VISUALSIM TRAINING

Agenda- Part 3: VisualSim

Basic Concepts

Key Library Components

Traffic

Plotting, Display and Statistics

Behavior Modeling

Database

Introduction to Virtual Connections

Concept of Delay

Resources

RegEx

Script Language

Statistics

Parameters, Variables & Data Structures

Definitions of Parameter and Variable

	Parameter	Variable
Define inputs	of any Data type	of any Data type
Input value	• fixed throughout the simulation	• Vary during simulation
Availability	• Available to the current window and all hierarchy blocks below this level	• Local - used in current window • Global - available in full model

Parameter- Review and Application

- Review
  - ✓Constants
  - ✓Cannot be changed with in a simulation run
  - ✓Can be changed across simulation runs
  - ✓Hierarchical
- Application
  - ✓Any attribute of the model that needs to be modified for the design space
  - ✓Usage: Network Speed, Cache Size, Simulation Time, Seed
  - ✓Example: Any model

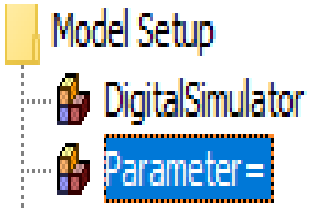
To Create New Parameter

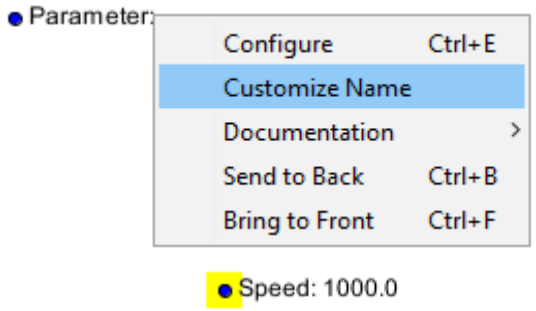
Model parameter:

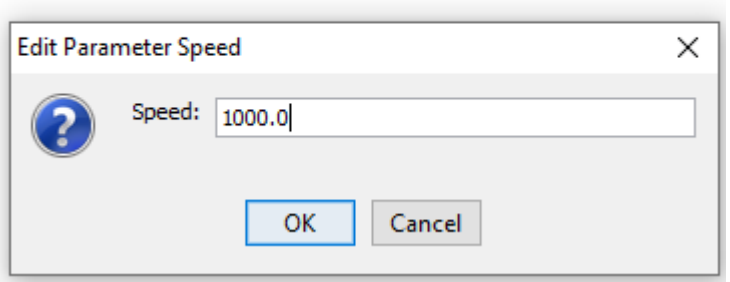
1. Drag-n-Drop the parameter from Library Folder **Model Setup >Parameter** ('parameter=') into an open Block Diagram Editor window.

2. Right-click to select **Customize Name** of parameter & enter a name. Name must be unique, else BDE will generate exception.

3. Double click the new parameter name to set the value of the parameter.



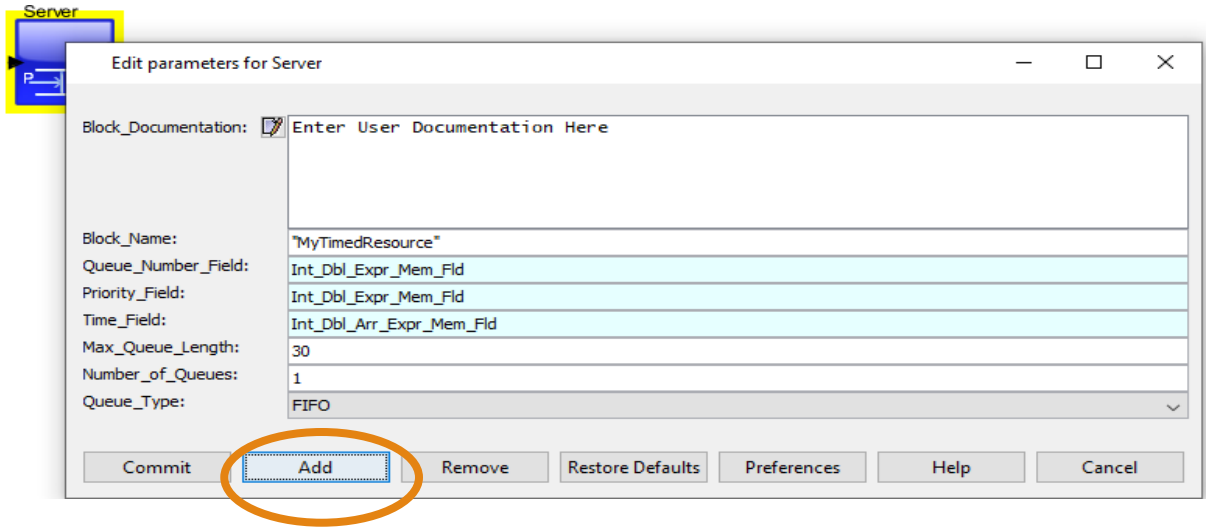


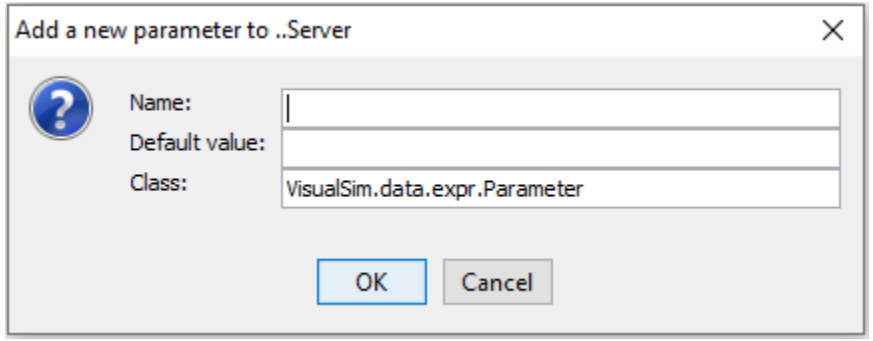


To Create New Parameter

Block Parameter :

Double click on the block and select **Add**. Enter the parameter name and value.





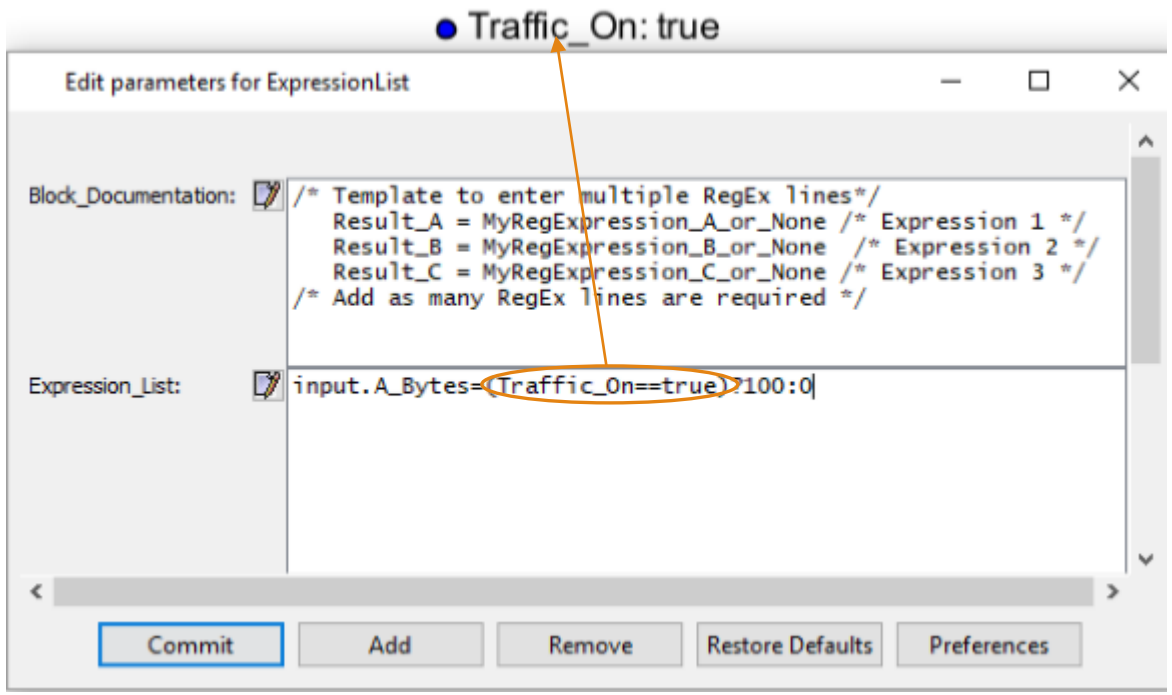
Parameter Types

Types	Values
String	“Queue1” “file://C:/VisualSim/filename.txt”
Integer Double Long	1 1.0 123L
Boolean	True
Array Matrix	{1,2,3} [1,2;3,4] Note: Can contain any data type
Expression	(Parameter1==4)?Parameter2:Parameter4
Data Structure	{first=1,second=”name”}

Parameter Usage

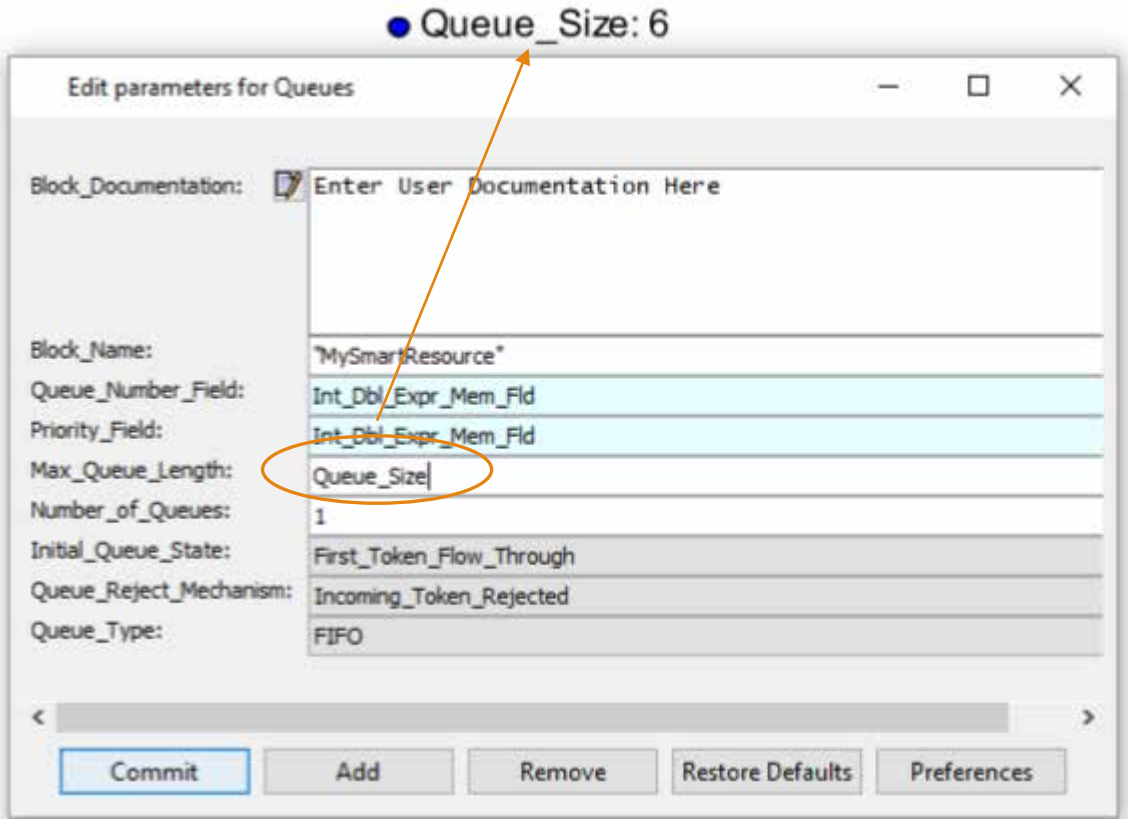
Parameter in Expression:

- Able to define mathematical expressions using parameter values. Parameters can only be on the right hand side of the expression.



Parameter Usage

Parameters to set Block attributes



Parameter List

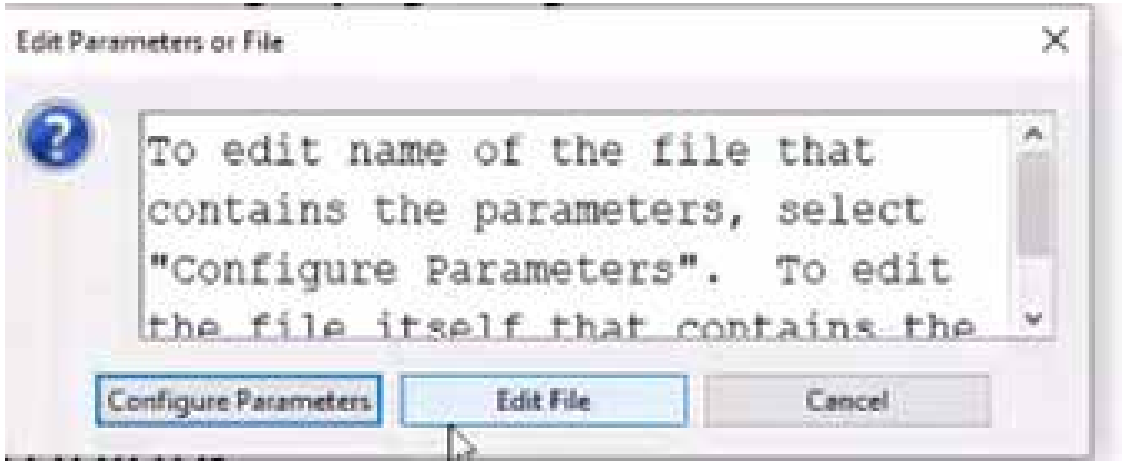
A block that reads the parameter names and their values from a file and sets corresponding parameters for the model.

