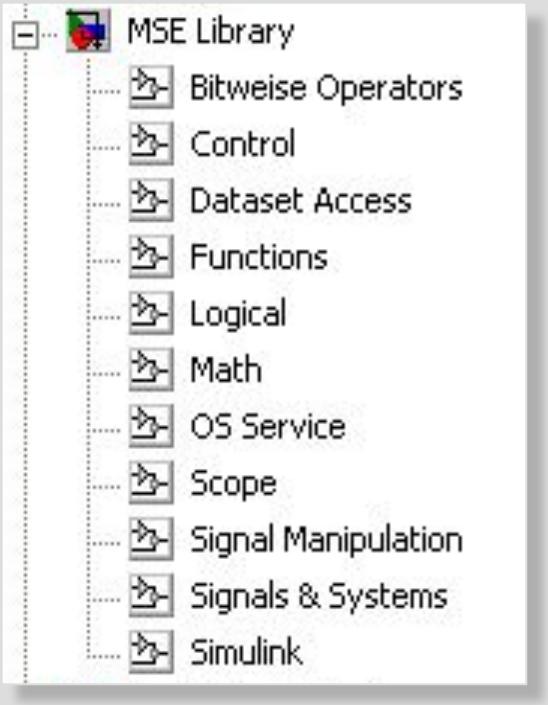
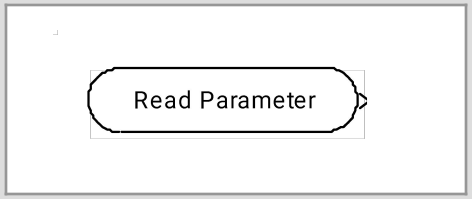
MSE block library for Simulink

Library – a function description language for an engineer



- Minimum set of elementary operations for composing control functions
- Integration with the data preparation system
- Automatic data scaling in operations
- Automatic data conversion into binary form and vice versa
- Advanced diagnostics of data using accuracy
- Automatic generation of efficient code

Dataset Access Section



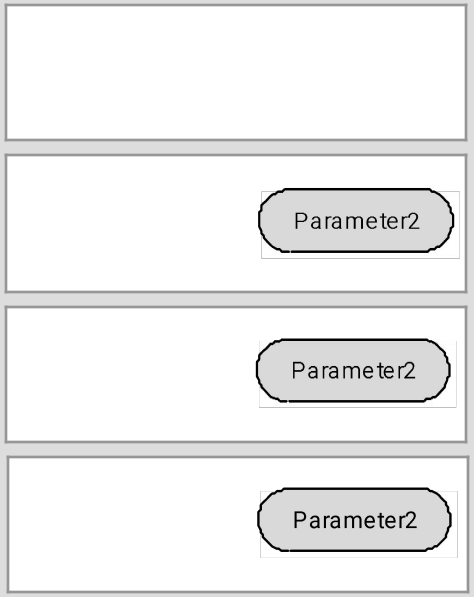
Read Parameter

A universal function for accessing a parameter. When reading tables it makes interpolation.

Name	Type	Bit resolution
Parameter1	RAM-scalar	1
Parameter2	RAM-scalar	1
Parameter3	RAM-scalar	0.3
Vector1	ROM-vector	1
Matrix1	ROM-matrix	1

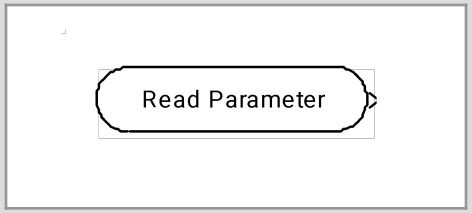
Simulink model

C code




```
Parameter2_DA = Parameter1_DA;

Parameter2_DA = (int16_T)(((int32_T)Parameter3_DA) * 3 / 10 );

Parameter2_DA = Vector1_DA;

Parameter2_DA = Matrix1_DA;
```

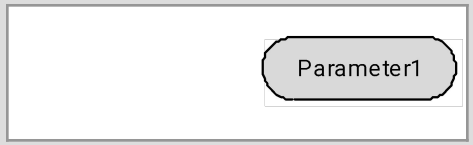
Dataset Access Section



Read Parameter without Interpolation

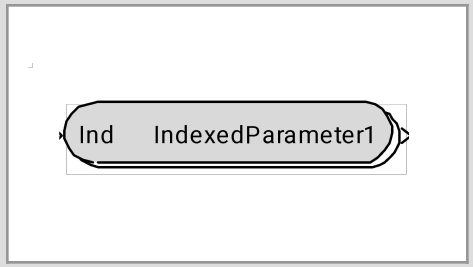
Extracts data from the nearest node of a parameter-table. It's used for calculation of dynamic table data.

Simulink model



C code

```
Parameter1_DA = Vector1_DA;
```



Read Indexed Parameter

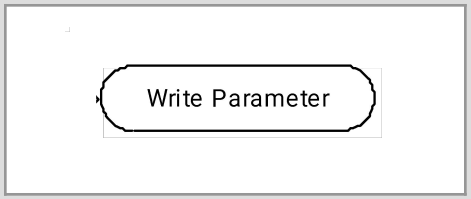
Reads parameters representing linear data arrays, for example corresponding to a number of engine cylinders. It's similar to the "Read Parameter" block.

Simulink model

C code

```
Parameter1_DA = IndexedParameter1_DA[Index_DA];
```

Dataset Access Section

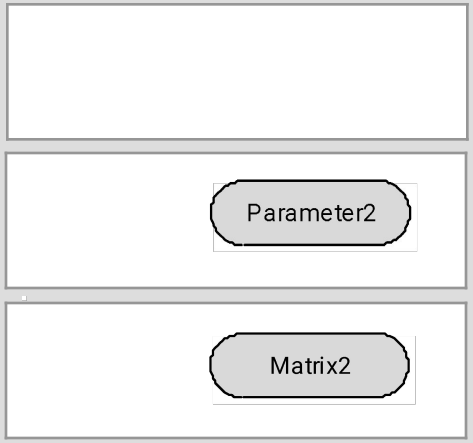


Write Parameter

Writes a value into a specified parameter.

Name	Type	Bit resolution
Parameter1	RAM-scalar	1
Parameter2	RAM-scalar	0.3
VarMatrix1	RAM-matrix	1

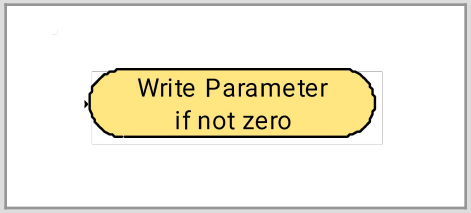
Simulink model



C code

```
Parameter1_DA = (int16_T)7;

Parameter2_DA = (int16_T)(7 * 10 / 3 );

Matrix2_DA = (int16_T)7;
```

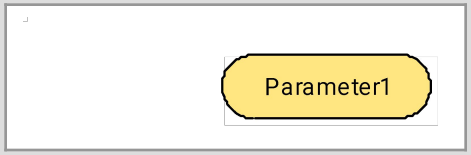
Dataset Access Section



Write if not zero

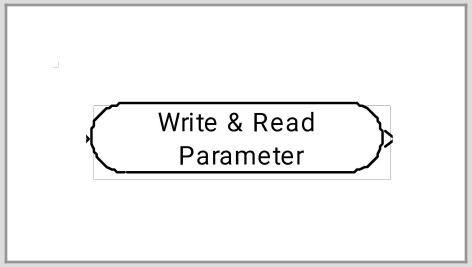
Writes into a parameter all but zero values. It's often used for writing codes of errors.

