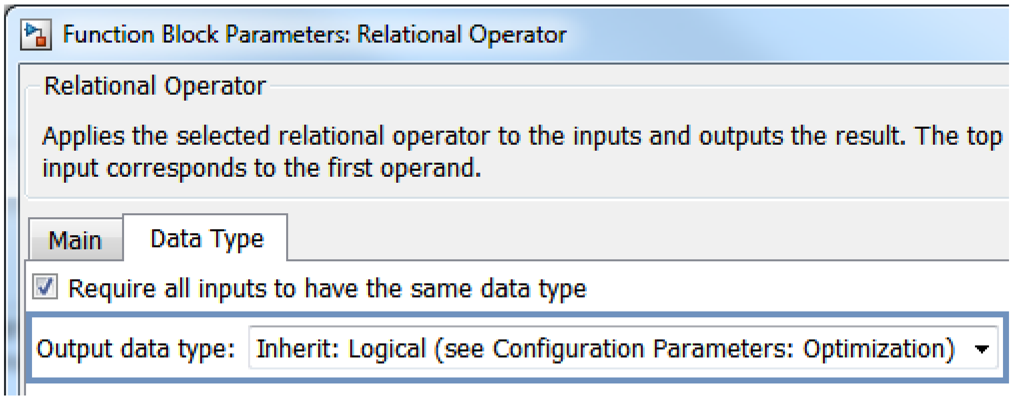
Logical Operator

Logical operators. For a single input, operators are applied
inputs, operators are applied across the inputs.

☑ Require all inputs and output to have the same data type

Output data type: boolean

Figure sdt_sa002-2: The data type of the output of the Logical Operator block is set to Boolean.


Function Block Parameters: Relational Operator

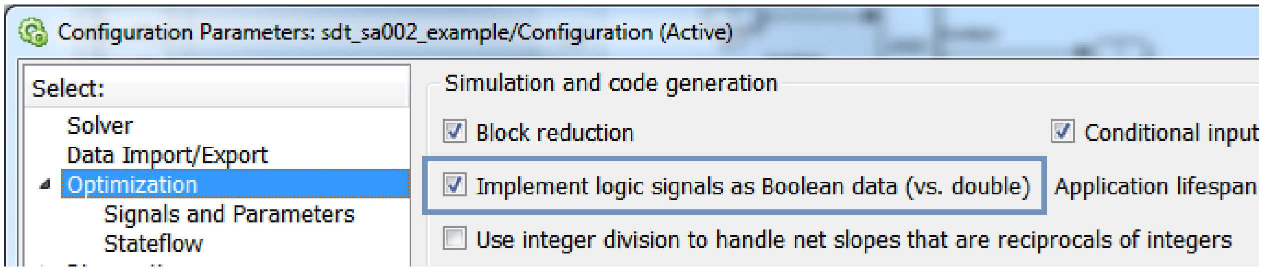
Relational Operator

Applies the selected relational operator to the inputs and outputs the result. The top input corresponds to the first operand.

☑ Require all inputs to have the same data type

Output data type: Inherit: Logical (see Configuration Parameters: Optimization)

Figure sdt_sa002-3: The data type of the output of the Relational Operator block is set to Logical.


Configuration Parameters: sdt_sa002_example/Configuration (Active)

Simulation and code generation

☑ Block reduction

☑ Conditional input

☑ Implement logic signals as Boolean data (vs. double)

Application lifespan

☐ Use integer division to handle net slopes that are reciprocals of integers

Figure sdt_sa002-4:The optimization property "Implement logic signals as Boolean data" is activated in the configuration parameters of the model.

Related Guidelines: ds_0025, jc_0011, hisl_0017, hisl_0018

Model Type: functional model

ISO 26262-6 Mapping: T1.1c Enforcement of Strong Typing, T8.1g No Implicit Type Conversion

mes_sltl_002 - Comparison of Floating-Point Signals

Copyright: Model Engineering Solutions GmbH (MES)

Priority: strongly recommended

Objective: functionality, simulation, portability

MATLAB Versions: all

TargetLink Versions: all

Embedded Coder Versions: all

Automatically Checkable: fully

Manual Review Required: no

Description:

Floating-point numbers must not be compared for equality "==" or inequality "~=" / "!=". Instead, the equality or inequality of floating-point numbers must be tested by comparing the difference of the floating-point numbers against a threshold value.

Rationale:

The internal representation of floating-point numbers may lead to rounding effects. Even though two floating-point numbers are obviously equal, the equality comparison leads to a negative result because the comparison operation considers all digits of each number. The direct comparison for equality or inequality may thus lead to unexpected results.

Counter Example:

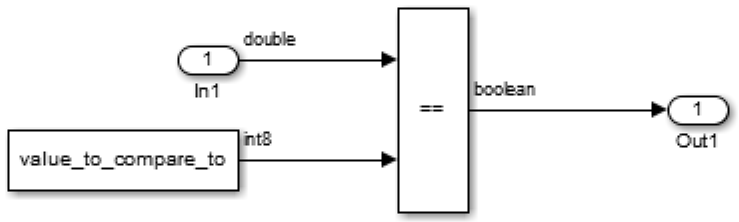


Figure mes_sltl_002-1: The value of Inport In1 is directly compared with the value of variable value_to_compare_to for equality.

Model Type: functional model, implementation model

ISO 26262-6 Mapping: T1.1e Use of Established Design Principles

4 Initialization of Conditionally Executed Subsystems and States

Conditionally executed subsystems, conditionally routed signals (merge blocks), block states, and state diagrams must always be initialized in a comprehensible, correct, and consistent manner. This applies to internal and output values.

Correct initialization comprises of:

- Checking of definition and propagation of default values (type checking, if necessary)
- Definition and recognizability of default paths (switch case blocks)
- Initialization of output values from conditionally executed subsystems and merge blocks
- Existence of default transitions in state diagrams
- Recognition of non-disjoint transition conditions in state diagrams
- Safeguarding of consistent initialization in the model simulation (Simulink) and generated code (TargetLink)