Edit File

- Parameter list contains parameter names and respective values.
- It is in textual format such that user can make changes in the file and values gets updated accordingly.

Configure Parameter

- Able to select different parameter configuration.
- Specify parameter from text editor itself instead of the tool.



Variable

- Variable is a variable or register
- Variable is a named location
  - ✓Local- Current window
  - ✓Global- Entire model
  - ✓Block- Use in Script and ExpressionList
- Used to communicate between blocks and routing
- Defined in VariableList (Global and local), ExpressionList blocks(Block) and Script (Block)
- Used in ExpressionList and Script
- Supports all standard data types
- Initialize using VariableList
  - ✓**Local**- ”myvariable local 0.0”
  - ✓**Global**- “myvariable global “string””
- Initialize using Script and ExpressionList blocks
  - ✓Done the first time variable is accessed
- Initialize with RegEx and Script
  - ✓Use only for block memory
- Types supported
  - Int, long, double, binary, string, data structure, array and boolean

Variables

Located under Model Setup and
Full_Library->Model->Variables
Full Library->Model->Utility->Checkers

Accessing Variables

Read_Variable

RD_WR_Variable

Write_Variable

RD_WR_N_Variable

Rd_Wr_Array

Define Variables

VariableList

Test

Variable_Dump

Variable_Monitor

Variables

Edit parameters for VariableList

Block_Documentation:

```
Array Variable: Array can contain any types. All items of an array must be the
same.  Variable can be initialized with all items different or can be set with
a default.  Default can be a DS Template or a value.
Data Structure Variable: Variable can be defined as a Data Structure.
There are two ways- defining the DS in the line or referencing a template.
Memory Type: Memory can be defined as global or local type.  Local is
available in the current window. Global is available to the entire model.
Global variables can be defined anywhere in the model.
```

Memory_Init_Text:

```
/* Memory Initialize Template
Name          Type          Value     */
Varint          global         1      ; /* Integer Variable */
Vardbl          global         1.0      ; /* Double Variable */
Vararray        global         {1,2,3,4} ; /* Array Variable. Array can contain any types. All items of an a
Array_of_Array  global         {50:{50:0}} ;  /* Array of Arrays */
Variable1       global         {10:Processor_DS} ; /* DS Array with Processor_DS template with 10 index.*/
VarDS           global         Processor_DS ; /* Defines a Variable with Data Structure with VS/data/Process
VarDS1          global         {first=1, second=2.0, third="string"} ; /* Fields defined here */
Varstr          global         Destination ; /* String Variable */
Varbool         global         TRUE ; /* Boolean Memory of TRUE or FALSE */
Varlong         global         1L ; /* Long Variable */
```

Commit | Add | Remove | Restore Defaults | Preferences | Help | Cancel

Variable Blocks and RegEx

- Variable_Monitor
  - ✓Trigger the block to output the current value of a pre-determined list of blocks. Output is an array.
  - ✓Generate a output when a variable out of the list is being written into or read from. Output is an array.
- Variable_Dump
  - ✓Outputs the current value of all the variables in the model- global, local and block. Output is an array.
- RegEx
  - ✓readAllVariables provides current value of all variables
  - ✓Check(“Name”)- Does the variable exist?

Accessing Variables

- Standard approach
  - ✓ Use the Variable name on the LHS and RHS of an expression
  - ✓ Global Variable can be accessed in any block in the entire model
  - ✓ Local Variable can be accessed from same window as the definition
  - ✓ Block Variable are accessed in a Script or ExpressionList
- Reading local Variable from outside the current Window
  - ✓ StringName = List hierarchies separated by '.' + "Variable_Name"
  - ✓ MyVariableToken = StringName.read()
- To overwrite the content of a local Variable
  - ✓ StringName ="My_Variable_Name"
  - ✓ MyVariableToken = StringName.write(New Value)
- Accessing block Variable from outside the script block
  - ✓ StringName = "My_Block_Variable_Name"
  - ✓ MyVariableToken = StringName.read(Block_Name)

VariableList- Applications

- Systems
  - ✓ Storage systems (maintain count of available words in the cache for each incoming request)
  - ✓ Example: Application Demo->Systems->Computer Model. Look at Memory_Init for initialize and L1 Cache->Decision4
- Network
  - ✓ In_Thru and Out_Thru (Continuous count of bytes at input and output ports. Used for statistics and for traffic shaping)
  - ✓ Example: Application Demo->Flow Control
- Hardware
  - ✓ L1, L2, L3 (Maintains the current content in an address at the Caches)
  - ✓ Example: Application Demo->Imaging->Video Cache Paging. Look at Memory_Init for initialize and all Processing blocks
- Software
  - ✓ Clusters (Flag for active and inactive Hardware within a cluster), Jobs (Current job executing at each core or processor)
  - ✓ Example: Application Demo->Other->SW Pool Size

What is a Data Structure?

- Data Structure is similar to “struct” in C
- Fields of the Data Structure represent
  - ✓Data transmitted through the model
  - ✓Represents the IC pins, frames, packets etc.
- Fields can hold the results of a mathematical or logical operation
- DS Fields can be

  Strings, Boolean, integers, doubles, arrays, matrix, data structures, long

Data Structure

Signals or transactions

- Propagate from block to block along the wires between ports
- Dynamically create and remove fields

Class type

- Containing named fields and associated value
- May be accessed using the period operator
  - look like “input.FieldName”

Field Name	Field Value
{BLOCK	"Traffic"
DELTA	0.0
DS_NAME	"DS_Traffic"
Field1	1.0
Field2	"str"
Field3	True
ID	1
INDEX	0
TIME	1.0E-10}

Supported Data Types

Data Type	Example
Integer	20
Long	2L
Double	1.0
String	"L2_Cache"
Boolean	true or false
Data Structure	{FldA=1, Fld2=3}
Binary String (Available only in the Data Structure blocks and is being deprecated.)	4'b100 (string for the Verilog format)
Array	{1.0, 2.0} or {{1.0,2.0},{3.0,4.0}}
Matrix	[1, 2; 3, 4]
Complex	4 + 2j
Fixed Point	fix(.37665, 6, 2)
Embedded Data Structure	Data_Structure "Processor_DS" or Data_Struct "C:.VisualSim.VS_AR.VisualSim.data.Processor_DS"

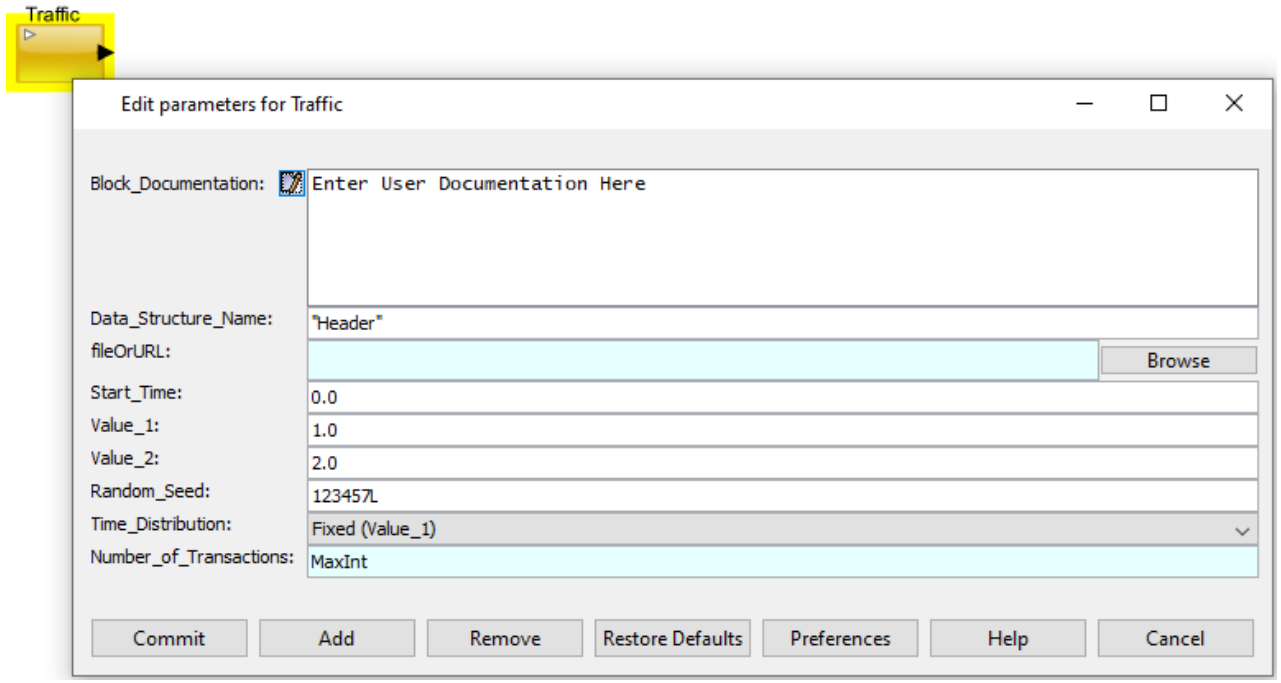
Base Data Structure: Header

All data structures have these six fields.

- BLOCK (Source name)
- DS_NAME (Template name)
- TIME (creation time-stamp)
- ID (sequence number)
- INDEX (integer scratch pad)
- DELTA (double scratch pad)

Common DataStructures

- Header- Construct custom Data Structure
- Processor_DS- Used by all hardware blocks
- Ether_DS- Used by TSN
- Task_Class- Used by Node blocks



Example of DataStructure Template

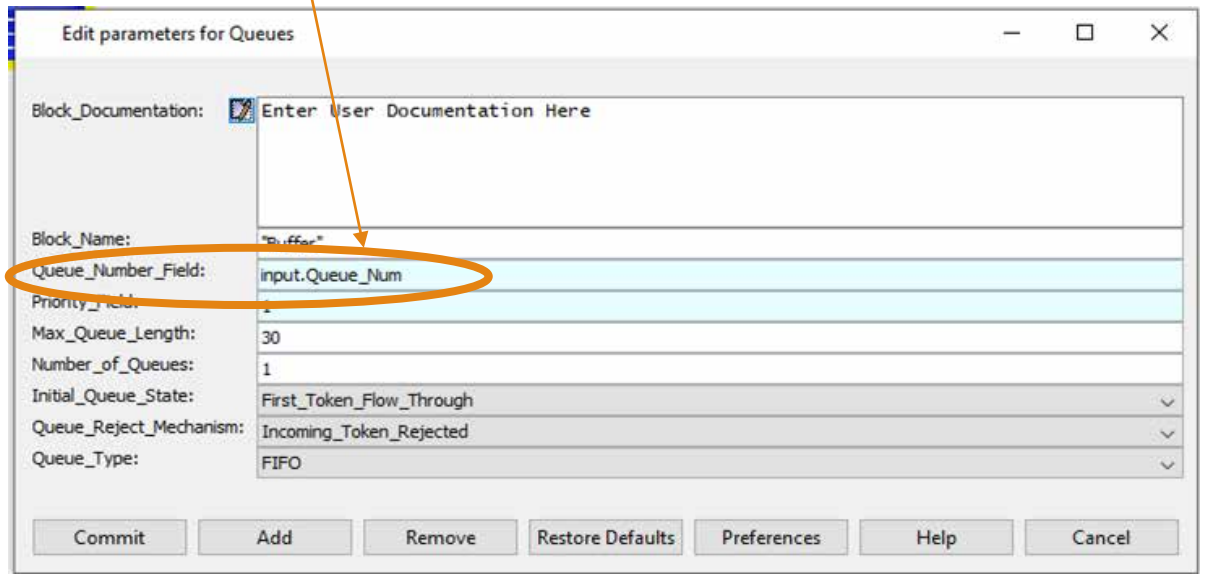
Standard Data Structures are located in VS_AR/VisualSim/data folder
User can create a new one and place it along with their own model
Access the Data Structure template using the Traffic block or newToken RegEx function

```
Cache Data Structure

      Note: Defines information needed in a Cache Request and Cache Line

Field Name              Type     Value         Comment                          */

Val_Bit                boolean  true     ; /* Cache Line: Valid Flag            */

Lock_Bit               boolean  false    ; /* Cache Line: Lock  Flag            */

Age_Bits               int      0        ; /* Cache Line: Age of Line           */

Tag_Bits               int      0        ; /* Cache Line: High  Address, Int    */

Addr_Bits              int      0        ; /* Cache Line: Mid   Address, Int    */

Word_Bits              int      0        ; /* Cache Line: Low   Address, Int    */

Access_Command         String   Read     ; /* Access: Command                   */

Access_Sequential      boolean  true     ; /* Access: Seq or Non-Sequential     */

Access_Bytes           int      4        ; /* Access: Bytes                     */

Access_Time            double   1.0      ; /* Access: Time to next Access       */

Access_Next            int      1        ; /* Access: Index to next Access      */

Data                   String   01AB     ; /* Data  : HEX                       */

Data_Structure         String   Pass     ; /* Data Structure Passed Through     */
```