Simulink model



C code

```
int16_T tmpVar_sInt16_0;
tmpVar_sInt16_0 = (int16_T)7;
if (tmpVar_sInt16_0 != 0) Parameter1_DA = tmpVar_sInt16_0;
```



Write & Read Parameter

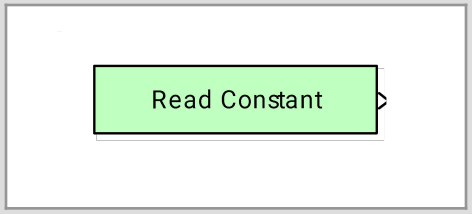
Checks the presence of external control over a parameter. If it's controlled, writes external data into the parameter. Otherwise writes an input signal value.

Simulink model

C code

```
if (Parameter2_DA_KEY)
    Parameter2_DA_EXT = Parameter1_DA;
else
    Parameter2_DA = Parameter1_DA;
Parameter3_DA = Parameter2_DA;
```

Dataset Access Section



Compile Time Constant

A constant value from a parameter set which doesn't require place in system's memory.

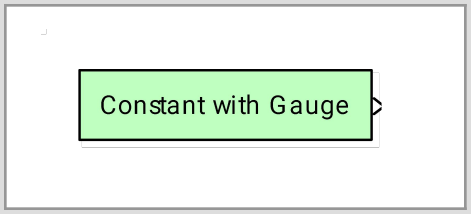
Name	Type	Bit resolution
Voltage	RAM-scalar	4.89 mV
SupplyVoltage	Constant	4.89 mV
RefVoltage	Constant	19.6 mV

Simulink model

C code

```
Voltage_DA = CONST_SupplyVoltage;
```

```
Voltage_DA = ((int16_T)CONST_RefVoltage) * 341 / 85;
```



Constant with Bit Gauge

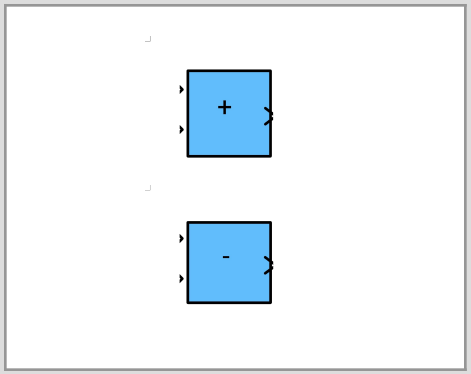
A constant that isn't included in a parameters set. It's used inside a model only.

Simulink model

C code

```
Voltage_DA = 675;
```

Math Section

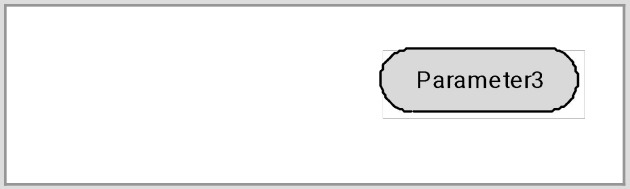


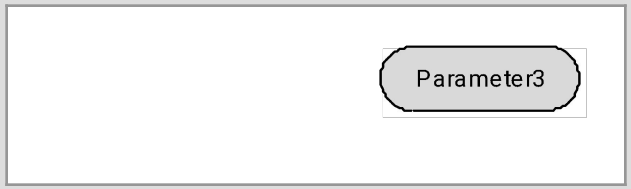
Add / Subtract

Adds (subtracts) values of input parameters.
Allows operations only for parameters having the same combination of its basic units.
Automatically chooses the best representation for the output signal.
During modeling makes diagnosis of arithmetic overflow.

Name	Bit resolution
Parameter1	1
Parameter2	0.3
Parameter3	0.3

Simulink model



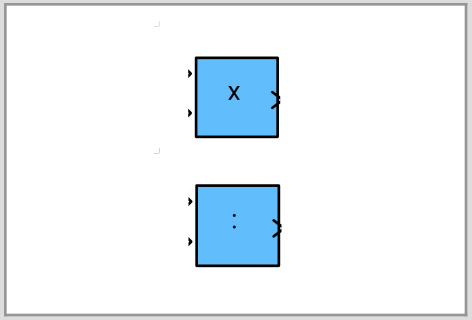


C code

```
Parameter3_DA = (int16_T)(((int32_T)( Parameter1_DA +
(int16_T)(((int32_T)Parameter2_DA) * 3 / 10 ))) * 10 / 3 );
```

```
Parameter3_DA = (int16_T)(((int32_T)( Parameter1_DA –
(int16_T)(((int32_T)Parameter2_DA) * 3 / 10))) * 10 / 3 );
```

Math Section

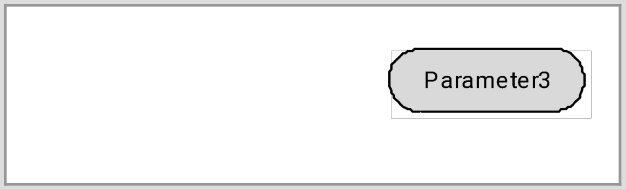


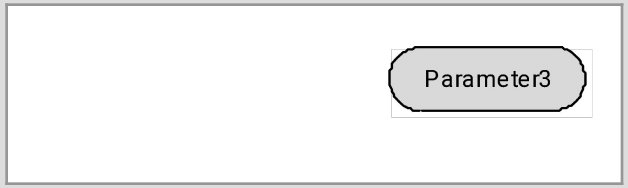
Multiple / Divide

Multiplies (divides) values of input parameters.
Allows operations for that input signals which give the same combination of basis units as output signal has.
During modeling makes diagnosis of output signal overflow (division by zero).