sdt_ic002 - Initial Value of the Output of Conditional Subsystems

Copyright: Model Engineering Solutions GmbH

Priority: recommended

Objective: functionality, reliability

Source: MES_FS_1_3

MATLAB Versions: all

TargetLink Versions: all

Embedded Coder Versions: all

Automatically Checkable: fully

Manual Review Required: no

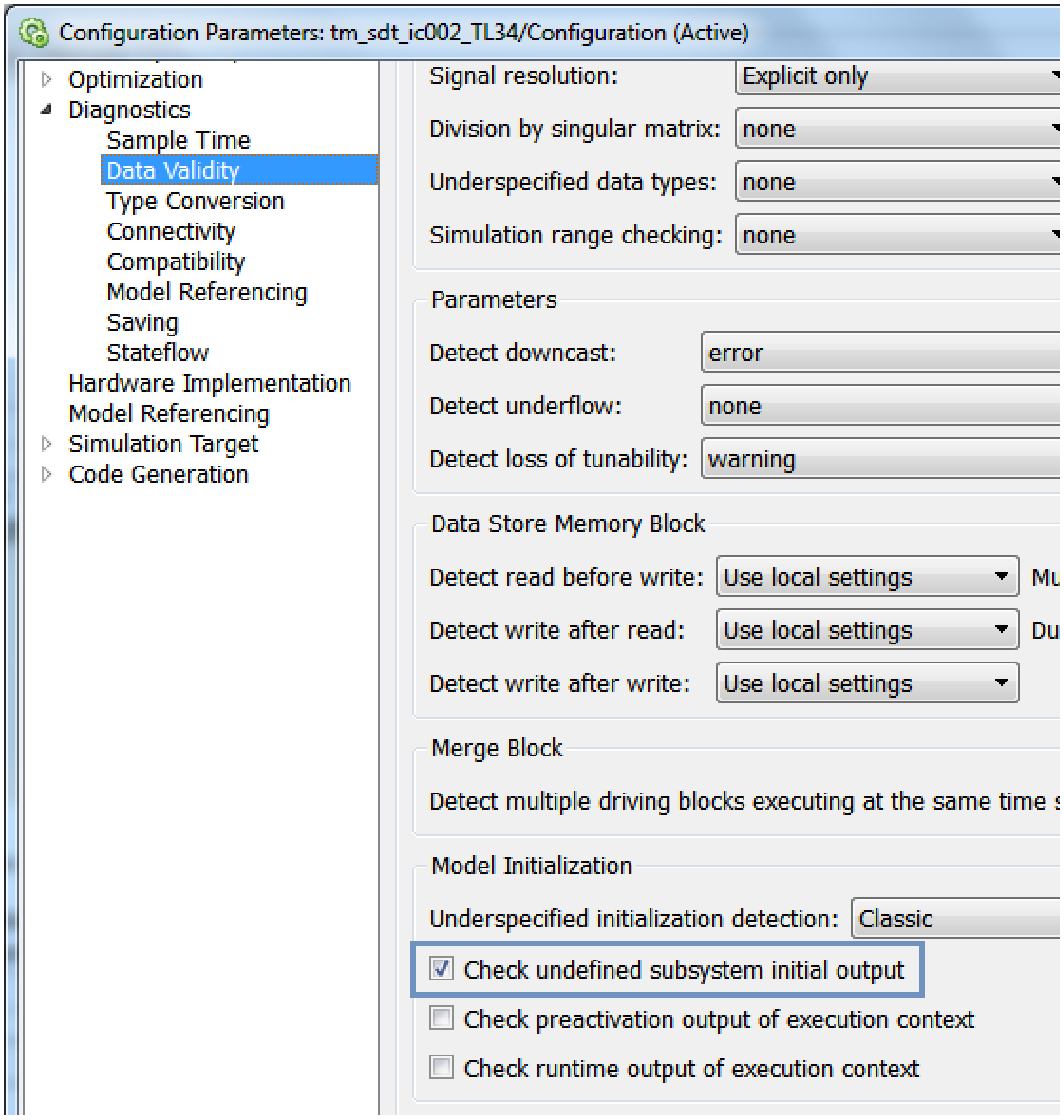
Description:

a) The initial output value of each Outport in a conditionally executed subsystem must be specified.
b) The diagnostic setting "Check undefined subsystem initial output" is to be set.

Rationale:

An undefined initial output value of an Outport in a conditionally executed subsystem may cause different simulation behavior between Simulink and TargetLink, because Simulink only propagates initial values if they are requested, whereas TargetLink always propagates initial values. Moreover, unassigned initial values violate MISRA C Rule 30, stating that all variables should be assigned a value before being used.

Example:



Configuration Parameters: tm_sdt_ic002_TL34/Configuration (Active)

- Optimization
- Diagnostics
  - Sample Time
  - Data Validity
  - Type Conversion
  - Connectivity
  - Compatibility
  - Model Referencing
  - Saving
  - Stateflow
- Hardware Implementation
- Model Referencing
- Simulation Target
- Code Generation

Signal resolution:	Explicit only
Division by singular matrix:	none
Underspecified data types:	none
Simulation range checking:	none

Parameters

Detect downcast:	error
Detect underflow:	none
Detect loss of tunability:	warning

Data Store Memory Block

Detect read before write:	Use local settings
Detect write after read:	Use local settings
Detect write after write:	Use local settings

Merge Block

Detect multiple driving blocks executing at the same time s

Model Initialization

Underspecified initialization detection: Classic

[✓] Check undefined subsystem initial output

[ ] Check preactivation output of execution context

[ ] Check runtime output of execution context

Figure sdt_ic002-1: The diagnostic setting "Check undefined subsystem initial output" is set.

Figure sdt_ic002-2 illustrates the difference between how Simulink (model-in-the-loop, MIL) and TargetLink (software-in-the-loop, SIL) handles non-initializated output values. The model contains an enabled subsystem that is activated by a step function at t=5. It outputs the constant value 3. Importantly, the enabled subsystem does not define an initial output value. In MIL mode, the output is set to zero before the subsystem is enabled as shown in the scope. In contrast, in TargetLink SIL mode, the constant 3 is used as the initial output value.