Field Usage

- To hold information
- Use in blocks

```
DISPLAY AT TIME                    ------ 0.0 ps ---
{BLOCK                           = "Traffic2",
DELTA                            = 0.0,
DS_NAME                          = "Header_Only",
Data_Size                        = 717,
ID                               = 1,
INDEX                            = 0,
Queue_Num                        = 1,
TIME                             = 0.0}
```



Agenda- Part 3: System Libraries

Traffic 165-177

File I/O 178-183

Plotting, Display and Statistics 184-190

Expression List 191-194

Database 195-202

Concept of Virtual Connections- 203-239

RegEx 240-255

Script Language 256-293

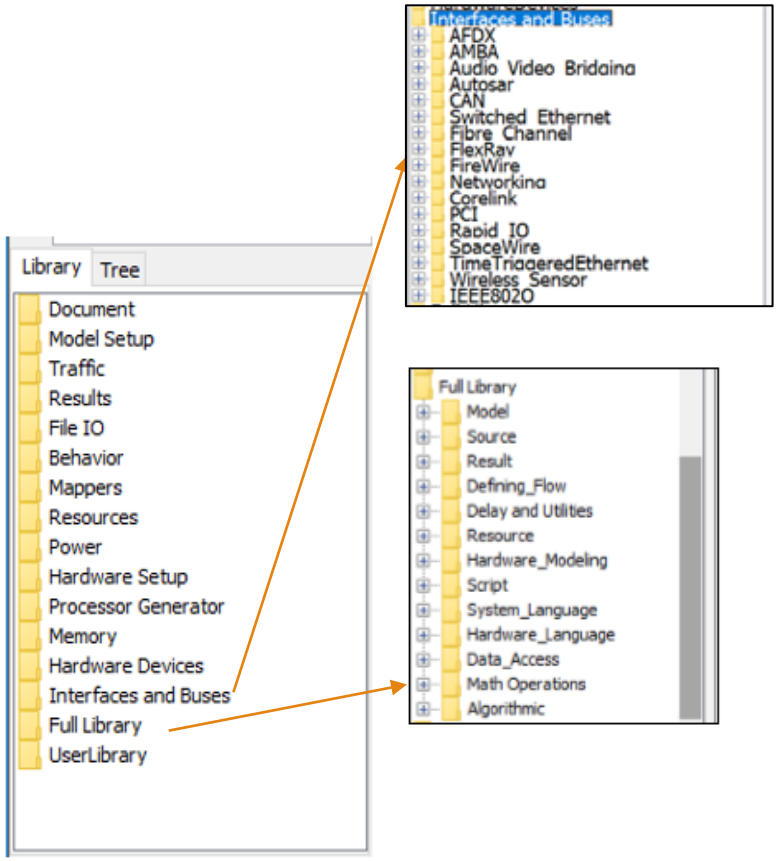
Resources 294-337

Debugging 339-365

Configuring Blocks 366-389

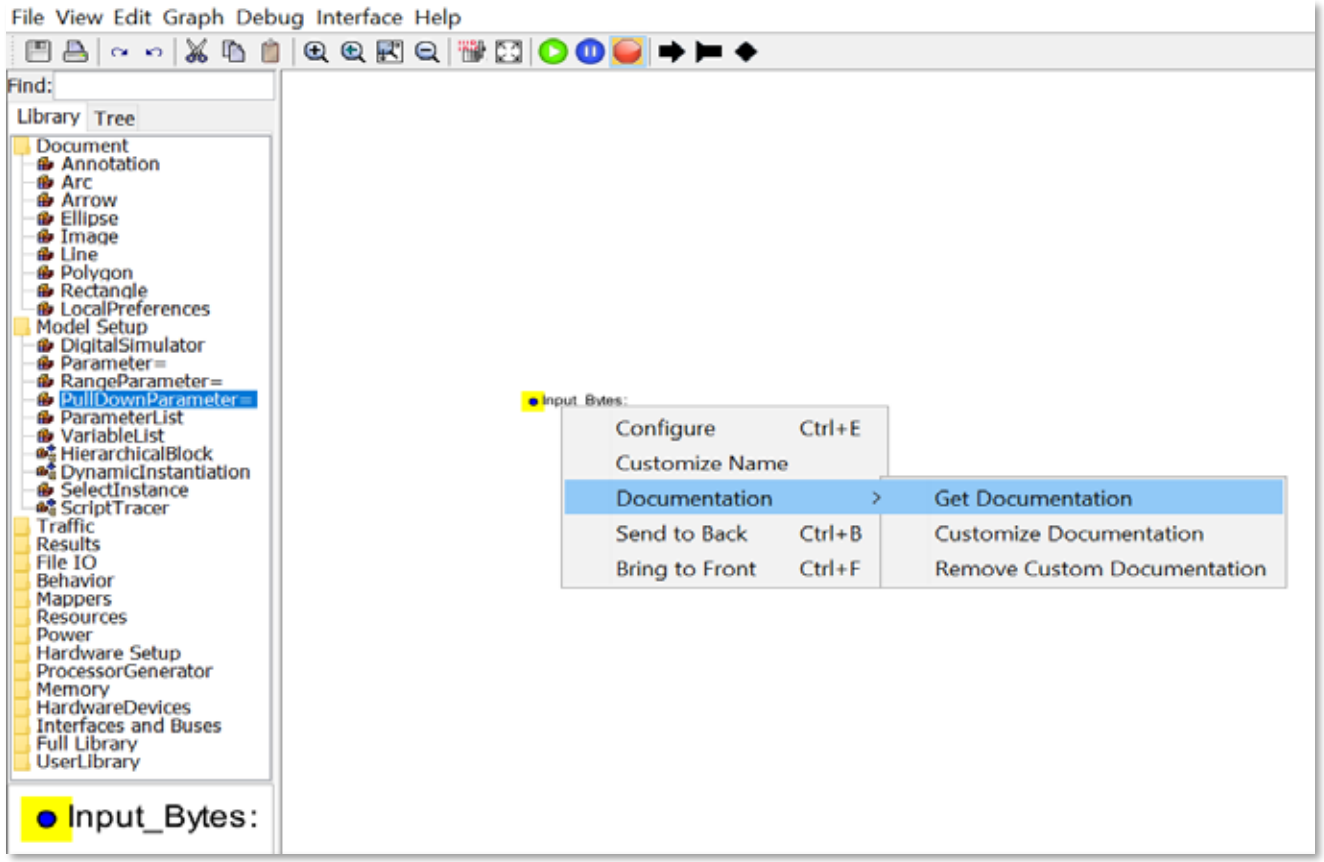
Key Library Components

Library Organization



Library	Components to cover in basic training
Model Setup	Digital Simulator, Parameter, Variable, Hierarchal Block
File IO	Database
Traffic	Traffic, Delay
Behavior	Expression List, Join, Fork, In and Out
Resources	Queues, Server, System Resource
Power	Power manager
Mappers	Mapper
Results	Text Display, Time Data Plotter

Access Documentation of Library Specifications



Online Video Introductions (1)

No	Library Block	Description	Block
1.	Digital Simulator	Implements the Discrete – event model of computation https://www.youtube.com/watch?v=IReOQrpLM64	
2.	Parameter	In order to give global parameters for the model https://www.youtube.com/watch?v=I3TwzKawIo8	Input_Bytes: 64
3.	Variables	The block is used to define memory locations https://www.youtube.com/watch?v=OLMiBkzgRoI	
4.	Traffic	Generate Input as per the specified data structure and time distribution https://www.youtube.com/watch?v=OLMiBkzgRoI	
5.	Expression List	Executes sequence of assignments statements and send the output by evaluating the expression https://www.youtube.com/watch?v=OLMiBkzgRoI	
6.	Delay	Generate delay while transmitting the packets https://www.youtube.com/watch?v=UUcW-3w7fQM	
7.	Hierarchical Block	Group a set of functional blocks that has function or device https://www.youtube.com/watch?v=I3TwzKawIo8	

Online Video Introductions (2)

No	Library Block	Description	Block
8.	Queues	Orders the incoming data from High priority to lowest https://www.youtube.com/watch?v=OLMiBkzgRoI	
9.	Server	Multidimensional Resource – Multiple Queues + time delay https://www.youtube.com/watch?v=OLMiBkzgRoI	
10.	Mapper	To model multi-threaded application, dynamically allocates to one of many resources https://www.youtube.com/watch?v=UWpUZGjArdo	
11.	System Resource	To link multiple concurrent behavior flow into single block https://www.youtube.com/watch?v=DmGyNIj58WU	
12.	Join and Fork	Join: Separate inputs from any output; Fork: Control execution of two concurrent flow https://www.youtube.com/watch?v=OLMiBkzgRoI	
13.	Power Manger	To evaluate battery discharge, instantaneous power, average power so on.. https://www.youtube.com/watch?v=vceZ-LLHyRc	

Online Video Introductions (3)

No	Library Block	Description	Block
14.	Text Display	Complete Statistics of data processed https://www.youtube.com/watch?v=JyxXdOc24IQ	
15.	Time Data Plotter	Plots the processed data on Y-axis against current simulation time in X-axis https://www.youtube.com/watch?v=JyxXdOc24IQ	
16	Database	Used as a lookup table for doing searches Demo Model https://www.youtube.com/watch?v=UWpUZGjArdo → M	
17.	IN	It accepts request from Virtual connections (MUX and OUT) and virtual machine and its wireless connection to reduce the number of links in BDE	
18.	OUT	Route the data to other parts of the model in BDE or any part of the global model and its wireless connection too https://www.youtube.com/watch?v=UWpUZGjArdo (Both IN and OUT)	

Complete Systems-Level Library

MIRABILIS design

Traffic • Distribution • Sequence • Trace file • Instruction profile **Reports** • Timing and Buffer • Throughput/Util • Ave/peak power • Statistics	**Power** • State power table • Power management • Energy harvesters • Battery • RegEx operators	**SoC Buses** • AMBA and Corelink • AHB, AB, AXI, ACE, CHI, CMN600 • Network-on-Chip • TileLink	**System Bus** • PCI/PCI-X/PCIe • Rapid IO • AFDX • OpenVPX • VME • SPI 3.0 • 1553B	**Processors** • GPU, DSP, mP and mC • RISC-V • Nvidia- Drive-PX • PowerPC • X86- Intel and AMD • DSP- TI and ADI • MIPS, Tensilica, SH	**ARM** • M-, R-, 7TDMI • A8, A53, A55, A72, A76, A77
Custom Creator • Script language • 600 RegEx fn • Task graph • Tracer • C/C++/Java • Python **Support** • Listener and Trace • Debuggers • Assertions	**Stochastic** • FIFO/LIFO Queue • Time Queue • Quantity Queue • System Resource • Schedulers • Cyber Security **RTOS** • Template • ARINC 653 • AUTOSAR	**Memory** • Memory Controller • DDR DRAM 2,3,4, 5 • LPDDR 2, 3, 4 • HBM, HMC • SDR, QDR, RDRAM **Storage** • Flash & NVMe • Storage Array • Disk and SATA • Fibre Channel • FireWire	**Networking** • Ethernet & GiE • Audio-Video Bridging • 802.11 and Bluetooth • 5G • Spacewire • CAN-FD • TTEthernet • FlexRay • TSN & IEEE802.1Q	**FPGA** • Xilinx- Zynq, Virtex, Kintex • Intel-Stratix, Arria • Microsemi- Smartfusion • Programmable logic template • Interface traffic generator **Software** • GEM5 • Software code integration • Instruction trace • Statistical software model • Task graph	**Interfaces** • Virtual Channel • DMA • Crossbar • Serial Switch • Bridge **RTL-like** • Clock, Wire-Delay • Registers, Latches • Flip-flop • ALU and FSM • Mux, DeMux • Lookup table

Minimizes the need for custom development and quick custom development language

Traffic, Reports and Interfaces

- Traffic
  - ✓Sequence, distribution-based, intermittent, files and clocks
- Plotters and Debugging Tools
  - ✓Real-time viewers, animation and breakpoint
  - ✓Text, export, statistics
- Pre-configured Analysis
  - ✓Power- Instantaneous, average and discharge
  - ✓Performance- Latency, buffer, hit-ratio, stall-times, utilization, throughput, I/Os second
  - ✓Battery- lifecycle, charge and discharge, capacity usage
- Interfaces
  - ✓C/C++/Java, Python, MatLab, Excel, XML
  - ✓File I/O, FPGA board, Database
  - ✓SystemC, HDL, STK

No Post Processing Required- Development to Analysis together

Resources, Hardware and Algorithms

- Performance Resource
  - ✓Active and Quantity Resources
  - ✓Channels, pipeline, SystemResource (schedulers), queues
- Cycle-Accurate Architecture Generators
  - ✓Processor (uP, DSP, Custom, GPU, TPU, and AI), memory, cache
  - ✓Profile-based software sequence generator, trace from fast functional model in GEM5 and ARM Fast Functional Library
  - ✓Linear, switched and Req-Ack bus
  - ✓Pipeline, DMA, Controllers
  - ✓Bridges, Switches (Blocking & non-blocking)
- Behavior
  - ✓Block-based, C-like scripting, Java/C/C++, SystemC, Python
- Application-Specific
  - ✓Signal and image processing, analog, controls
  - ✓Wired Networking and Wireless Sensor Networks