Name	Bit resolution
Parameter1	1
Parameter2	0.3
Parameter3	0.3

Simulink model



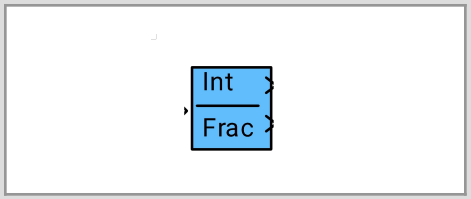


C code

```
Parameter3_DA = ( (int16_T)(((int32_T)Parameter1_DA) *
((int32_T)Parameter2_DA)));
```

```
Parameter3_DA = ( (int16_T)(((((int32_T)Parameter1_DA) * 100) /
Parameter2_DA) / 9 ));
```

Math Section

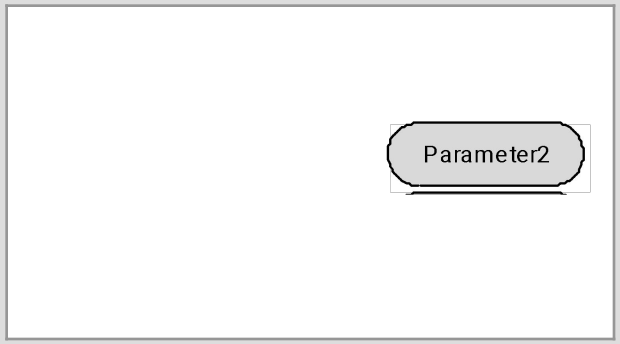


Integer / fractional

Splits a physical value of an input signal to its integer and fractional parts.

Name	Bit resolution
Parameter1	0.3
Parameter2	1
Parameter3	0.3

Simulink model

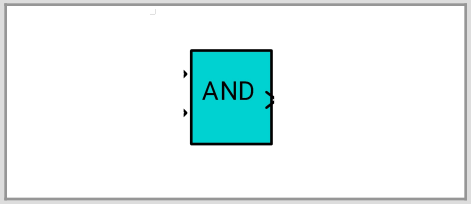


C code

```
int16_T block_B_IntegerFractional_o1;
int16_T block_B_IntegerFractional_o2;

block_B_IntegerFractional_o2 = Parameter1_DA % 3;
block_B_IntegerFractional_o1 = Parameter1_DA – block_B_IntegerFractional_o2;
Parameter3_DA = block_B_IntegerFractional_o2;
Parameter2_DA = (int16_T)(((int32_T)block_B_IntegerFractional_o1) * 3 / 10 );
```

Logical Section

An input value is TRUE (1) if it is nonzero and FALSE (0) if it is zero. The output type is specified with block settings. An output value is 1 if TRUE and 0 if FALSE.



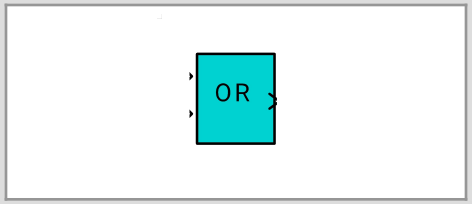
AND

Performs the AND logical operation on its inputs. TRUE if all inputs are TRUE

Simulink model

C code

```
Parameter4_DA = ( ( Parameter1_DA && Parameter2_DA &&
                    Parameter3_DA) ? 1 : 0);
```



OR

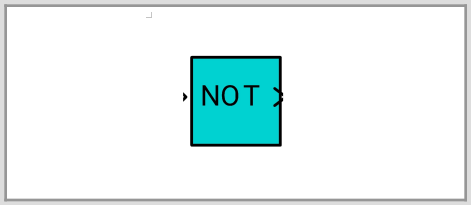
Performs the OR logical operation on its inputs. TRUE if at least one input is TRUE

Simulink model

C code

```
Parameter4_DA = ( ( Parameter1_DA || Parameter2_DA ||
Parameter3_DA) ? 1 : 0);
```

Logical Section



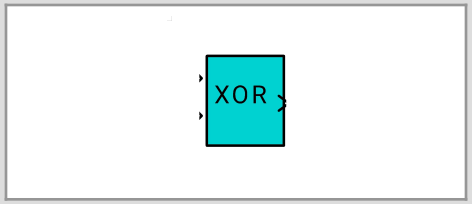
NOT

Performs the NOT logical operation on its input. TRUE if the input is FALSE.

Simulink model

C code

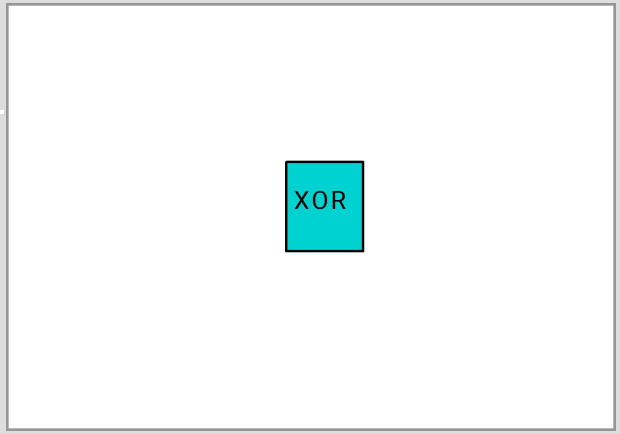
```
Parameter2_DA = ( Parameter1_DA ? 0 : 1);
```



XOR

Performs the XOR logical operation on its inputs. TRUE if only one input is TRUE.

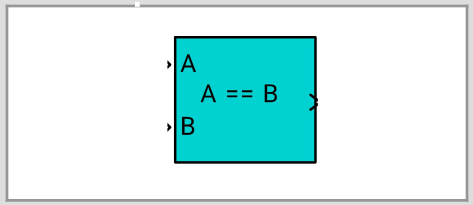
Simulink model



C code

```
uint8_T block_B_Xor;

block_B_Xor = 0;
if (Parameter1_DA != 0)
  ++block_B_Xor;
if (Parameter2_DA != 0)
  ++block_B_Xor;
block_B_Xor = block_B_Xor == 1 ? 1 : 0;
Parameter3_DA = block_B_Xor;
```

Logical Section



If

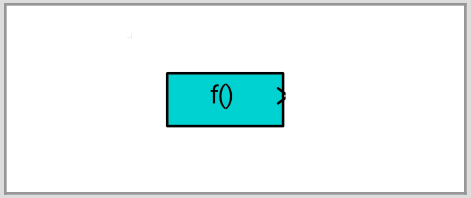
Sets the output into 1 if a specified condition for two inputs is true otherwise into 0. Available conditions are =, ≠, ≤, ≥, <, >.

Simulink model

C code

```
Parameter3_DA = ( (Parameter1_DA <= Parameter2_DA) ? 1 : 0);
```

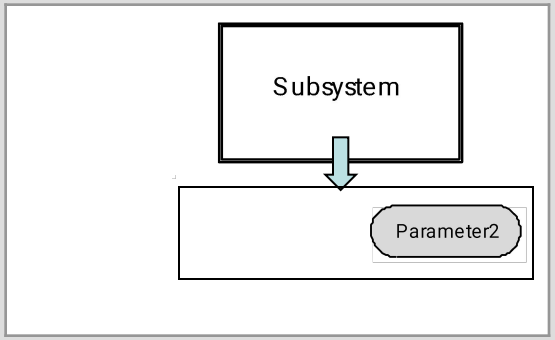
Control Section



Function Call

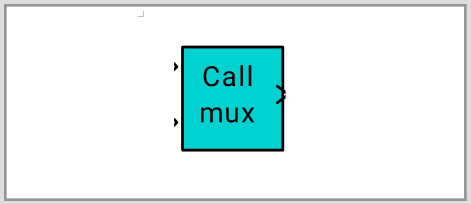
Runs the linked subsystem.