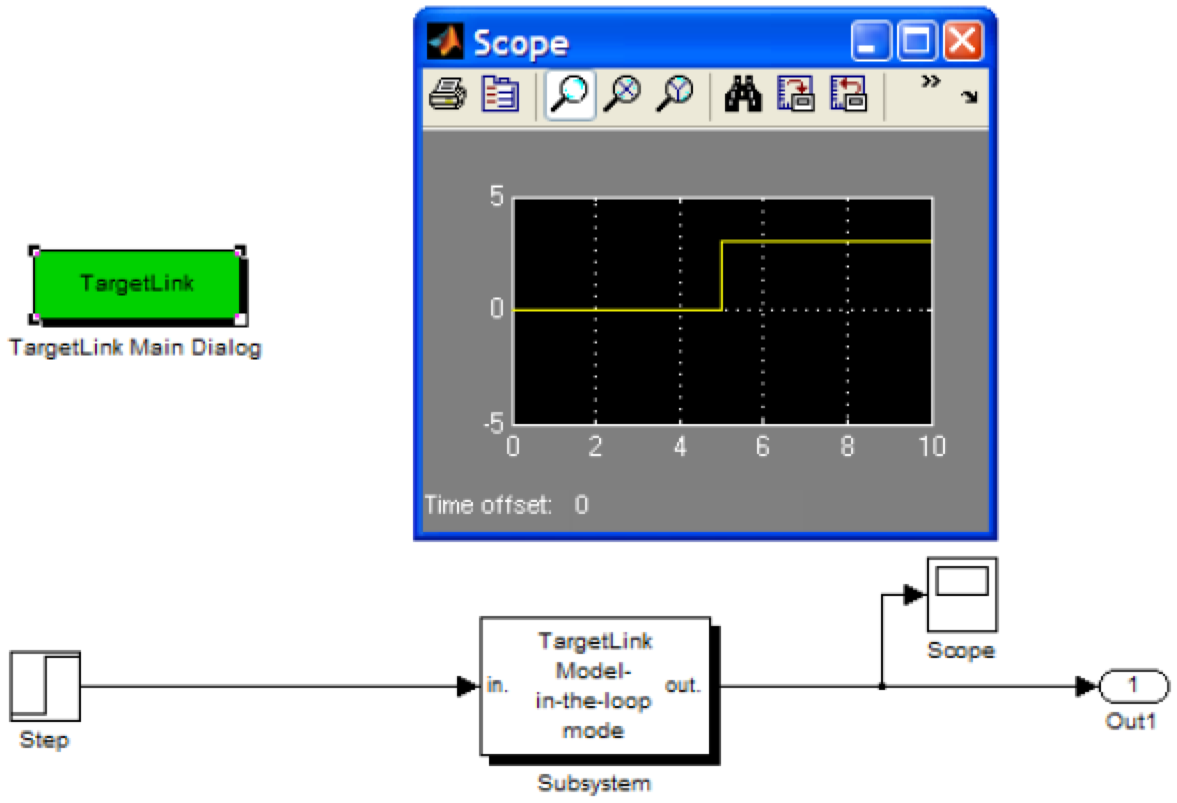


Figure sdt_ic002-2: Model-in-the-loop simulation.

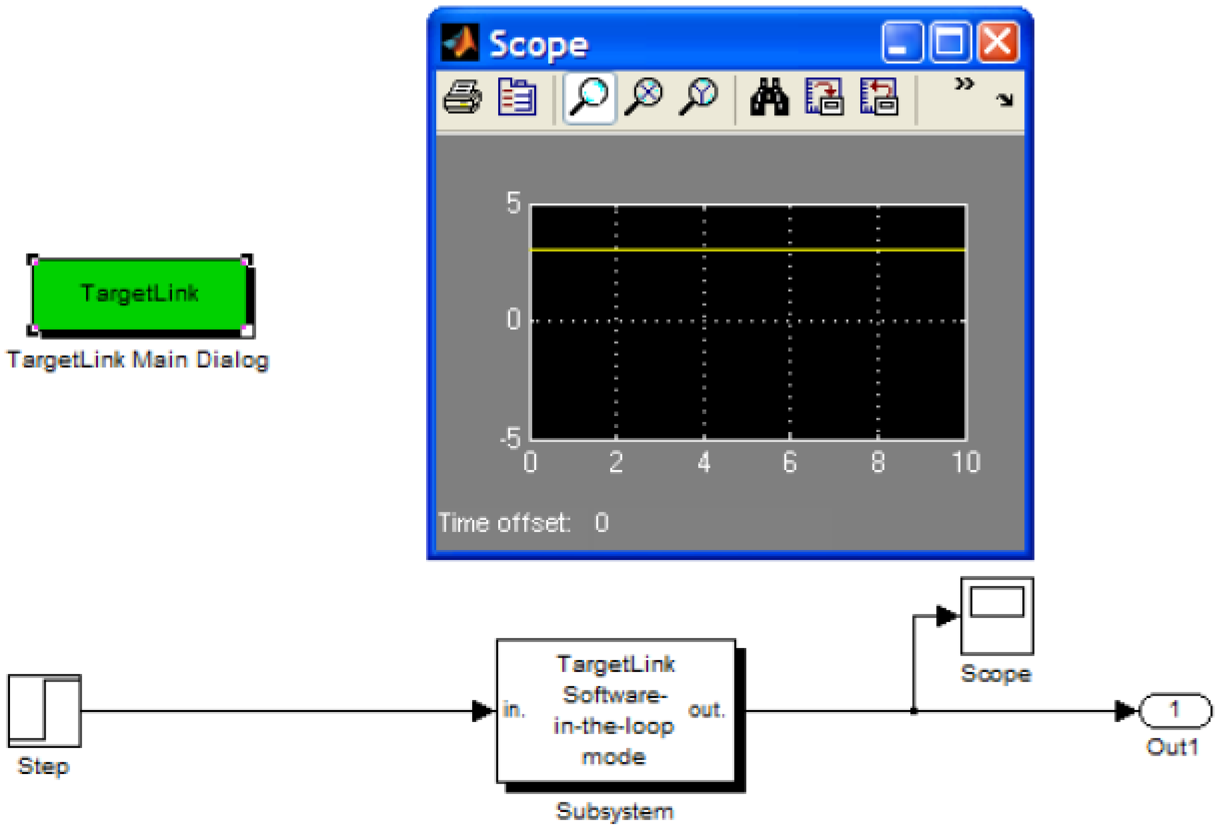


Figure sdt_ic002-3: Software-in-the-loop simulation.

Related Guidelines: MISRA Rule 30: Variables must be assigned a value, before being used

Model Type: functional model

ISO 26262-6 Mapping: T8.1c Initialization of Variables

sdt_ic001 - Initial Value of the Output of Merge Blocks

Copyright: Model Engineering Solutions GmbH

Priority: recommended

Objective: functionality, reliability

Source: MES_FS_1_3

MATLAB Versions: all

TargetLink Versions: all

Embedded Coder Versions: all

Model Engineering Solutions GmbH

Automatically Checkable: fully

Manual Review Required: no

Description:

For each Merge block, an initial output value must be explicitly defined. Inheriting the initial value by specifying '[]' is not allowed.

Rationale:

If the initial output value of a Merge block is not explicitly defined and more than one source block of the Merge block defines an initial value, Simulink arbitrarily selects one of these initial values. Here, Simulink seems to use the source block which was created at last. Thus, an undefined (inherited) initial value of a Merge block may cause different simulation behavior between MIL and SIL.

Counter Example:

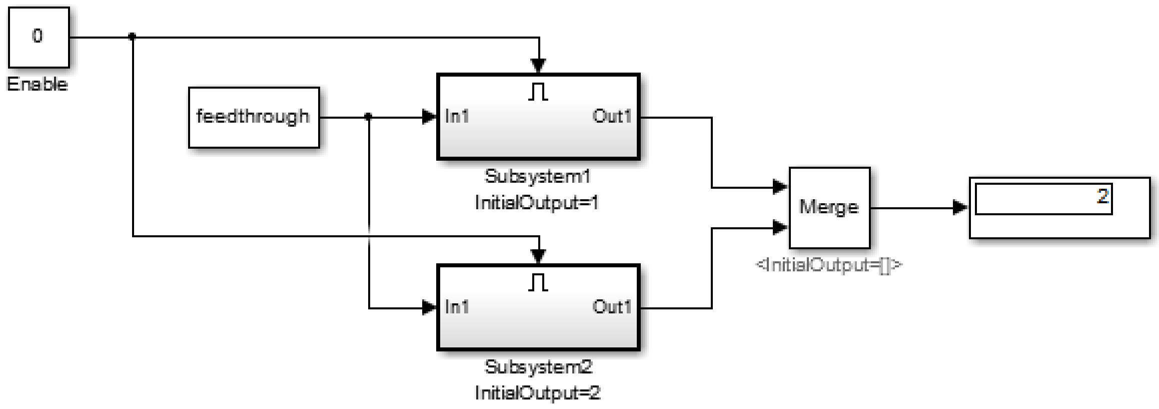


Figure sdt_ic001-1: Arbitrary inheritance of an initial value because of different initial values of the Merge block input signals. The Merge block has no initial value defined and Simulink inherits its initial value from Subsystem2, even if both sources are not enabled.