No Programming Required- Accelerate model development

Selecting the right block- 1

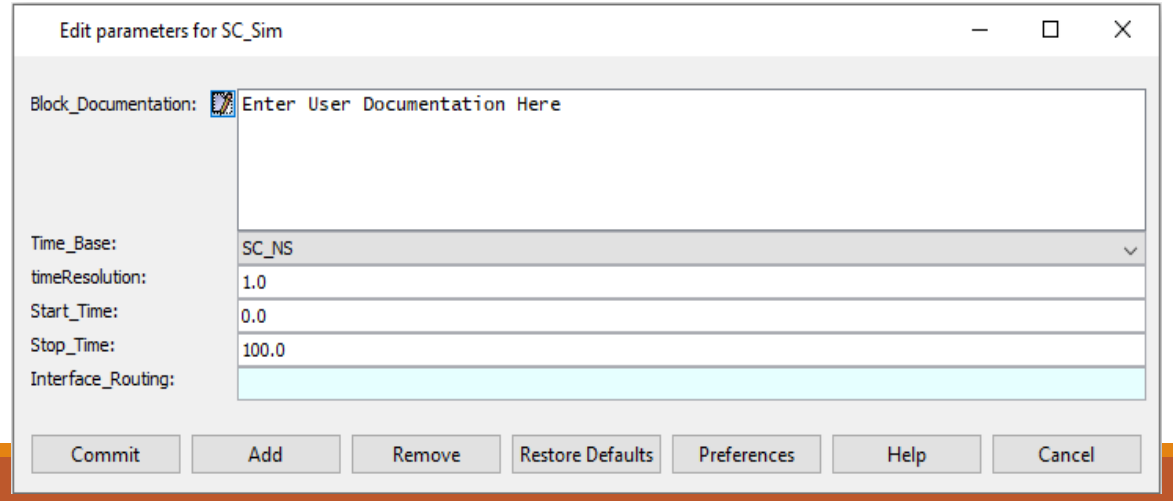
- Traffic, Test Bench, Clock
  - ✓Traffic>Clock
  - ✓Traffic>Wireshark Network or VCD Hardware trace
- Analysis, Reports, Display
  - ✓Results->TimeDataPlotter
  - ✓Results->Statistics
  - ✓Result->TextDisplay
- Math and Logical
  - ✓Use RegEx language
- Write/Read File
  - ✓Traffic Reader
  - ✓File Reader
  - ✓File Writer
  - ✓Excel
  - ✓XML
- Import
  - ✓C Code
  - ✓Application
  - ✓SystemC
  - ✓Verilog
- Event Resources
  - ✓Queues
  - ✓Server
- Timed Resources
  - ✓One Queue to One Server- Timed Q
  - ✓Symmetrical: Server_N_Resource
  - ✓Combine multiple parallel resources- Server (a.k.a Smart_Timed_Resource)
  - ✓Distributed requests- System Resources

Selecting the right block- 2

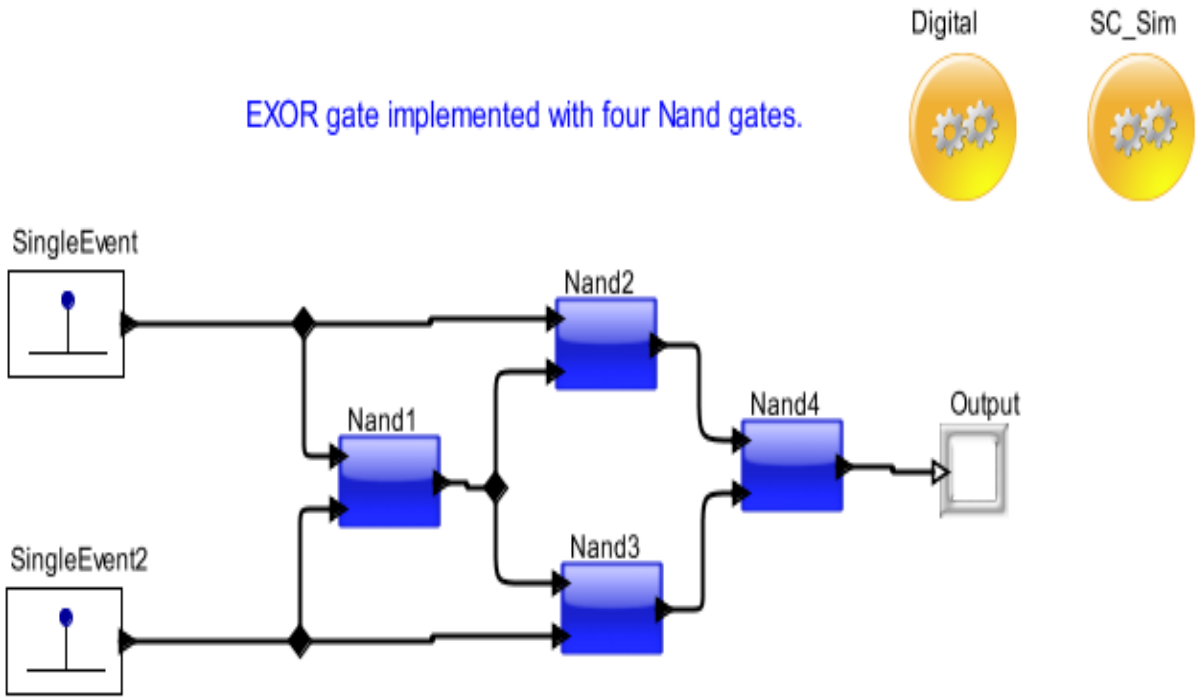
- Quantity
  - ✓ Quantity_Based
- Channel
  - ✓ Used to define logic for each Server
  - ✓ 1-to-many or 1-to-1 channels
- Hardware
  - ✓ Standard blocks
- RTOS
  - ✓ Server: SLOT type
  - ✓ Queues + Script for custom scheduling
- Behavior
  - ✓ Algorithm- Script
  - ✓ Existing algorithm- C, Java
  - ✓ Sequence- Expression List
  - ✓ Sequence with routing- Expression List
- Lookup
  - ✓ Arrays or Database
- Temporary storage
  - ✓ If content is not important, then use Queue
  - ✓ If content determines activity, use arrays

Integration with SystemC

- **Full Library -> Hardware Language -> SystemC -> SC_Sim**
- Provides timed interface between VisualSim and SystemC
- Timed interface - Synchronization between VisualSim and SystemC simulator

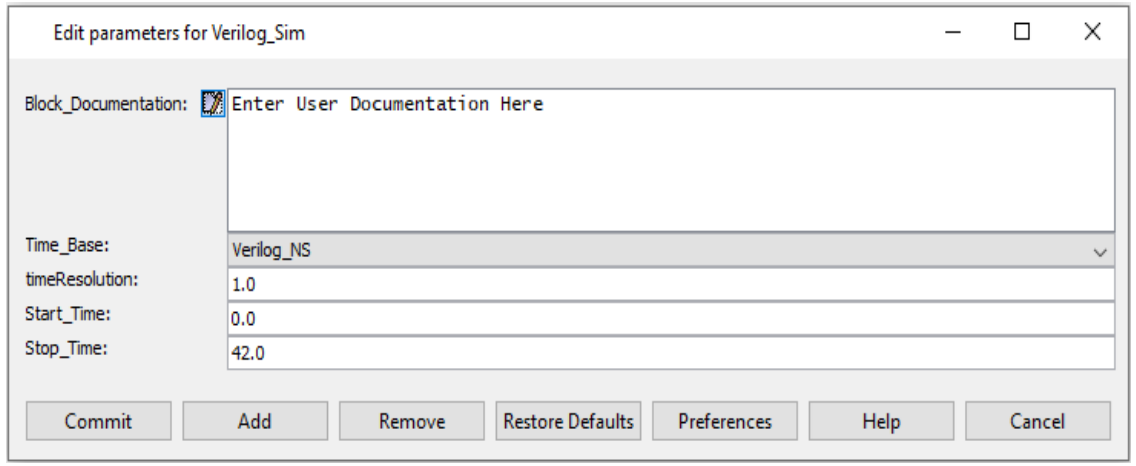


SystemC model- Example

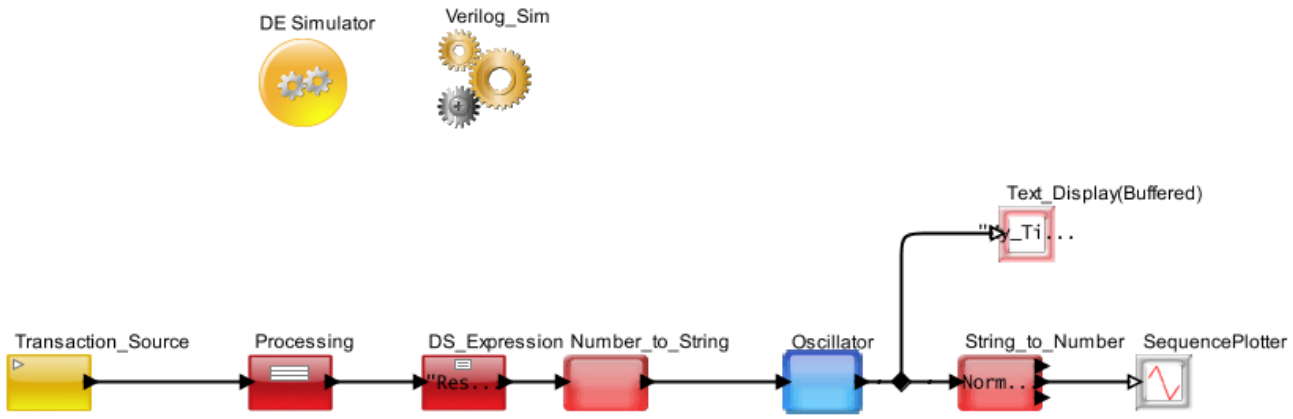


Integration with Verilog

- **Full Library -> Hardware Language -> Verilog-> Verilog_Sim**
- Provides timed interface between Visualsim and Verilog
- Timed interface - Synchronization between VisualSim and Verilog simulator



Verilog model- Example



Assembling a System Model

Selecting the Right Block

To define sensor, I/O or interface

- Traffic blocks

To define a trigger or an event

- Traffic block

To define field values or set variable values

- ExpressionList

To do simple math computation to determine delay value, extract fields for plotting

- ExpressionList

For arbitration, logic or software details

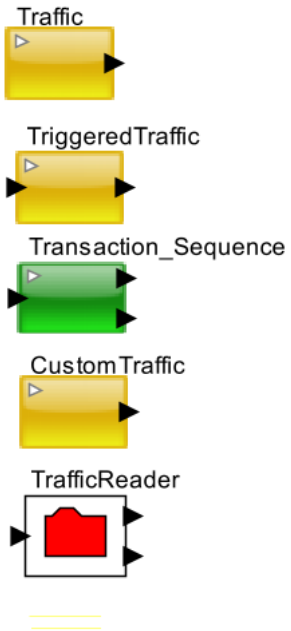
- Scripts

Traffic

Data Structure Generation

- Traffic
  - ✓ Time distribution
- TriggeredTraffic
  - ✓ Requires an input trigger to generate DS
- Transaction Sequence
  - ✓ Custom list of operations
- Custom Traffic
  - ✓ Periodic distribution
- Trace
  - ✓ Read from a file

- Using RegEX
  - ✓ **newToken(Value)**

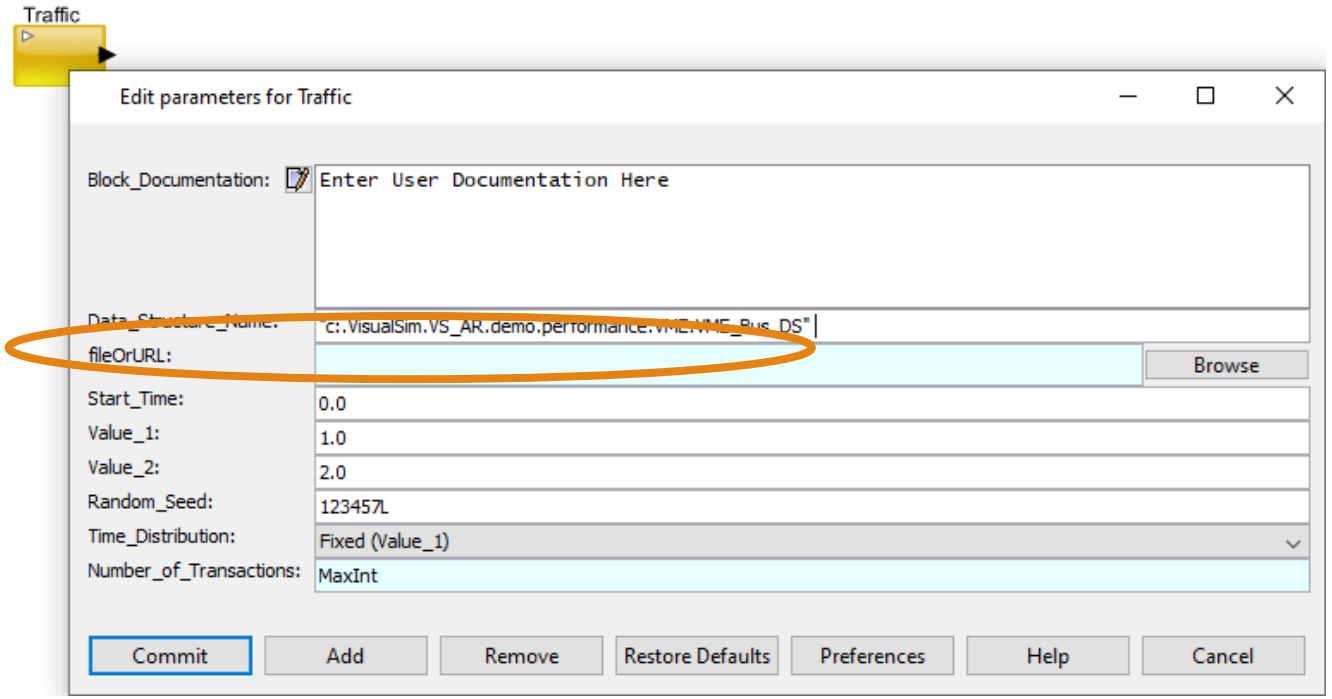


Defining Data Structure in Traffic Blocks

Data Structures template

- .txt can be located anywhere
- .class located in VS_AR/VisualSim/data

- Absolute path is required for accessing files located anywhere.
- File name if located in the $VS/VisualSim/data directory.