Simulink model



C code

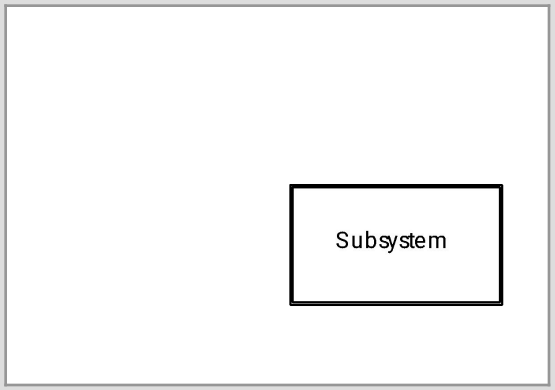
```
Parameter2_DA = Parameter1_DA;
```



Call Multiplexer

Allows to run one subsystem by means of several signals of the function-call type.

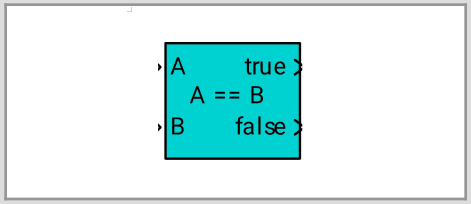
Simulink model



C code

```
void block_Subsystem(int_T controlPortIdx, int_T tid)
{
  // Operators of Subsystem;
}
block_Subsystem(0, 0);
block_Subsystem(1, 0);
block_Subsystem(2, 0);
block_Subsystem(3, 0);
```

Control Section

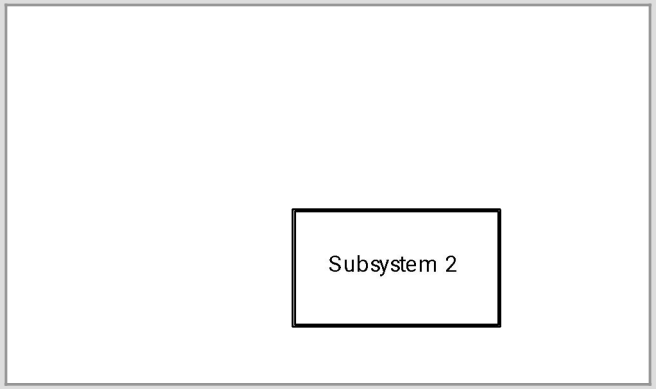


If

Checks the specified condition for two inputs and then it runs one of the linked subsystems according to the comparison result.

Name	Bit resolution
Parameter1	1
Parameter2	0.3

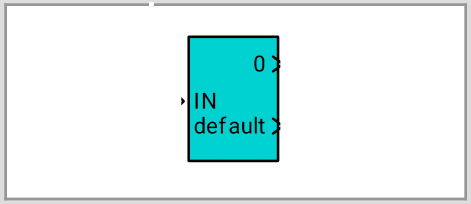
Simulink model



C code

```
if (Parameter1_DA >= (int16_T)(((int32_T)Parameter2_DA) * 3 / 10 ))
 {
   // Operators of Subsystem1;
 }
else
 {
   // Operators of Subsystem2;
 }
```

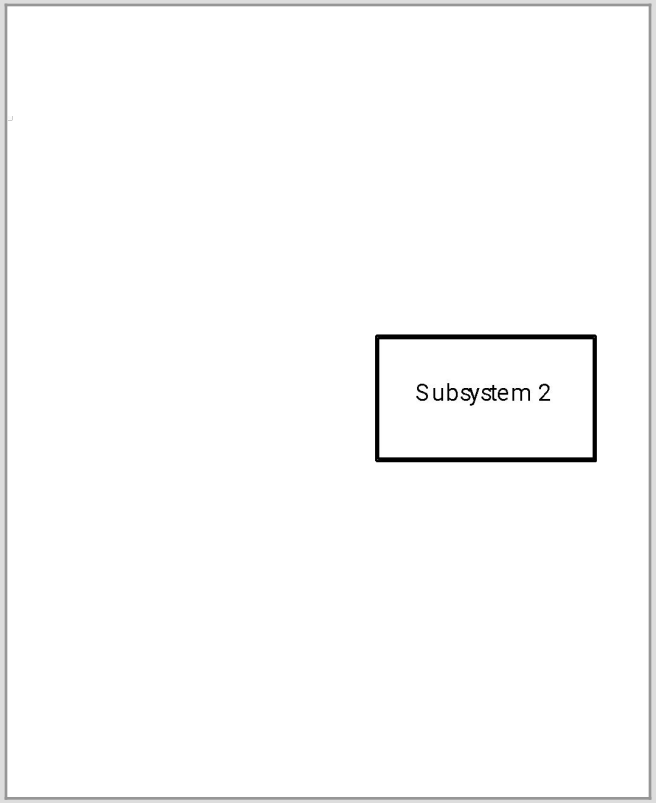
Control Section



Case

Calls one of two or greater subsystems connected to its output ports depending on an input signal value.

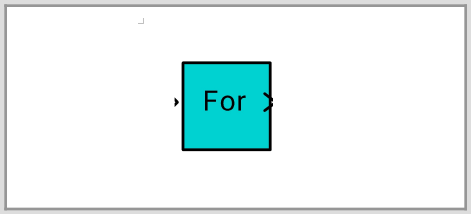
Simulink model



C code

```
switch (Variant_DA)
 {
  case 0:
   {
     // Operators of Subsystem1;
   }
   break;
  case 1:
   {
     // Operators of Subsystem2;
   }
   break;
  default:
   {
     // Operators of Subsystem3;
   }
 }
```

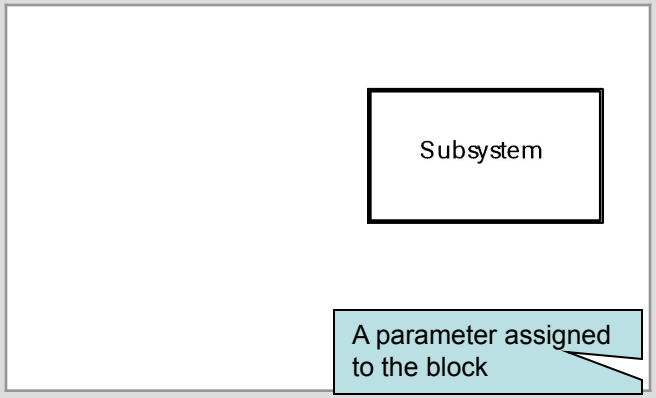
Control Section



For

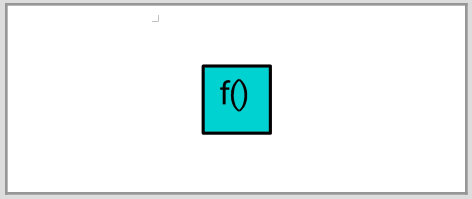
Calls cyclic execution of a subsystem connected to its output port. An iteration number is set by an input parameter value.