Model Type: functional model

ISO 26262-6 Mapping: T8.1c Initialization of Variables

sdt_ic003 - Default Paths of Switch Case Blocks

Copyright: Model Engineering Solutions GmbH

Priority: recommended

Objective: code generation, readability

Source: MES FS 1.3

MATLAB Versions: all

TargetLink Versions: all

Embedded Coder Versions: irrelevant

Automatically Checkable: fully

Manual Review Required: no

Description:

The "Show default case" option of each Switch case block must be set and the default case must be properly connected.

Rationale:

All Switch Case instructions in the generated code should have a default branch to avoid undefined states during program execution.

Example:

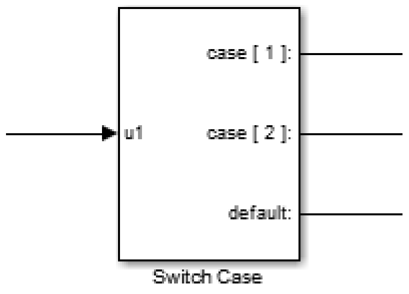
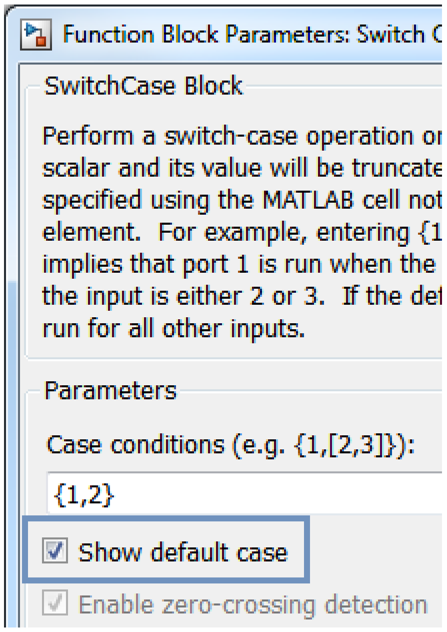


Figure sdt_ic003-1: A Switch Case block with a default branch


Function Block Parameters: Switch C

SwitchCase Block

Perform a switch-case operation or
scalar and its value will be truncate
specified using the MATLAB cell not
element. For example, entering {1
implies that port 1 is run when the
the input is either 2 or 3. If the de
run for all other inputs.

Parameters

Case conditions (e.g. {1,[2,3]}):

{1,2}

☑ Show default case

☑ Enable zero-crossing detection

Figure sdt_ic003-2: The option "Show default case" of a Switch Case block is selected

Related Guidelines: hisl_0011 A, hisl_0011 B, dSPACE ds_0037 (Use of the Switch Case block)

Model Type: functional model

ISO 26262-6 Mapping: T1.1d Use of Defensive Implementation Techniques, T1.1e Use of Established Design Principles

sdt_ic004 - Else Paths of If Blocks

Copyright: Model Engineering Solutions GmbH

Priority: strongly recommended

Scope: SDT

Objective: code generation, readability

Source: MES FS 1.3

MATLAB Versions: all

TargetLink Versions: all

Embedded Coder Versions: irrelevant

Automatically Checkable: fully

Manual Review Required: no

Description:

The "Show else condition" option for each If block must be set and the else path must be properly connected.

Rationale:

All If instructions in the generated code should have an else branch to avoid undefined states during program execution.

Example:

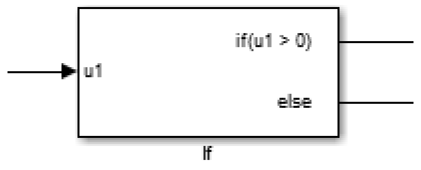
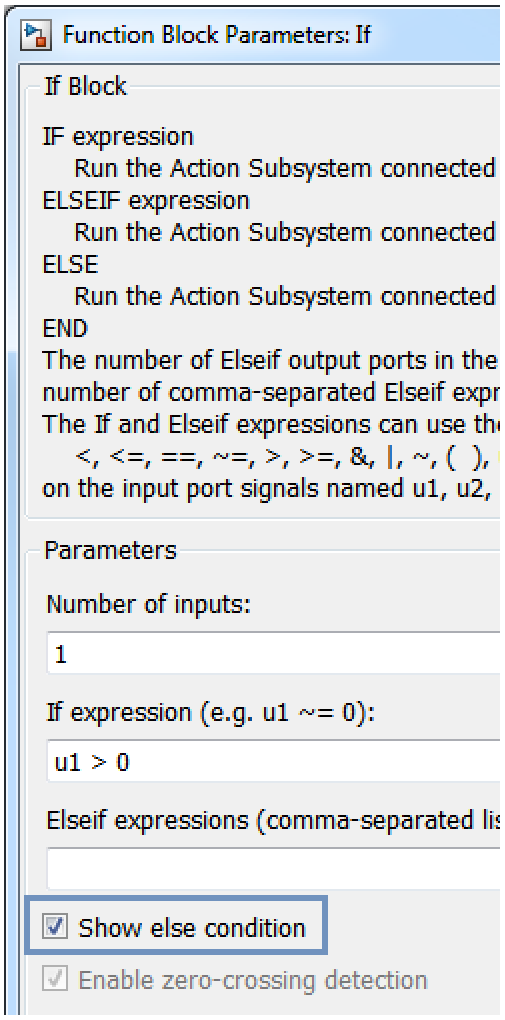


Figure sdt_ic004-1: An If block with an else branch


Function Block Parameters: If

If Block

IF expression
    Run the Action Subsystem connected
ELSEIF expression
    Run the Action Subsystem connected
ELSE
    Run the Action Subsystem connected
END
The number of Elseif output ports in the
number of comma-separated Elseif expr
The If and Elseif expressions can use th
    <, <=, ==, ~=, >, >=, &, |, ~, ( ),
on the input port signals named u1, u2,

Parameters

Number of inputs:
1

If expression (e.g. u1 ~= 0):
u1 > 0

Elseif expressions (comma-separated lis

[✓] Show else condition

[✓] Enable zero-crossing detection

Figure sdt_ic004-2: The option "Show else condition" of an If block is selected

Related Guidelines: hisl_0010

Model Type: functional model

ISO 26262-6 Mapping: T1.1d Use of Defensive Implementation Techniques, T1.1e Use of Established Design Principles

5 Interface Layout

Interfaces between system boundaries should be kept as narrow as possible. Duplication of identical signals should be avoided.