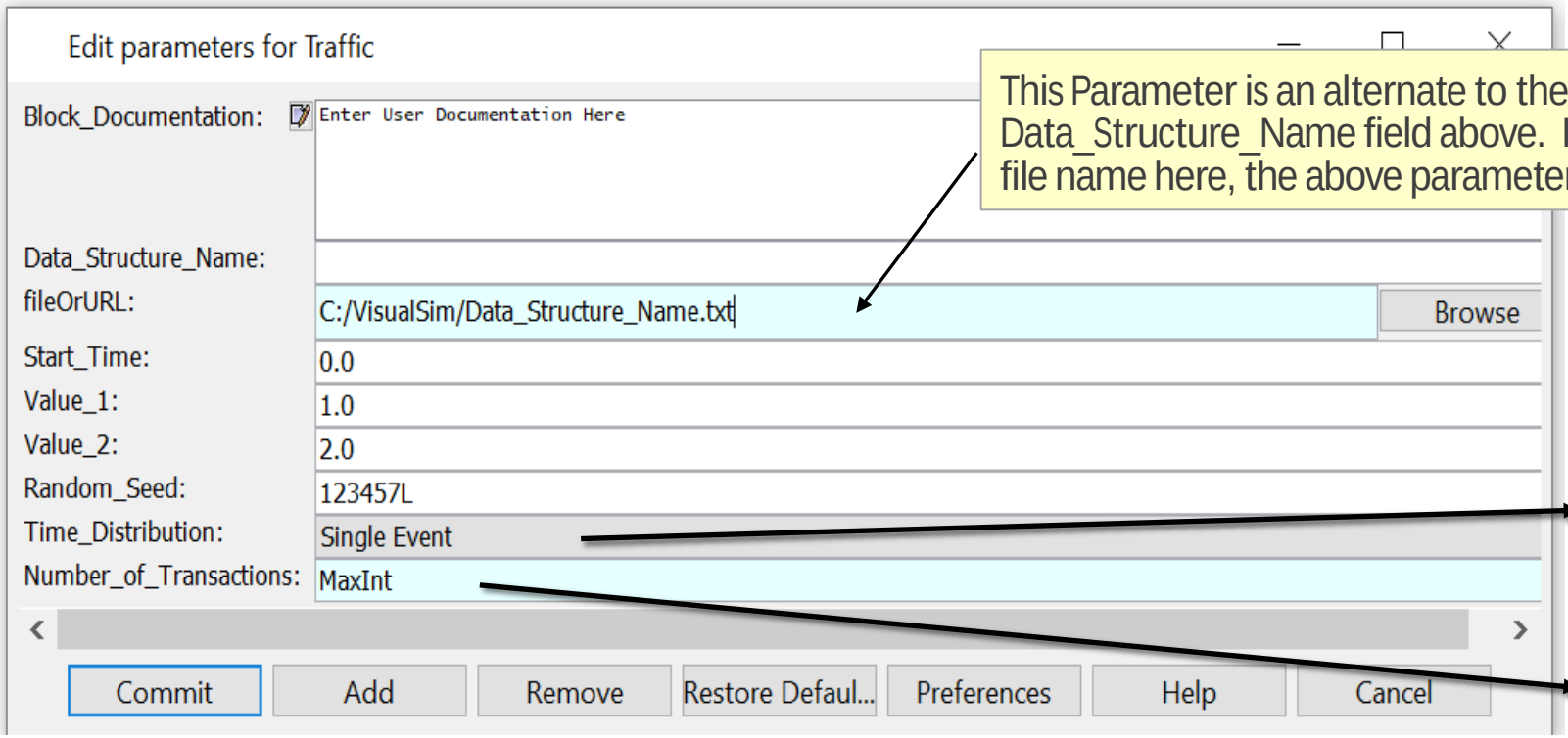


Types

- **Statistics Distribution-** Single request, periodic or fixed, uniform within a range, normal, exponential
- **Custom**- Based on a combination of data size and interface speed. Can also be triggered by external event
- **Trace file-** Existing file from hardware bus, network, software thread execution sequence, instruction order
- **Sequence**- Special case to typically debug with a order such as command of “Read, Write, Write, Read , or packet sizes of “128, 1512, 256”

Traffic

- Double click to configure



This Parameter is an alternate to the Data_Structure_Name field above. If the user defines a file name here, the above parameter is not considered.

Select the "Time_Distribution" according to design

Restrict the number of transactions

Traffic- Application

- System
  - Signal from sensors
    - ✓ Example: Application Demo->System->Functional
  - User action
    - ✓ Example: Application Demo->Automotive->Abstract SH4
  - Network packet
    - ✓ Example: Application demo->Networks->GiE
- Network
  - Input for each channel, interface or port
    - ✓ Example: Application demo->Systems->Flow Control

Traffic- Application

Hardware

- Read or Write request
  - Example: Library Demo->Hardware->Bus Switch Control->Defining Read Operation for Shared Bus. Notice the input data structure has the A_Command field updated with Read or Write.
- Instruction sequence to execute on a processor
  - Example: Library Demo->Hardware->Core Architecture->Basic Processor Model. Looking at the Single Event to start the simulation, Generate Instructions to create a stream for the Processor to execute and the Processor_Done to trigger the next stream to the Processor
- Input from Interface
  - Example: Application Demo->Processor->Xilinx PowerPC. Notice the Ethernet and PCI interface

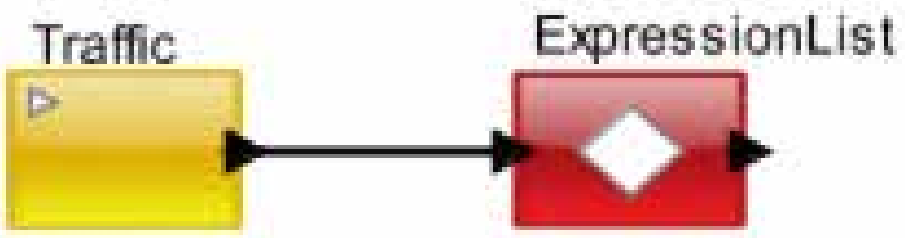
Traffic- Application

Software

- Request from SystemResource to execute
  - Example: Application Demo->Computer->RTOS->RTOS Modeling. Here there are different software threads that have different profile. The Virtual Machine is the RTOS that schedules these tasks onto different resources. The Sequence and the RTOS form the input
- Trigger a periodic execution
  - Example: Application Demo->Computer->RTOS->ARINC 653. The combination of the Transaction Source + Processing.
- Periodic execution of a System_Resource
  - Example: Application Demo->System->Flow Control. Look inside the Smart_Controller code. You will notice the delay between loop execution. This is a form of traffic. Also, the pop input to the X_On and X_Off are triggers that send data out of the Queues. The Decision block connected to the Xon_Queue has an output condition that determines if this traffic must be passed into the model

Type I - Statistical

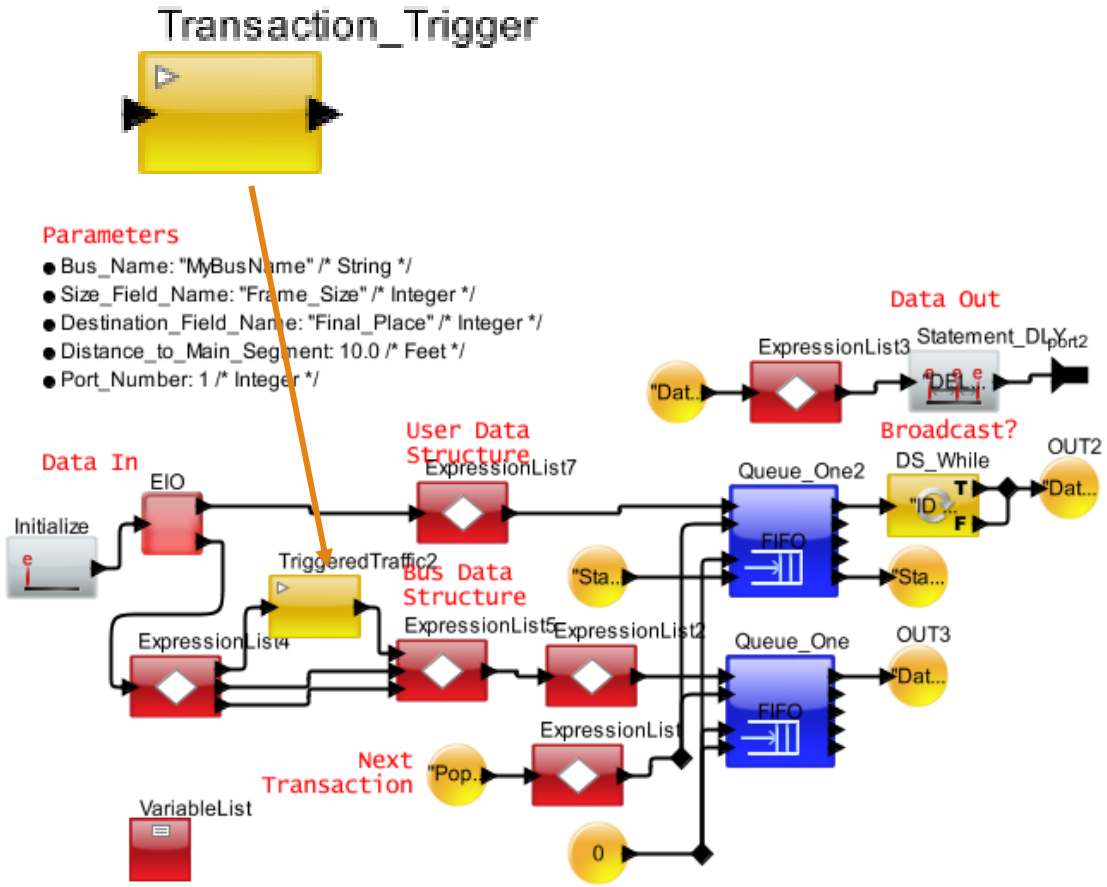
- Define a distribution
- Parameters for mean and standard distribution
- Specify values for the Data Structure fields. It can be source, destination, data, priority or bus delay



Type II - Custom

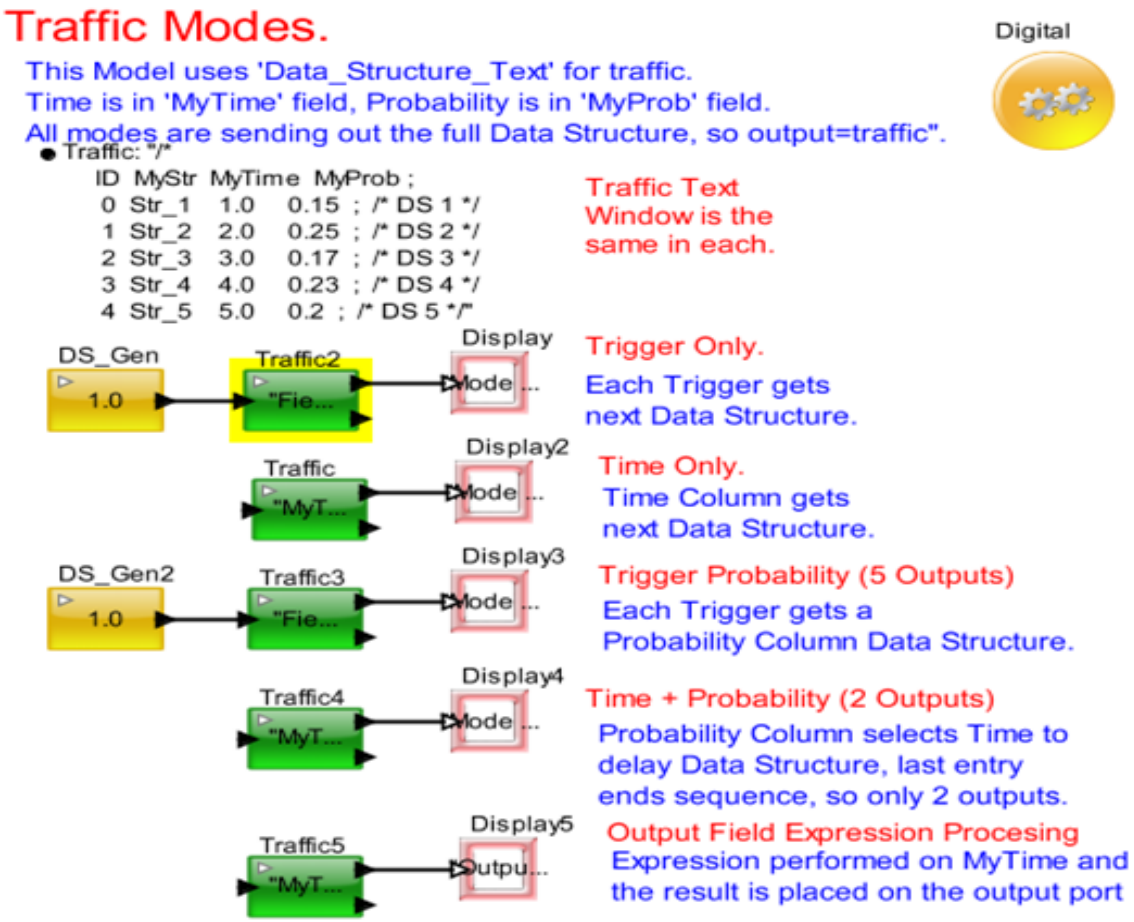
- If a custom distribution is required or the Data Structure is generated as a function of another activity, or triggered during the flow, use the **Triggered Traffic.**