Simulink model



C code

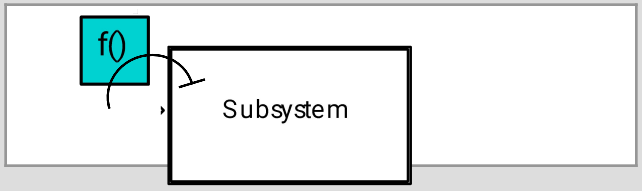
```
int16_T tmpVar_sInt16_0;

for (tmpVar_sInt16_0 = 0; tmpVar_sInt16_0 < Parameter1_DA;
   tmpVar_sInt16_0++)
{
  Index_DA = tmpVar_sInt16_0;
  // Operators of Subsystem;
}
```



Trigger

Allows to call a subsystem with a signal of a function-call type.

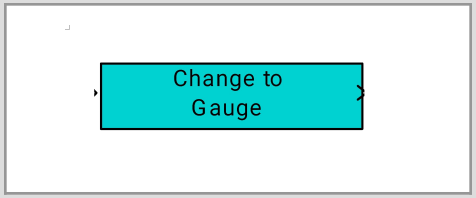
Simulink model



C code

```
// Operators of Subsystem called by a condition;
```

Signal Manipulation Section



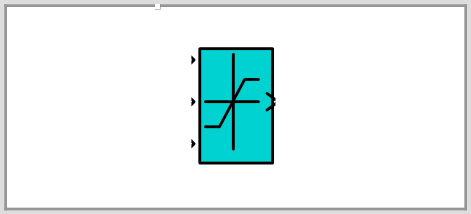
Change Bit Description

Converts an input gauge into the output one.

Simulink model

C code

```
Parameter2_DA = ( (int16_T)(((int32_T)Parameter1_DA) * 10 / 3 ));
```



Saturation

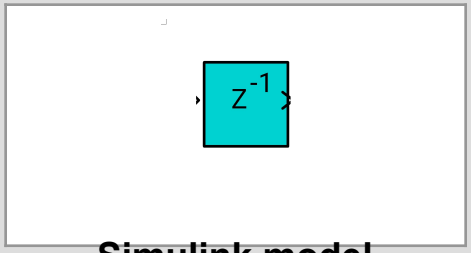
Limits its input signal value within upper and lower thresholds. The block requires the same bit gauges for all input signals.

Simulink model

C code

```
int16_T block_B_Saturation;

If (Voltage_DA < 205) block_B_Saturation = 205;
else if (Voltage_DA > 1023) block_B_Saturation = 1023;
      else block_B_Saturation = Voltage_DA;
VoltageLim_DA = block_B_Saturation;
```

Signal Manipulation Section



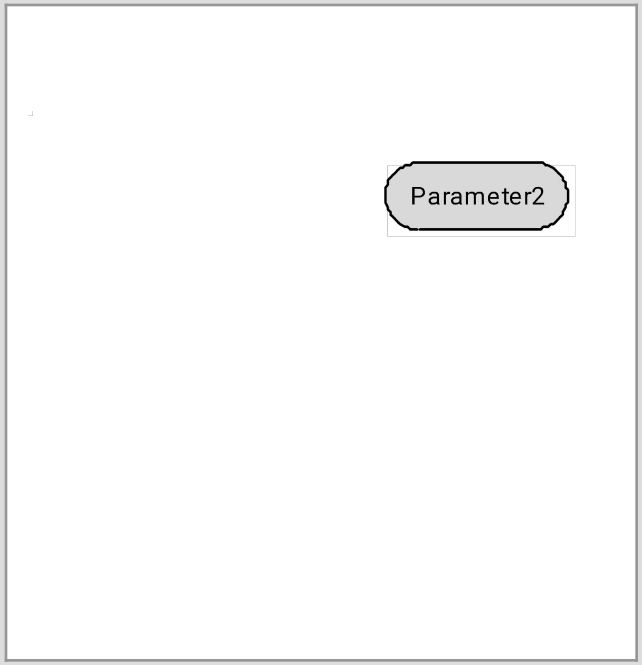
Simulink model

Delay

Assigns the last input signal value to its output signal. That is it realizes one

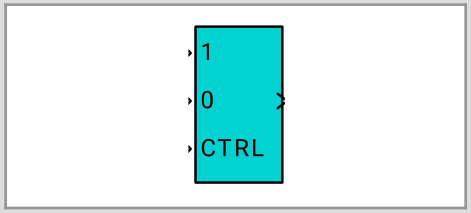
model step delay.

C code




```
int16_T block_B_Delay;
int16_T block_B_ReadParameter1;

DISABLE_OPTIMIZATION
block_B_Delay = Buffer_DA;
ENABLE_OPTIMIZATION
Parameter2_DA = block_B_Delay;
block_B_ReadParameter1 = Parameter1_DA;
DISABLE_OPTIMIZATION
if (block_B_Delay == Buffer_DA)
{
  Buffer_DA = block_B_ReadParameter1;
}
ENABLE_OPTIMIZATION
```

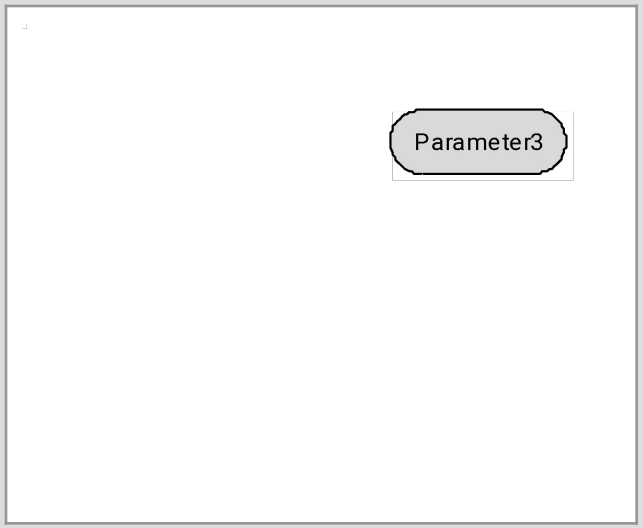
Signal Manipulation Section



Multiport Switch

Sends an input signal value to its output port. Index of the input port is defined by a value of its control signal.