Narrow interfaces are not only clearer, they also greatly simplify interface maintenance and documentation. Duplicated signals, on the other hand, can easily lead to misunderstandings, as their common source and the resultant value coupling is no longer identifiable on the other side of the system boundary. Furthermore, they increase the interface overhead unnecessarily (thereby increasing the generated code) without any associated gain in functionality.

sdt_il001 - Avoidance of Duplicated Port Signals

Copyright: Model Engineering Solutions GmbH

Priority: recommended

Objective: usability, functionality, maintainability, reliability

Source: MES_FS_1_3

MATLAB Versions: all

TargetLink Versions: all

Embedded Coder Versions: all

Automatically Checkable: fully

Manual Review Required: no

Description:

The interface of a system must be as small as possible. To this end,

- all Inports and Outports of a system must contain signals from different sources, and
- shadow Inport blocks are not to be used.

Rationale:

To keep the interface of a system as small as possible, duplicate signals must be avoided. Identical signals entering a system increase the overhead of the system interface and lead to inefficient code. Moreover, they may lead to errors if the common source is changed.

Example:

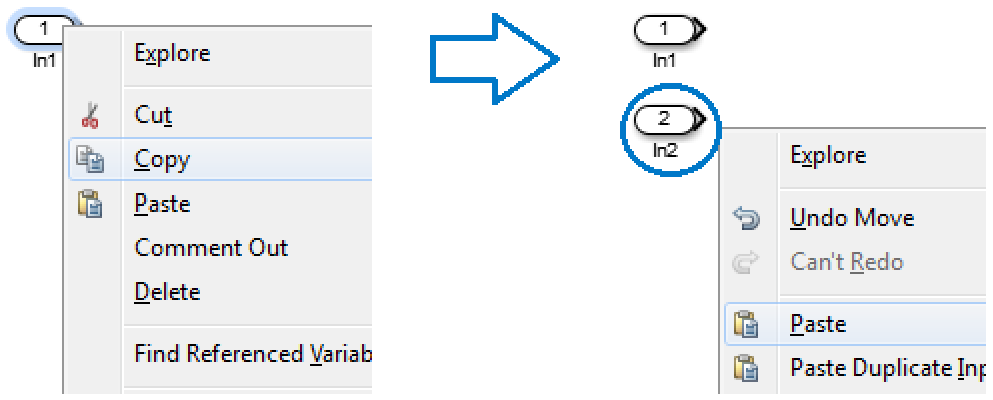


Figure sdt_il001-1: Proper Inport duplication

Counter Example:

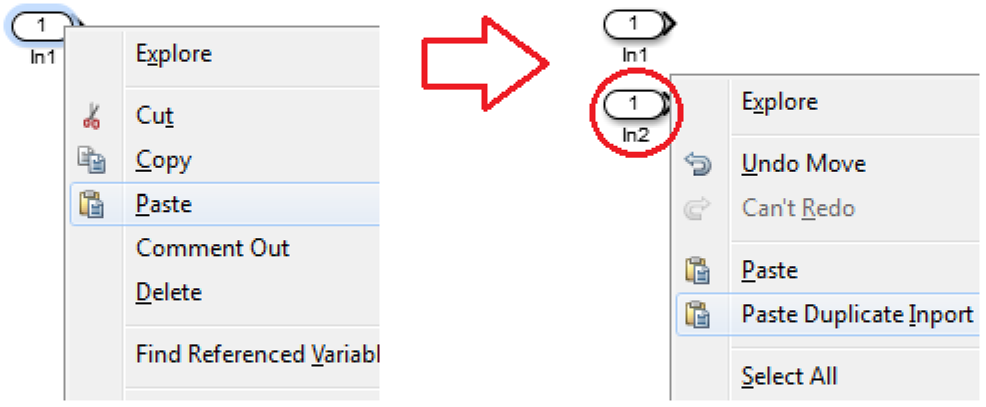


Figure sdt_il001-1: Specification of duplicate Inports

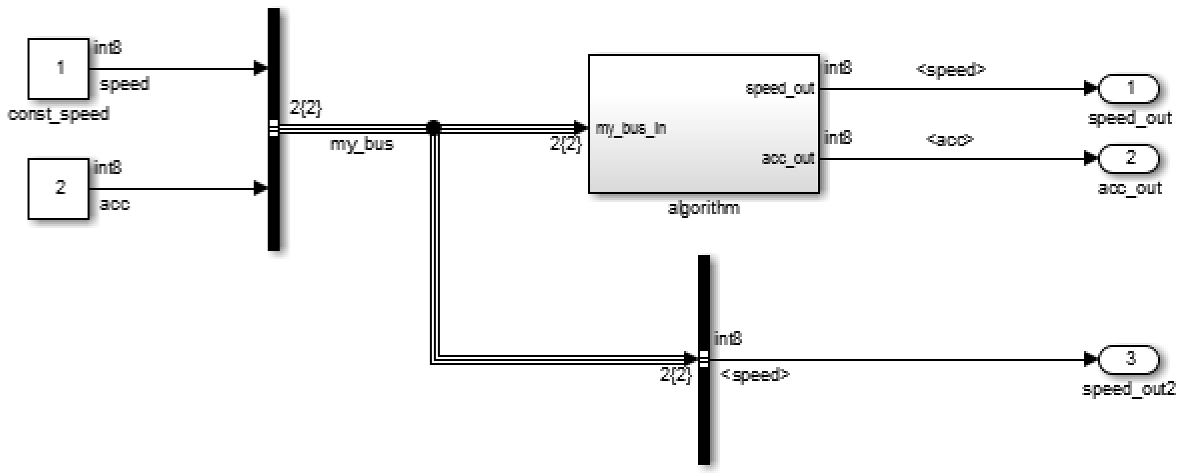


Figure sdt_il001-2: Outports 1 and 3 have the same source (Constant block "const_speed"). These blocks may be removed to decrease the complexity of the system and its interface.

Model Type: functional model

ISO 26262-6 Mapping: T1.1a Enforcement of Low Complexity, T1.1e Use of Established Design Principles

6 Visibility of Data and Signal Flow

mes_sltl_001 - Magic Constants in Constant Blocks

Copyright: Model Engineering Solutions GmbH (MES)

Priority: strongly recommended

Objective: readability, usability, maintainability, reliability

MATLAB Versions: all

TargetLink Versions: all

Embedded Coder Versions: all

Automatically Checkable: fully

Manual Review Required: no

Description:

The value of a Constant block must be entered as a named constant or variable. Expressions entered as value in a Constant block must not contain numerical values.
Exceptions: The values 0 and 1 are allowed.

Rationale:

The use of named constants rather than explicit numerical numbers allows it to be more easily maintained when it is used in multiple locations.Furthermore the rational of the constant is documented by a named constant.

Counter Example:

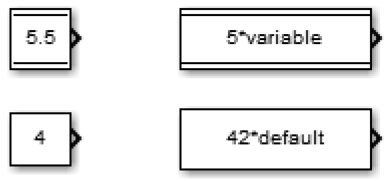


Figure mes_sltl_001: Magic constants entered as values in a Constant block.