- Every time the input port is triggered the Triggered Traffic block generates a transaction



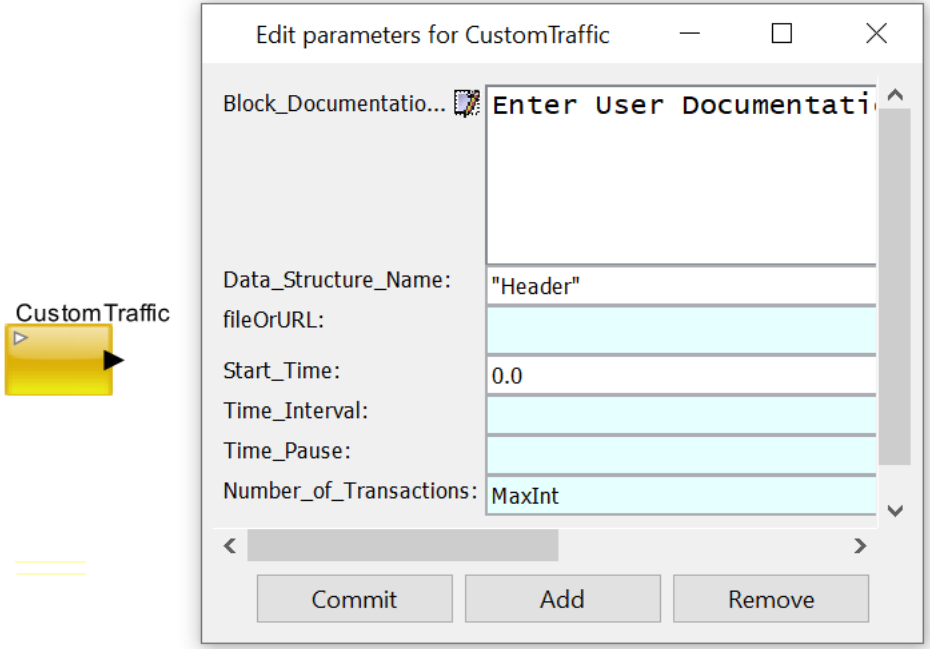
Type III – Transaction Sequence

- Generate transactions or Data Structures in a specific sequence
- Define sequences in the parameter window or specify a file + path
- Time interval between Data Structure is a parameter
- Specify an output processing using the Regular Expression (RegEx) Language



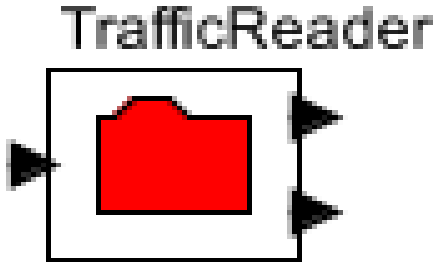
Type IV – Custom Traffic

- Generate data structure during the T_Interval period
- Stalls all transmission during the T_Pause.
- Equally distributes the Number_Of_Transactions during the T_Interval range.



Type V - Trace File Based

- Use a trace dump from the network, cache, memory or processor pipeline
- Traffic Reader block
- The expected file is
  - ✓ ASCII text
  - ✓ Any number of columns and rows
  - ✓ Each column has the first row with header and the second row with types
  - ✓ The header name is the field name
  - ✓ Any number of rows and columns are supported



Example of Trace

{InstructionCount = 11299, InstructionCycle = 57455, ICacheHit = 4730, ICacheMiss = 952, DCacheReadHit = 4012, DCacheReadMiss = 383, DCacheWriteHit = 2752, DCacheWriteMiss = 65, DDR_Read = 1335, DDR_Write = 448, SimHostTime = 0.024972, SimTargetTime = 57456, Index = 0, ioReadCount={4, 0, 0, 30, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 34}, ioWriteCount={2, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 2, 9}, processorID=0}

Traffic Creation VI: Clock based

- Use the Clock block
- Generates rising (pos) and falling (neg) edge

Traffic Creation VII: Event-and Queue- based Traffic

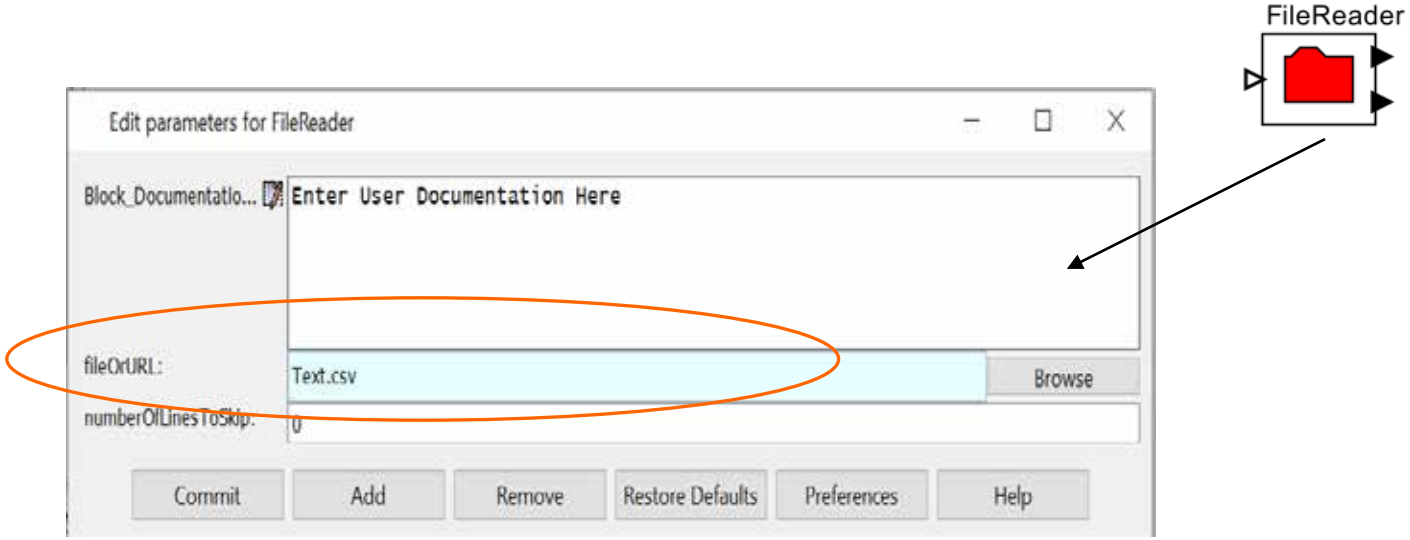
- Event
  - ✓Use the Distribution + Script block
  - ✓Traffic rate depends on the event return
  - ✓Add a field called Event_Name with a string value
  - ✓Script can WAIT or TIMEQ on Event_Name
- Queue
  - ✓Traffic rate is fixed
  - ✓Queue(a.k.a Smart_Resource) and Arbiter controls the data transfer to the Bus
  - ✓Requests can be queued or dropped based on response rate
  - ✓Can use Events for the response trigger

Read and Write from File System

File I/O

File Reader

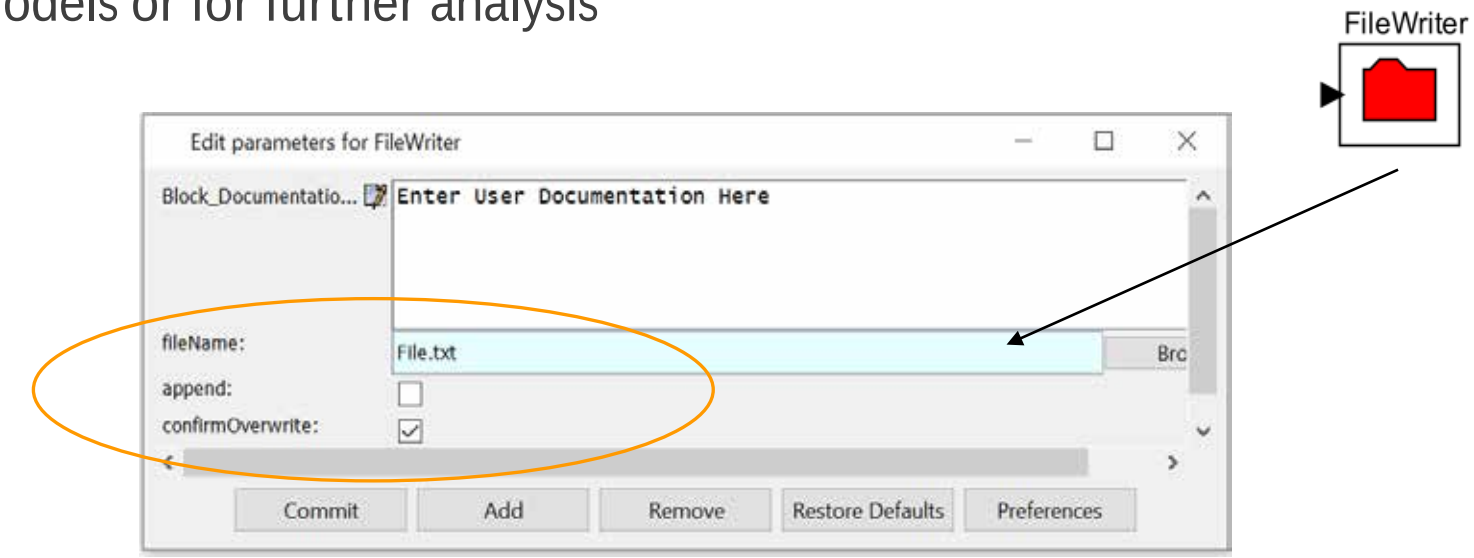
- Read from a file
- Column value can have a different data type.



File I/O

File Writer

- Write any data to a file
- Capture data structure value or model statistics for use in other models or for further analysis



Trace File Based

Trace dump from

- Network, cache, memory or processor pipeline

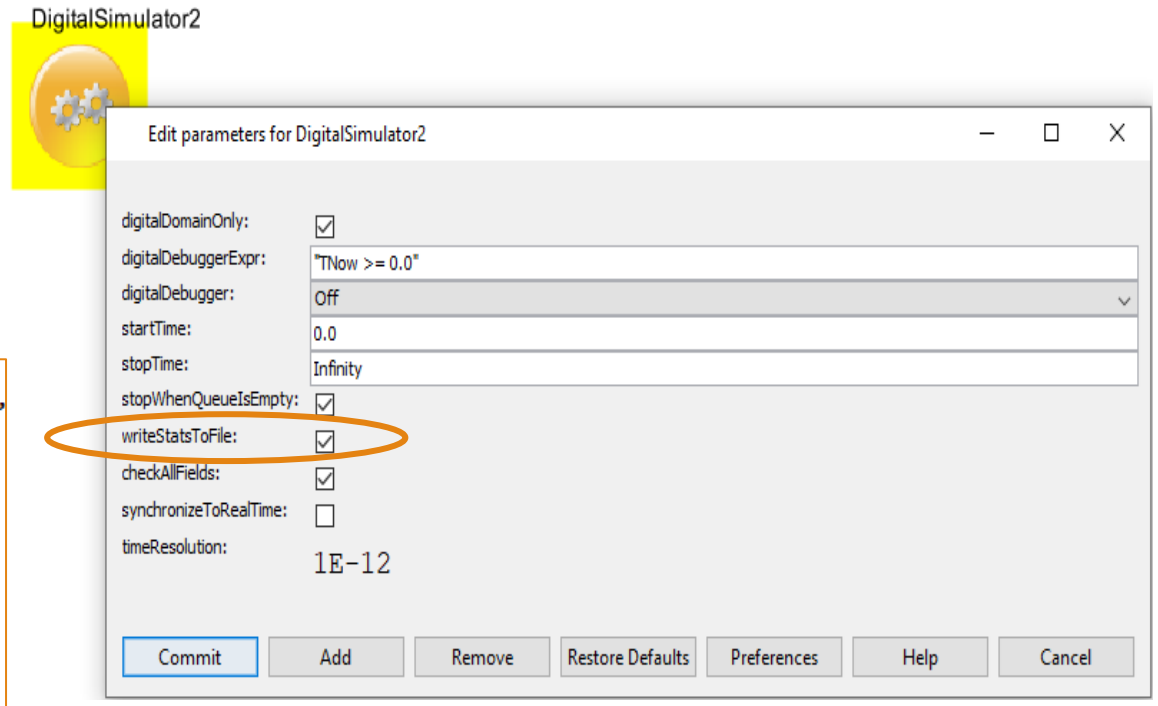
Traffic Reader

- Expected file
  - ✓ASCII text
  - ✓Any number of columns and rows
  - ✓Each column, first row with header, and second row with types
  - ✓The header name is the field name
  - ✓Any number of rows and columns are supported