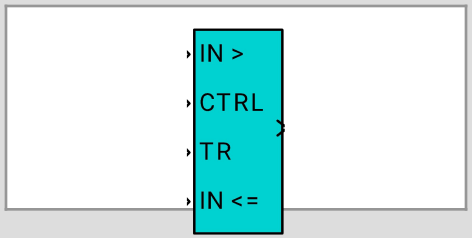
Simulink model



C code

```
int16_T block_B_MultiportSwitch;

switch (Variant_DA)
{
 case 0:
    block_B_MultiportSwitch = Parameter1_DA;
    break;
 default:
    block_B_MultiportSwitch = Parameter2_DA;
}
Parameter3_DA = block_B_MultiportSwitch;
```

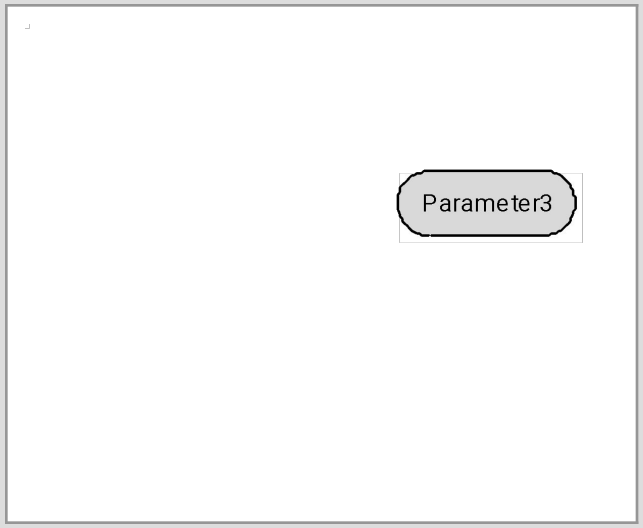
Signal Manipulation Section



Two-In Switch

Sends to its output port signals from the port “IN <=“ or “IN >” depending on a comparison result of the control signal “CTRL” with the threshold TR”.

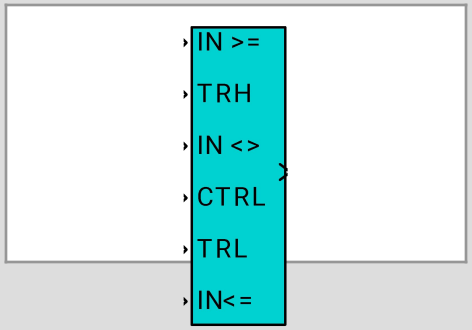
Simulink model



C code

```
int16_T block_B_TwoinSwitch;

if (Value_DA > Threshold_DA)
{
  block_B_TwoinSwitch = Parameter2_DA;
}
else
{
  block_B_TwoinSwitch = Parameter1_DA;
}
Parameter3_DA = block_B_TwoinSwitch;
```

Signal Manipulation Section



Three-In Switch

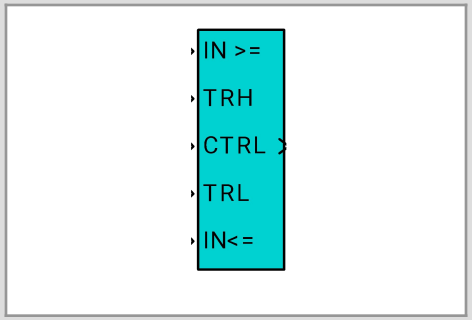
Compares a value of the “CTRL” signal with the threshold values “TRL” and “TRH”. Depending on a comparison result sends the signal “IN <=“, “IN <>” or “IN >=“ to its output port.

Simulink model

C code

```
int16_T block_B_ThreeinSwitch;

 if (Value_DA >= HighThreshold_DA)
{
   block_B_ThreeinSwitch = Parameter3_DA;
}
else if (Value_DA <= LowThreshold_DA)
{
   block_B_ThreeinSwitch = Parameter1_DA;
}
else
{
   block_B_ThreeinSwitch = Parameter2_DA;
}
Parameter4_DA = block_B_ThreeinSwitch;
```

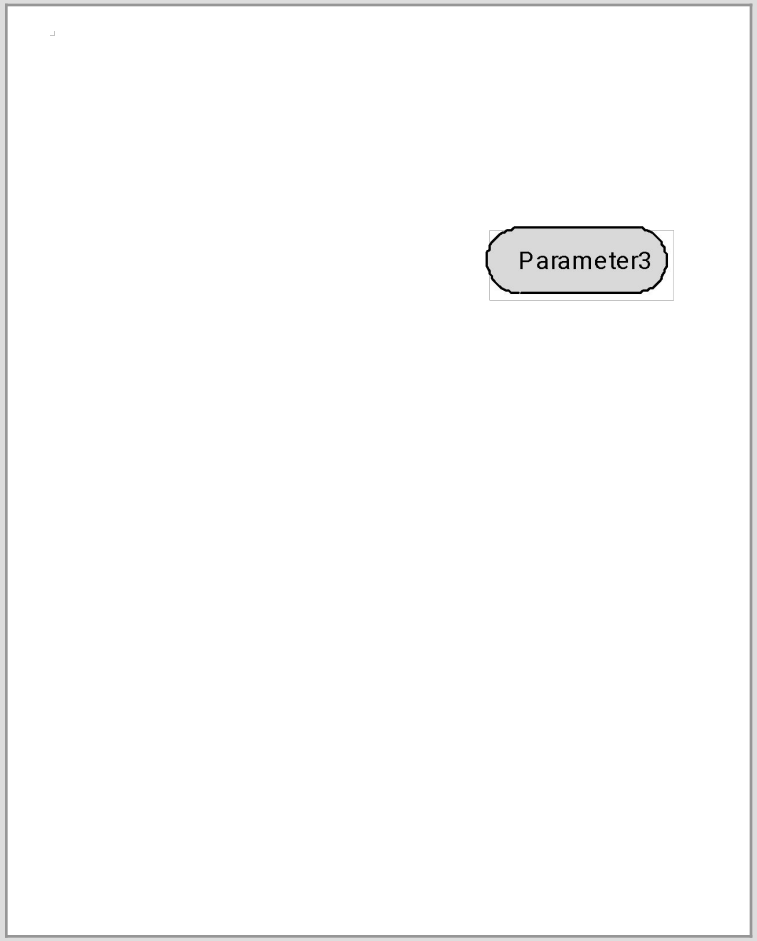
Signal Manipulation Section



Switch with hysteresis

A signal from the port “IN<=“ is sent to the output until a signal of the port “CTRL” is lower than the one of “TRH”. As soon as a “CTRL” value exceeds a “TRH” level a “IN >=“ signal is sent to the output. It’s kept until a “CTRL” value is higher than a “IN <=“ value.

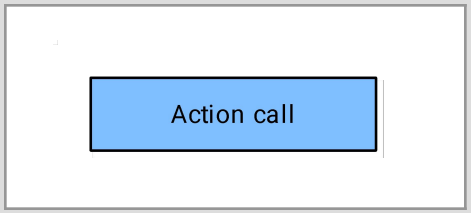
Simulink model



C code

```
uint16_T block_tmp_SwitchwithHysteresis; // static
int16_T block_B_SwitchwithHysteresis;

if (ControlValue_DA >= HighThreshold_DA)
{
  block_tmp_SwitchwithHysteresis = 1;
}
else if (ControlValue_DA <= LowThreshold_DA)
{
  block_tmp_SwitchwithHysteresis = 0;
}
if (block_tmp_SwitchwithHysteresis)
{
  block_B_SwitchwithHysteresis = Parameter2_DA;
}
else
{
  block_B_SwitchwithHysteresis = Parameter1_DA;
}
Parameter3_DA = block_B_SwitchwithHysteresis;
```

OS Service Section



Action Call

Calls an external function with input and output parameters.