Model Type: functional model

ISO 26262-6 Mapping: T1.1e Use of Established Design Principles

sdt_sc005 - Propagation of Signal Names

Copyright: Model Engineering Solutions GmbH

Priority: recommended

Objective: readability, maintainability

Source: MES_FS_1_3

MATLAB Versions: all

TargetLink Versions: all

Embedded Coder Versions: all

Automatically Checkable: fully

Manual Review Required: no

Description:

Signal names should be propagated as far as possible. To this end, the signal propagation mechanism of Simulink should be used to automatically name signals along a sequence of signal lines.

- If a signal name is used along a sequence of signals, it should be named at the signal directly connected to the signal source, and

- this name should be propagated to all subsequently connected signal lines by setting the signal property "Show propagated signals" to "on".

Rationale:

Signal propagation ensures the consistency of signal naming, shows the data flow clearly, and reduces the effort to rename any signals.

Example:

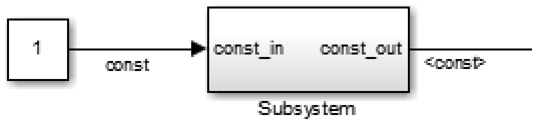
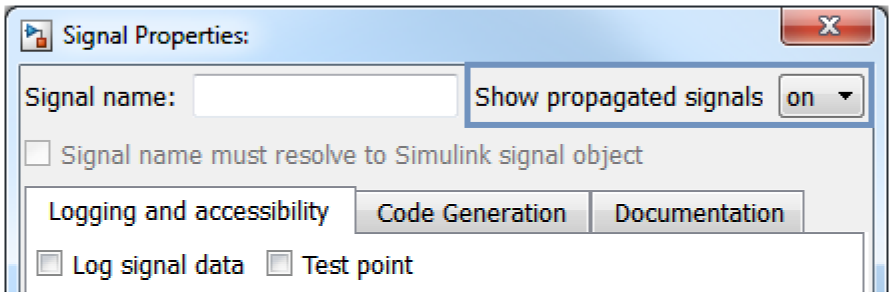


Figure sdt_sc005-1: The signal is explicitly named at its source. That signal name is then propagated to any subsequently connected signal lines.


Signal Properties:

Signal name: [ ]

Show propagated signals: on

☐ Signal name must resolve to Simulink signal object

☐ Log signal data ☐ Test point

Figure sdt_sc005-2:The option "Show propagated signals" is set to "on" to show the propagated signal name.

Model Type: functional model

ISO 26262-6 Mapping: T1.1e Use of Established Design Principles

sdt_sc006 - Signal Naming and Propagation

Copyright: Model Engineering Solutions GmbH

Priority: strongly recommended

Scope: SDT

Objective: readability, maintainability

Source: MES FS 1.3

MATLAB Versions: all

TargetLink Versions: all

Embedded Coder Versions: irrelevant

Automatically Checkable: fully

Manual Review Required: no

Description:

Each output signal of a subsystem must be named at its source signal. That signal name must then be propagated to the Outport block.

Rationale:

Signal propagation ensures the consistency of signal naming, shows the data flow clearly, and reduces the effort to rename signals.

Example:

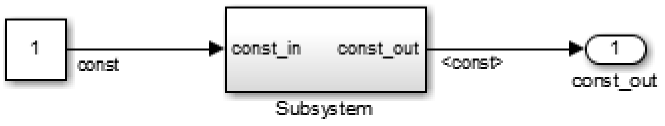
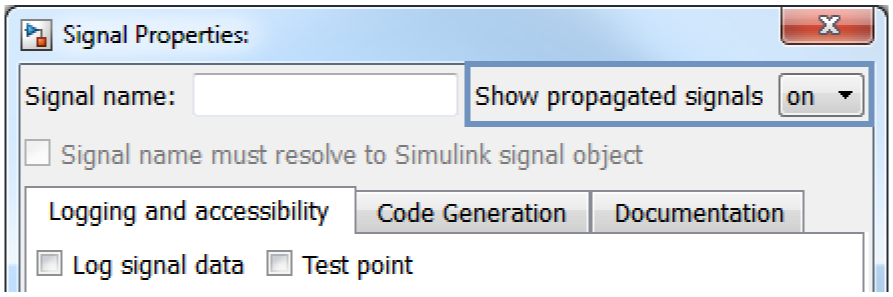


Figure sdt_sc006-1: The signal name at the input of the Outport is propagated from its source.


Signal Properties:
Signal name must resolve to Simulink signal object
Logging and accessibility
Code Generation
Documentation
Log signal data
Test point

Signal name:

Show propagated signals: on

Figure sdt_sc006-2:The option "Show propagated signals" is set to "on" to show the propagated signal name.

Model Type: functional model

ISO 26262-6 Mapping: T1.1e Use of Established Design Principles

7 Stateflow Transition Paths

mes_sf_001 - No Loops in Multi Segmented Transitions

Copyright: Model Engineering Solutions GmbH

Priority: recommended

Scope: MES

Objective: workflow, functionality

MATLAB Versions: all

TargetLink Versions: irrelevant

Embedded Coder Versions: irrelevant

Automatically Checkable: partly

Manual Review Required: yes

Description:

In a Stateflow chart, multi segmented transitions that model a possibly non terminating loop are not allowed. It must be ensured that each condition that terminates a loop is eventually evaluated to true. As such each loop can be left.

The Stateflow pattern wizard generates For, While and DoWhile loop patterns. Without special precaution these patterns may violate this guideline.

Rationale:

A loop that consists of multi segmented transitions may lead to an infinite cycle at runtime if its terminating condition is never going to be evaluated to true.

Counter Example:

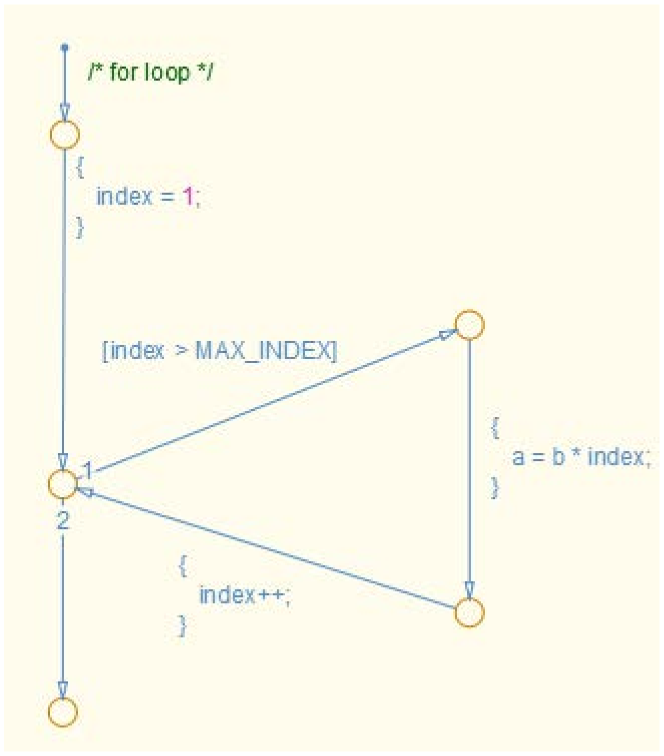


Figure mes_sf_001-1: A For loop that can never be left if MAX_INDEX is set to a value that is less than one. In that case, the terminating condition "index <= MAX_INDEX" of the loop can never be fulfilled.