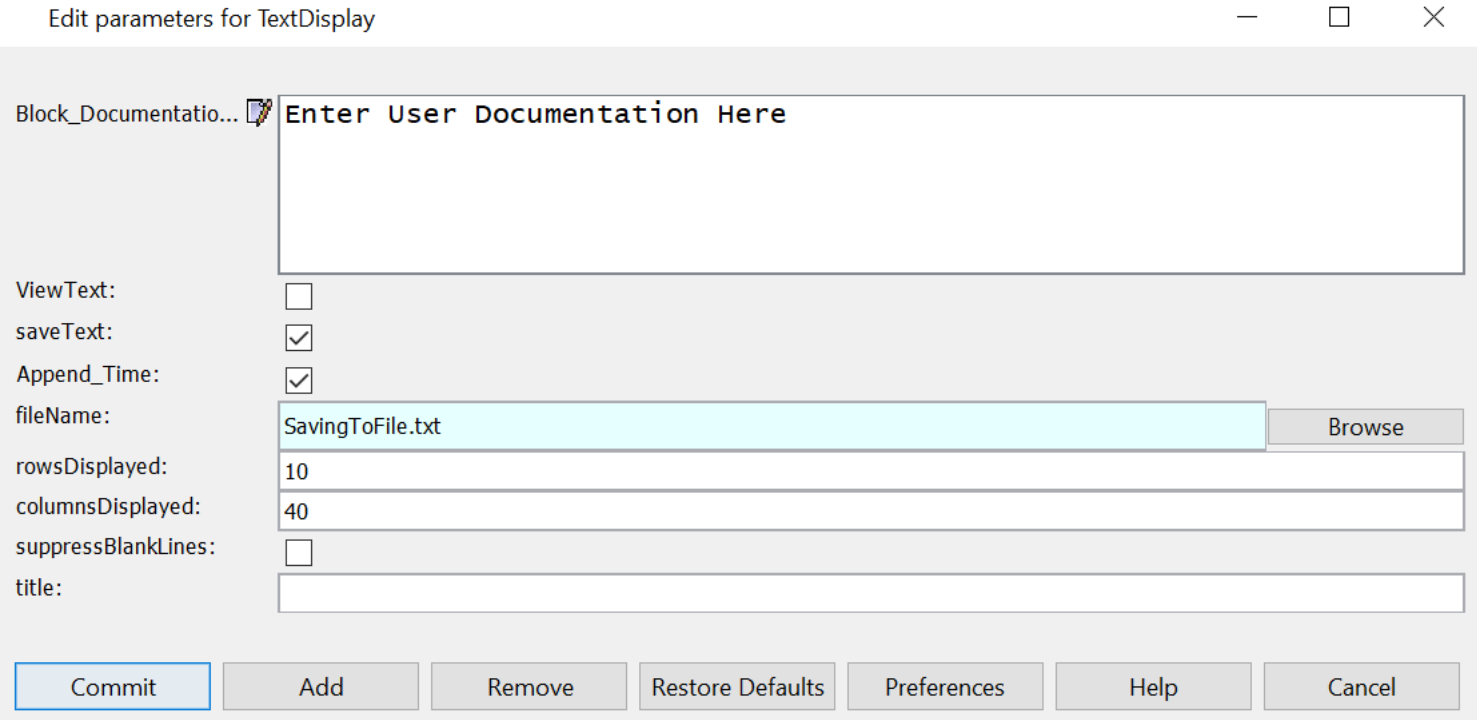
writeStats To File

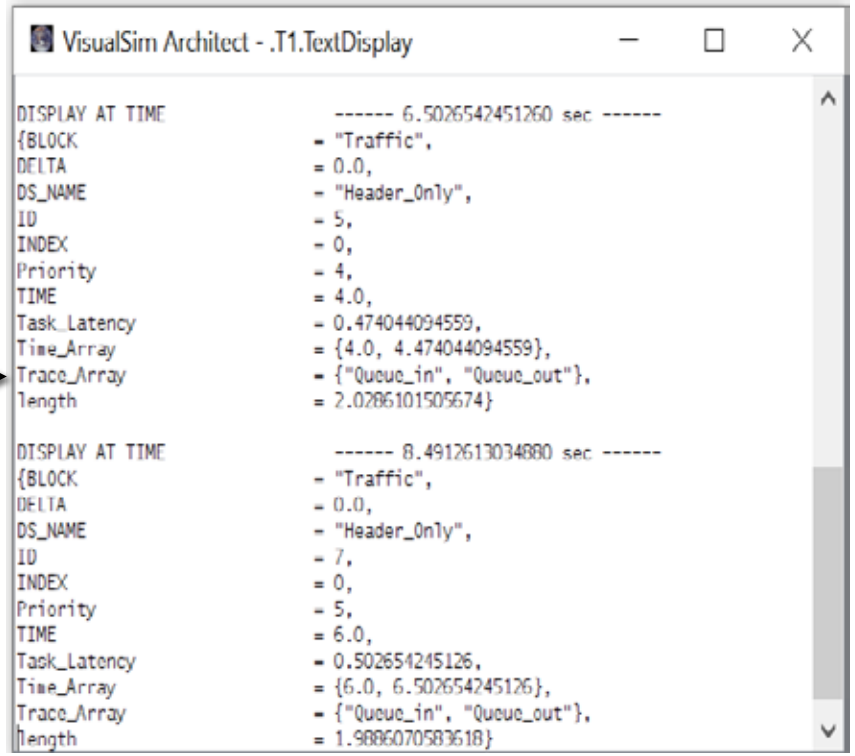
- Generates Statistics for all the blocks in the model at the end of simulation
- Writes into a Text File in the model directory

```
Queue_Statistics                6.00000000000 sec
{BLOCK                        = "SR_SrExtend_example.SystemResource_Extend",
DELTA                         = 0.0,
DS_NAME                       = "Queue_Common_Stats",
ID                            = 1,
INDEX                         = 0,
Number_Entered                = 7,
Number_Exited                 = 1,
Number_Rejected               = 0,
Occupancy_Max                 = 6.0,
Occupancy_Mean                = 3.7777777777778,
Occupancy_Min                 = 1.0,
Occupancy_StDev               = 1.4740554623802,
Queue_Number                  = 1,
TIME                          = 6.0,
Total_Delay_Max               = 4.0,
Total_Delay_Mean              = 4.0,
Total_Delay_Min               = 4.0,
Total_Delay_StDev             = 0.0,
Utilization_Mean              = 0.0}
```



SaveText in Plotters and TextDisplay





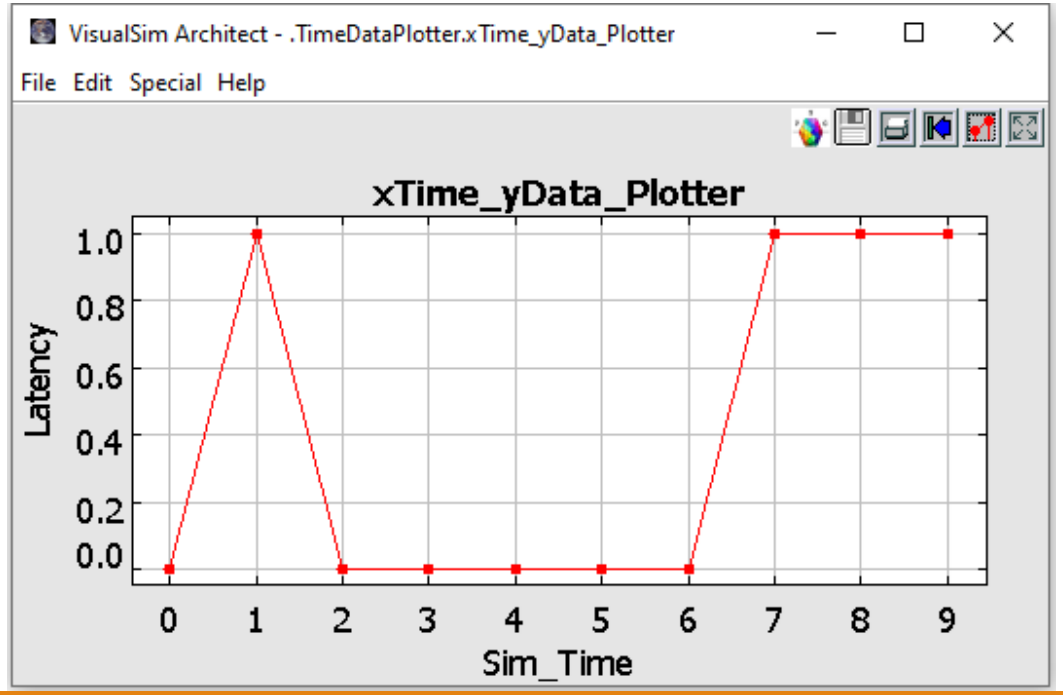
Plotting, Displays and Statistics

Result

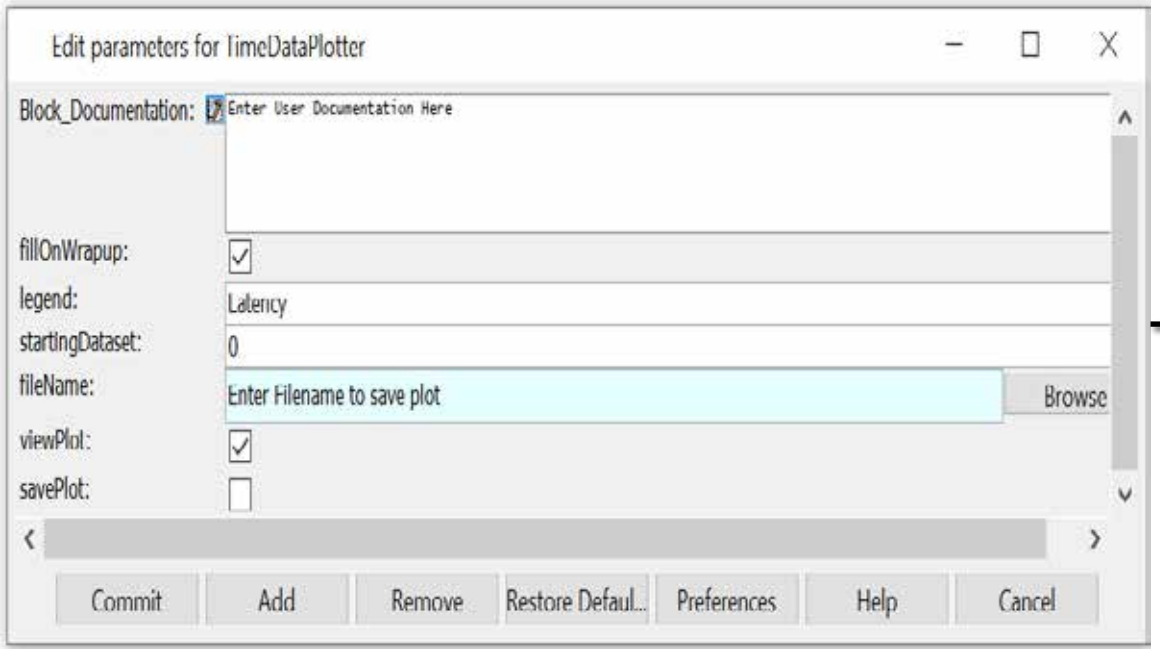
- **Statistics**
  - ✓ ResourceStatistics
  - ✓ Statistics blocks to collect statistics at intermediate points
- **Assertions or tests**
  - ✓ High/low value for scalar
  - ✓ Conditional model activity
  - ✓ Model termination
- **Collect data**
  - ✓ Write to screen or to files (Excel, text or XML)
- **Plot data**
  - ✓ Bar, Histogram or XY plots
  - ✓ Special viewers- Matrix, Image, MPEG and speakers
- **3D- Interactive Creation**
  - ✓ Create custom animated views that resemble the system

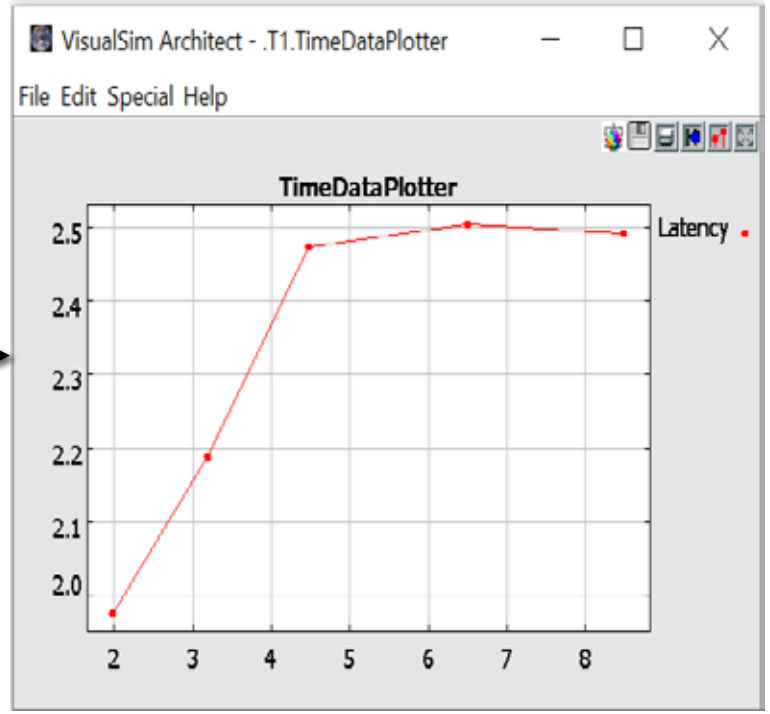
TimeData Plotter

- Plot double values against simulation time
- View or save the results of the simulation in a XY format.
- Used to depict latency, throughput and other variables that vary against time.