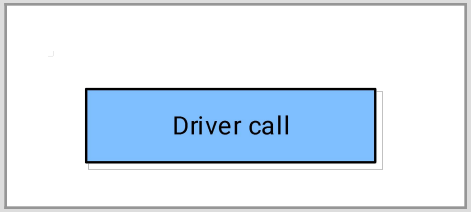
Name	Bit resolution
Parameter1	1
Parameter2	0.3

Action inputs	Bit resolution
Input1	1
Input2	1

Simulink model

C code

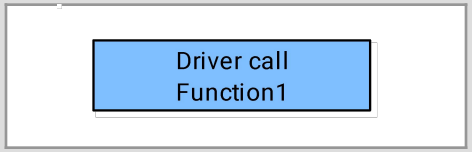
```
Action_SET_PARAM(Parameter1_DA,
        ((int16_T)(((int32_T)Parameter2_DA) * 3 / 10 )));
```



Driver Call

Calls an external function without parameters.

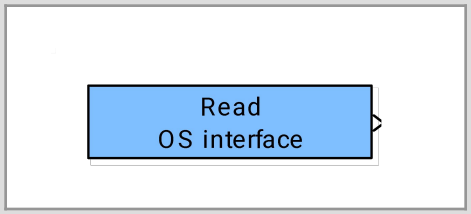
Simulink model



C code

```
Function1();
```

OS Service Section



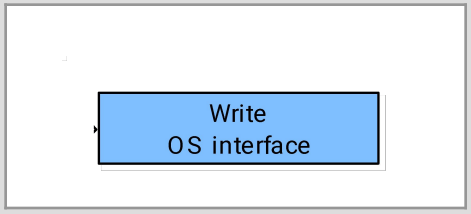
Read System Variable

Reads a system variable to transmit its value from the system software level to the application software level.

Simulink model

C code

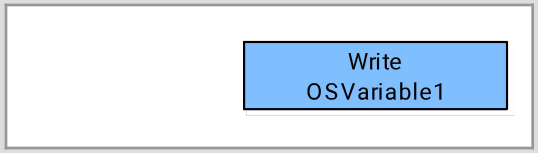
```
Parameter1_DA = OSVariable1;
```



Write System Variable

Writes a parameter into a system variable to transmit its value from the application software level to the system software level.

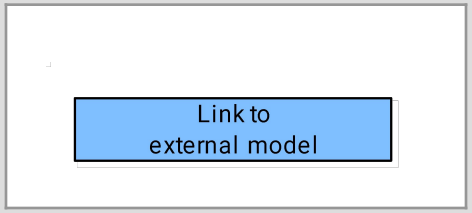
Simulink model



C code

```
OSVariable1 = Parameter1_DA;
```

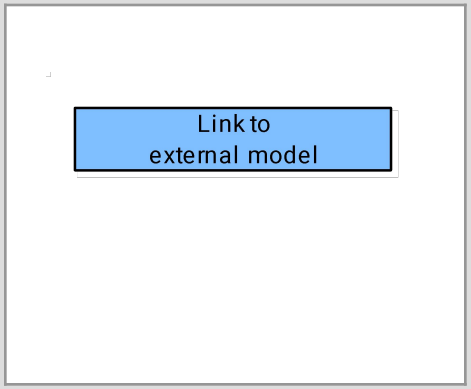
OS Service Section



Link to External Model

Calls an external model located in an external file.

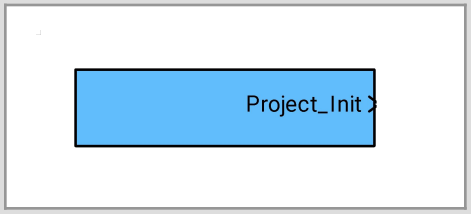
Simulink model



C code

```
Model1_step();

void Model_initialize(boolean_T firstTime)
{
  if (firstTime) {
    Model1_initialize(1);
  }
}
```

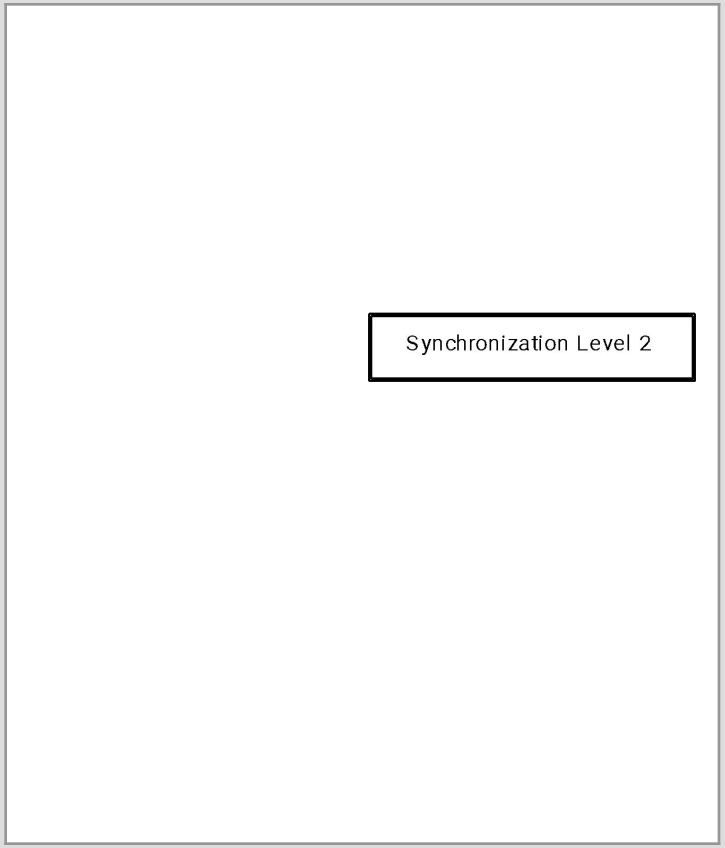
OS Service Section



OS Scheduler

Provides running synchronization levels containing control functions.

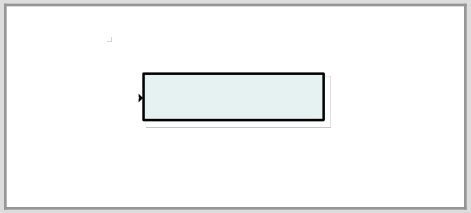
Simulink model



C code

```
void _Level1()
{
   // Operators of subsystem 'Synchronization Level 1'
}
void _Level2()
{
   // Operators of subsystem 'Synchronization Level 2'
}
void Project_Init()
{
   // Operators of subsystem 'Initialization'
}

const _FUNC_ XHUGE TaskTable[LEVELS_COUNT] = {    // Functions
  _Level1,_Level2 };

const unsigned char PriorsTable[LEVELS_COUNT] = {      // Priorities
  0x00, 0x00, 0x00 };
```

Bitwise Operators Section



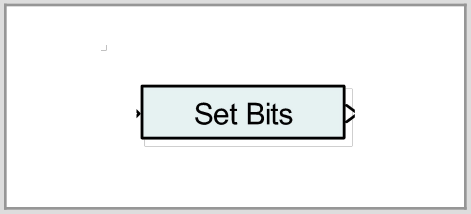
Get Bits

Returns bit values extracted from an input parameter.