Model Type: functional model

ISO 26262-6 Mapping: T1.1b Use of Language Subset, T1.1d Use of Defensive Implementation Techniques

mes_sf_002 - Variable Assignments in Transition Segments

Copyright: Model Engineering Solutions GmbH

Priority: strongly recommended

Scope: MES

Objective: workflow, functionality, reliability

MATLAB Versions: all

TargetLink Versions: irrelevant

Embedded Coder Versions: irreleveant

Automatically Checkable: fully

Manual Review Required: no

Description:

Within a multi segmented transition path, a variable is only to be assigned a value on a transition action or a condition action of the last segment of the transition path. The last segment is the transition whose destination is a state or a junction with no outgoing transitions.

Rationale:

Transitions that lead from a source object to a target object are often composed of multiple segments. These segments usually consist of transitions and connective junctions. The resulting transition paths are evaluated segment by segment. If the evaluation of a segment fails, the Stateflow backtracking mechanism comes into play. Hence, another segment is chosen and evaluated until a complete, valid transition path is found. However, Stateflow does not define its backtracking function to actually revoke actions of a valid segment when the complete transition path is invalid. Instead, the assignment of the interim result is preserved.

These side effects during evaluation of the transition graph are hard to understand and hard to maintain.

Example:

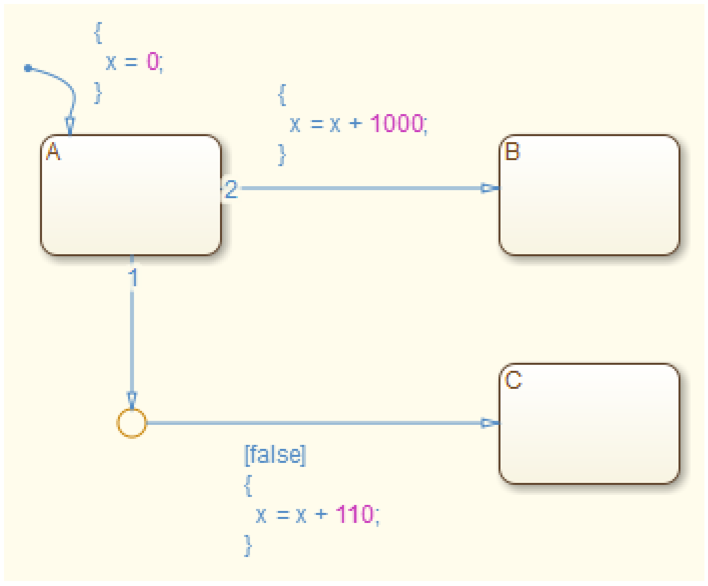


Figure mes_sf_002-1: Variable x is assigned a value on the last transition segment of each transition path. The resulting value of x is 1000.

Counter Example:

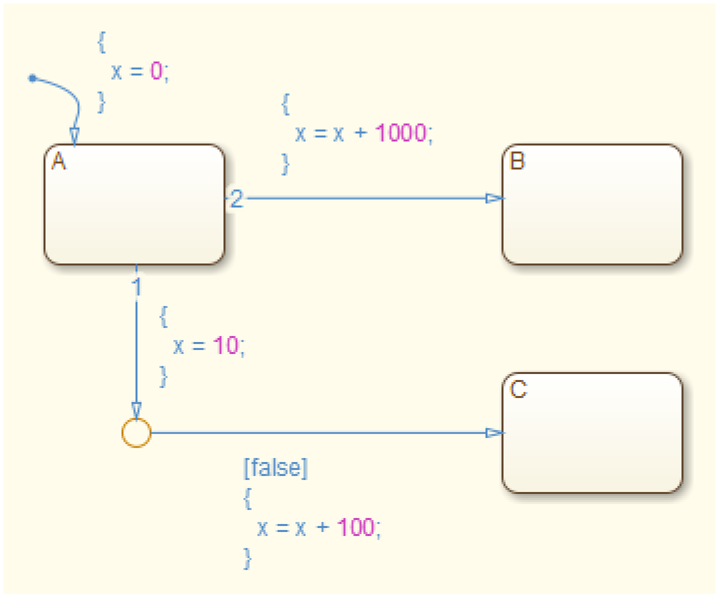


Figure mes_sf_002-2: Variable x is assigned a value on a transition segment that is first evaluated, but then being backtracked as the condition of the last transition of that transition path is false. Therefore, the resulting value of x is 1010 instead of 1000. Figure mes_sf_002-1 shows a corrected version of this example.

Model Type: functional model

ISO 26262-6 Mapping: T1.1b Use of Language Subset, T1.1e Use of Established Design Principles

8 Interaction of Stateflow States with Events

mes_sf_003 - Interaction between Parallel States

Copyright: Model Engineering Solutions GmbH

Priority: strongly recommended

Scope: MES

Objective: workflow, functionality

MATLAB Versions: all

TargetLink Versions: irrelevant

Embedded Coder Versions: irrelevant

Automatically Checkable: fully

Manual Review Required: no

Description:

In a Stateflow chart, any two parallel states that directly influence each other by events must be either a sender of events or a receiver of events. Such a state may not be both a sender and a receiver.

Rationale:

Restricting any two parallel states to being either a sender of events or a receiver of events assures that the sending and the receiving of these events is done in the same cycle and collisions with other actions of the next cycle are avoided.

Example:

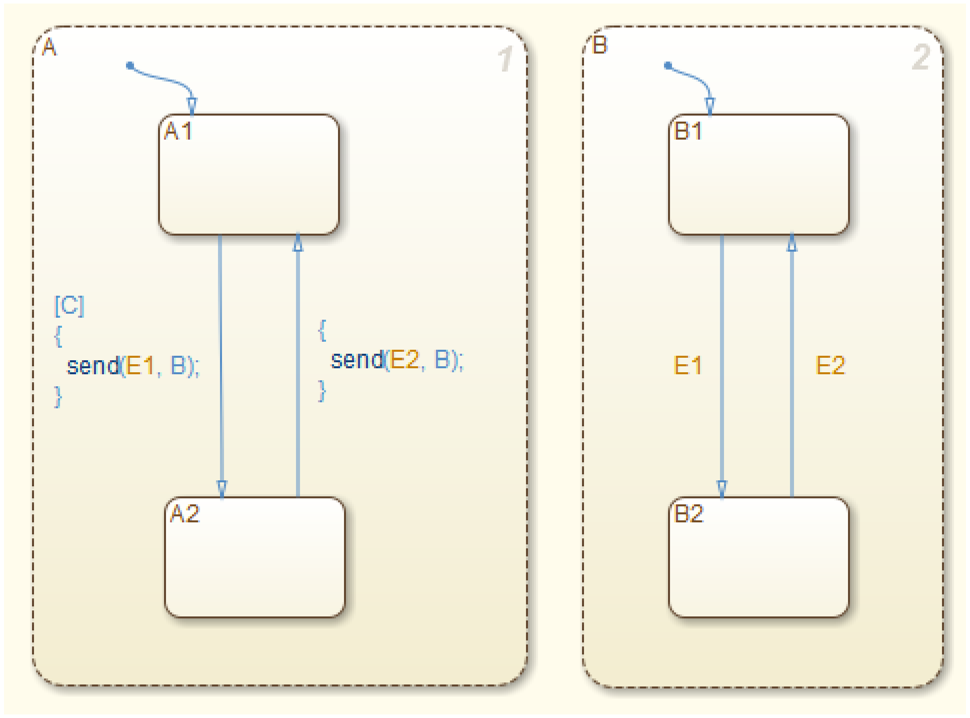


Figure mes_sf_003-1: The two parallel states A and B communicate with each other by events. State A only sends events while state B only receives the events that are sent by A.

Model Type: functional model

ISO 26262-6 Mapping: T1.1b Use of Language Subset, T1.1d Use of Defensive Implementation Techniques

mes_sf_004 - Freeness of Early Return Logic Problems

Copyright: Model Engineering Solutions GmbH

Priority: recommended

Scope: MES

Objective: code generation, verification/validation

Reference: Stateflow > User Guide > Stateflow Chart Programming > Chart Simulation Semantics > Early Return Logic for Event Broadcasts

MATLAB Versions: all

TargetLink Versions: irrelevant

Embedded Coder Versions: irrelevant

Automatically Checkable: fully

Manual Review Required: no

Description:

Each Stateflow chart must be free of any possible early return logic problem.

Rationale:

Stateflow charts run on a single thread. Therefore, each chart must interrupt its current activity to process an event. Any activity that is based on an event broadcast from a state or transition action may conflict with the current activity. To this end, the Stateflow semantics makes use of the early return logic. That behavior may lead to unexpected infinite cycles, elements of the chart that are never reached or actions that are never going to be evaluated.

Example:

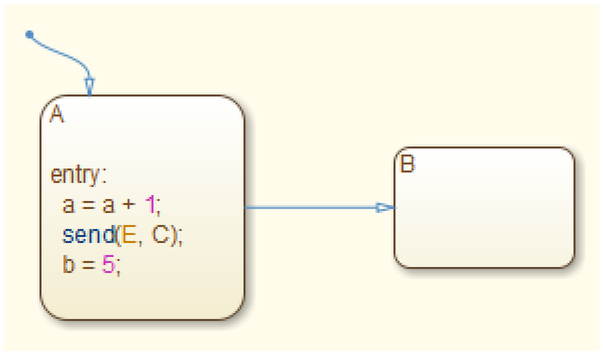


Figure mes_sf_004-1: In contrast to Figure mes_sf_004-3, action "b=5;" is executed because event E is sent as a directed event broadcast.

Counter Example:

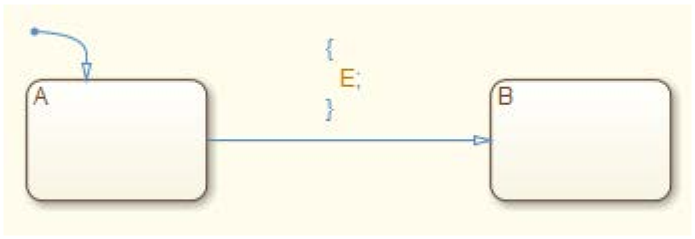


Figure mes_sf_004-2: Due to the early return logic, state A is infinitely often activated and state B is never reached.

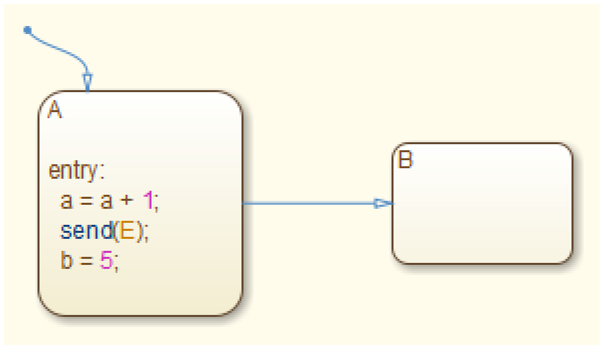


Figure mes_sf_004-3: The action "b=5;" is never executed because of the early return logic implemented as event E is send.

Model Type: functional model

ISO 26262-6 Mapping: T1.1b Use of Language Subset, T1.1d Use of Defensive Implementation Techniques

9 Configuration

mes_sk_001 - Configuration Parameters for Simulink

Copyright: Model Engineering Solutions GmbH

Priority: recommended

Scope: MES

Objective: verification/validation, maintainability, simulation

MATLAB Versions: all

TargetLink Versions: all

Embedded Coder Versions: irrelevant

Automatically Checkable: fully

Manual Review Required: no

Description:

Each project must define an appropriate setting for each Simulink Configuration parameter. These settings must be used for each model within that project.

Rationale:

Conformance to the guideline supports reproducible simulation results and eases the comparison of simulations and tests.

Model Type: functional model

ISO 26262-6 Mapping: T1.1e Use of Established Design Principles

mes_sk_002 - Advanced Parameters and Configuration Settings for TargetLink

Copyright: Model Engineering Solutions GmbH

Priority: recommended

Scope: MES

Objective: code generation, verification/validation, functionality, maintainability

MATLAB Versions: all

TargetLink Versions: all

Embedded Coder Versions: irrelevant

Automatically Checkable: fully

Manual Review Required: no

Description:

Each project must define an appropriate setting for each TargetLink Advanced parameter and configuration setting. These settings must be used for each model within that project.

Rationale:

Conformance to the guideline supports reproducible simulation results and eases the comparison of simulations and tests. Further it ensures that code generator leads to comparable results.

Model Type: implementation model

ISO 26262-6 Mapping: T1.1e Use of Established Design Principles

mes_is_0002 - User-specified State/transition Execution Order

Copyright: Model Engineering Solutions GmbH

Priority: mandatory

Scope: MES

Objective: code generation, readability, verification/validation, workflow

MATLAB Versions: all

TargetLink Versions: all

Embedded Coder Versions: irrelevant

Automatically Checkable: fully

Manual Review Required: no

Description:

The execution order of parallel states and of transitions must be specified by the user, otherwise misinterpretations w.r.t. the semantics of the chart may occur. For that purpose it is mandatory to activate the Stateflow chart option 'User specified state/transition execution order'.

Rationale:

The execution order of parallel states should not depend on the geometrical arrangement of the states in the chart. Therefore, to avoid unwanted side effects and to assure the desired functionality/interaction of the states the user has to set the execution order explicitly. If the option "User specified state/transition execution order" is not set, Stateflow will assign execution orders according to the geometrical arrangement of the states within a chart.

Related Guidelines: hisf_0002 A

Model Type: functional model