Simulink model

C code

```
Parameter3_DA = ( (Parameter1_DA & 0x4) == 0x4 ? 1 : 0);
Parameter2_DA = ( (Parameter1_DA & 0x1) == 0x1 ? 1 : 0);
Parameter4_DA = ( (Parameter1_DA & 0x8000) == 0x8000 ? 1 : 0);
```



Set Bits

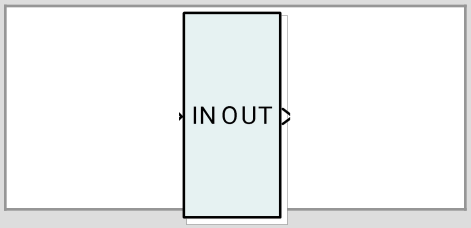
Sets bits of an output parameter according to a specified bit mask.

Simulink model

C code

```
Parameter2_DA = ( Parameter1_DA | 0x8085);
```

Bitwise Operators Section



Pack Bits

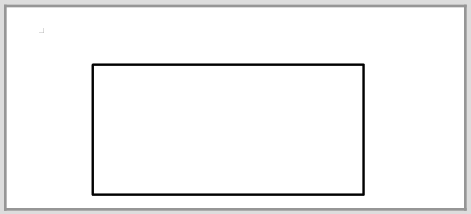
Sets bits of an output parameter according to input logical parameters.

Simulink model

C code

```
uint16_T block_B_PackBits;

block_B_PackBits = LogicParameter1_DA != 0 ? Parameter1_DA | 0x1 :
  Parameter1_DA & 0xFFFE;
block_B_PackBits = LogicParameter2_DA != 0 ? block_B_PackBits | 0x4 :
  block_B_PackBits & 0xFFFB;
block_B_PackBits = LogicParameter3_DA != 0 ? block_B_PackBits | 0x8000 :
  block_B_PackBits & 0x7FFF;
Parameter2_DA = block_B_PackBits;
```

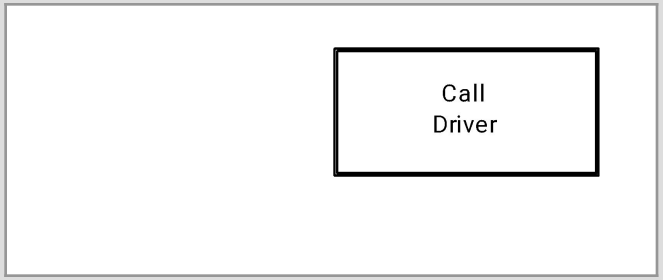
Signals & Systems Section



Subsystem

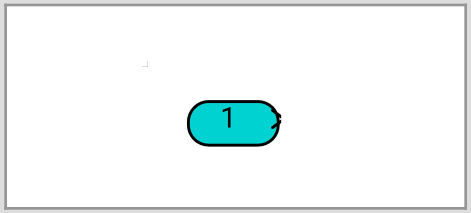
It is a container for the library blocks.

Simulink model



C code

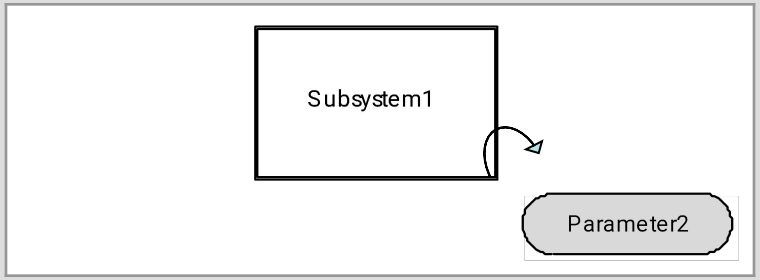
```
// Operators of subsystem 'Calculation of Operators'

// Operators of subsystem 'Call Driver'
```



Input Port

Transmits a signal to a subsystem.

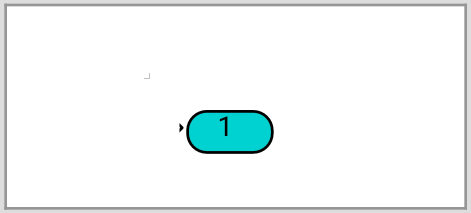
Simulink model



C code

```
Parameter2_DA = ( Parameter1_DA);
```

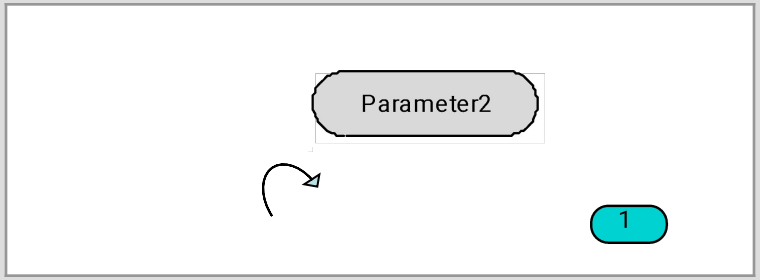
Signals & Systems Section



Output Port

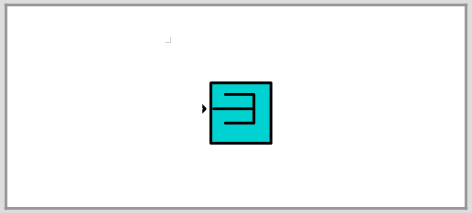
Transmit a signal out of a subsystem.

Simulink model



C code

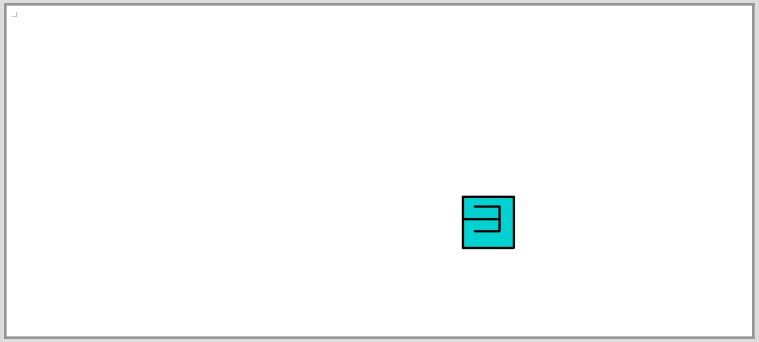
```
Parameter2_DA = ( Parameter1_DA);
```



Terminator

Terminates unused branches.

Simulink model



C code

```
if (Parameter1_DA >= Parameter2_DA)
  {
    // Operators of subsystem 'Subsystem1'
  }
else
  {
  }
```