TimeData Plotter

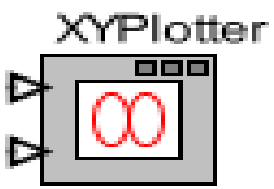


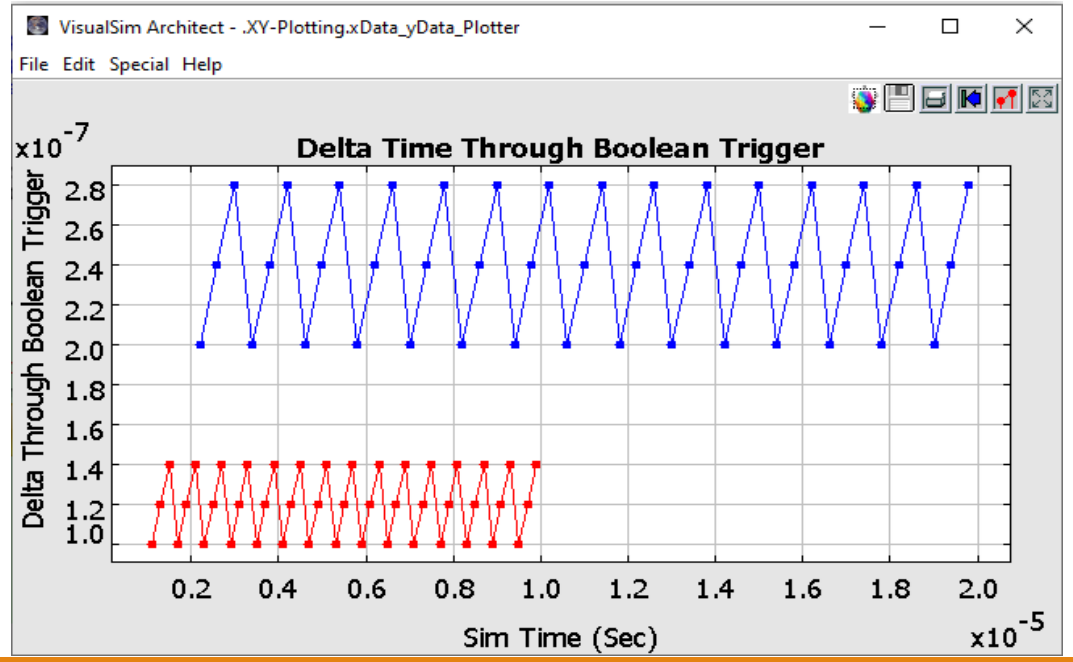


Return

XY Plotter

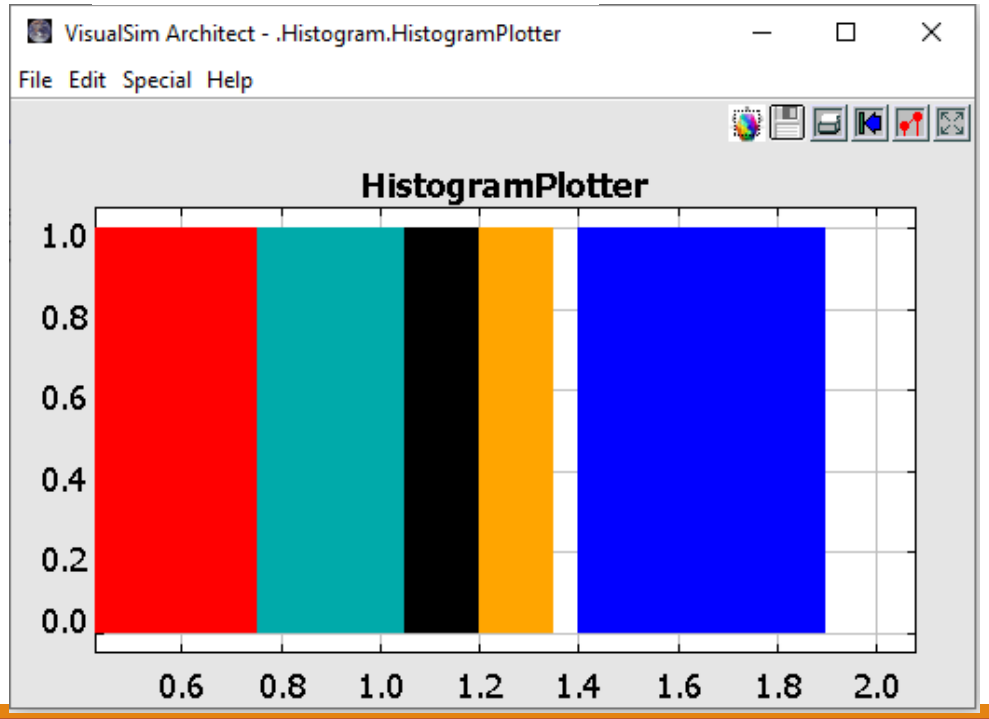
- Any scalar value against any other scalar value. Both values must arrive synchronously.
- The X- and Y-axis can have different data values.
- Plots can be Latency vs. Packet Size or Task Delay vs. Processor Speed.
- The parameters of this block match the fields (or RegEx) of the incoming Data Structure to determine the coordinates, color and trace identifier (Dataset).
- Values, color, legend defined in fields of incoming data structure. Plot similar to XYPlotter





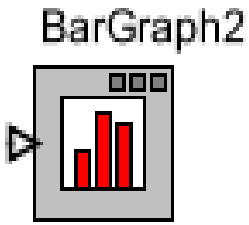
Histogram

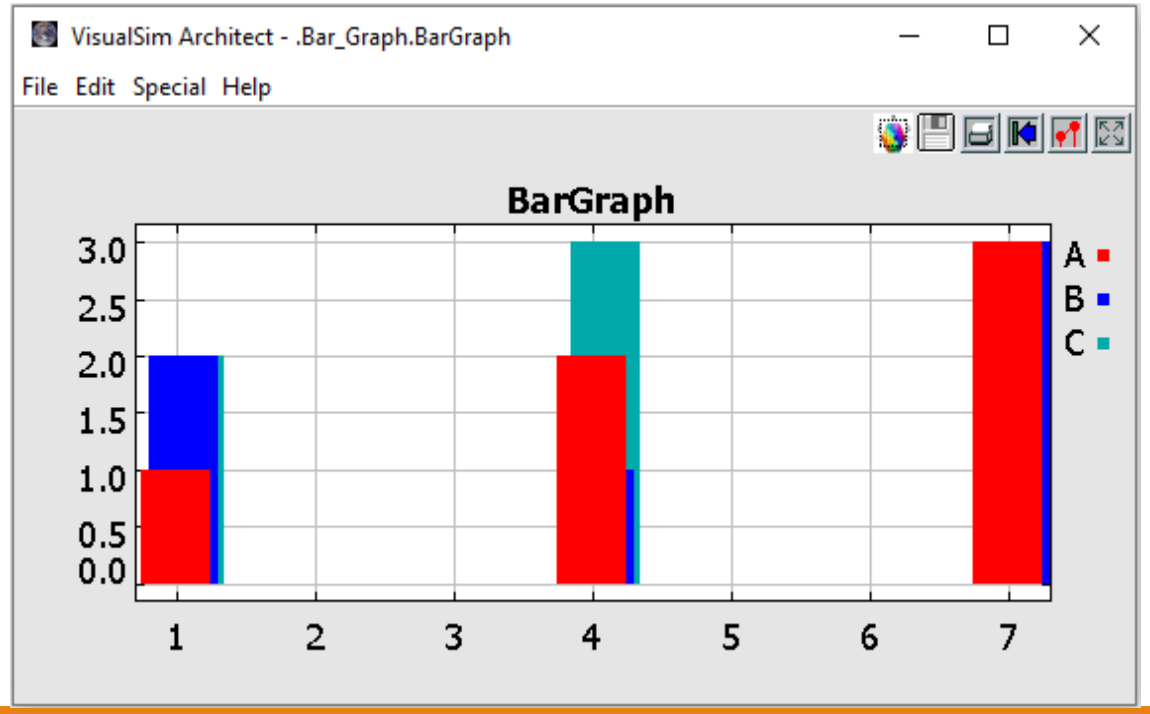
- The plotter accepts data on the input and plots them as a histogram.
- View the plot in real-time or save for future viewing.



Bar Graph

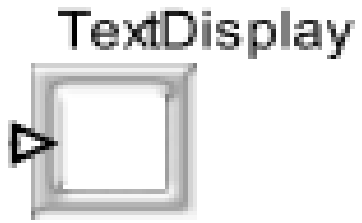
- The bar graph plots series
- The input is an array

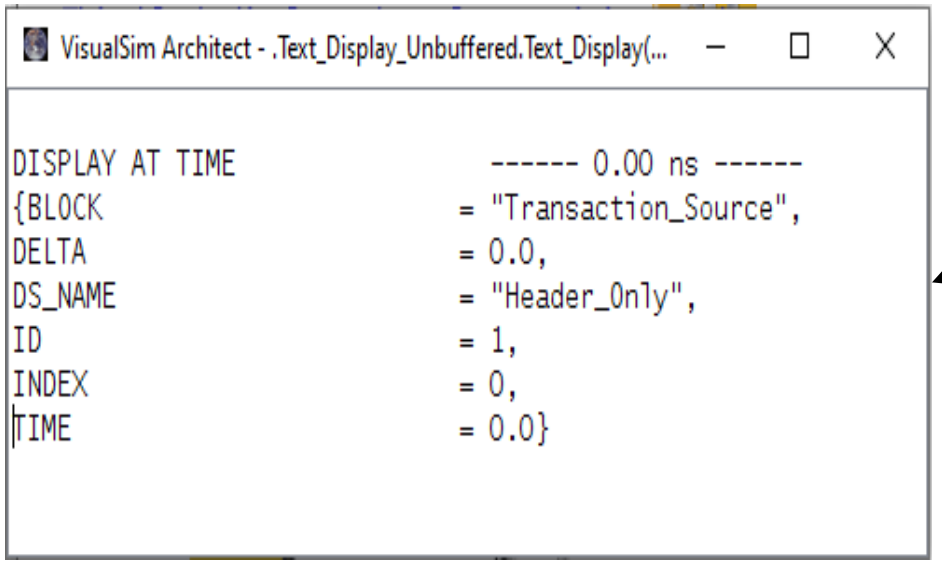


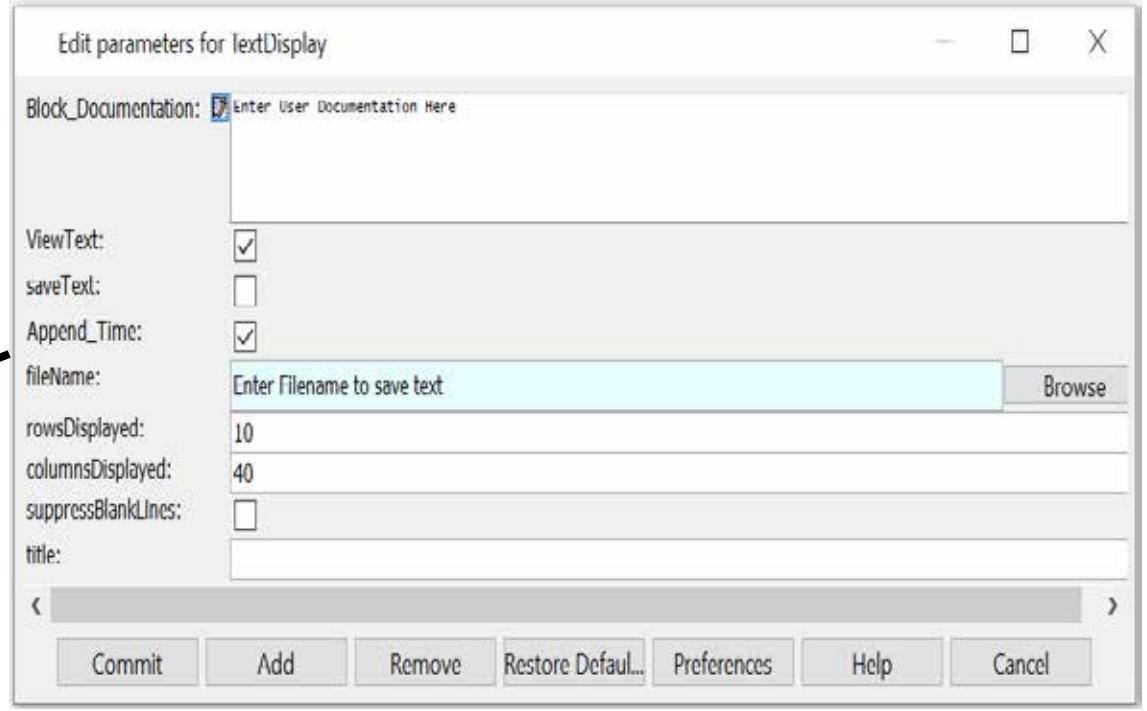


Text Display

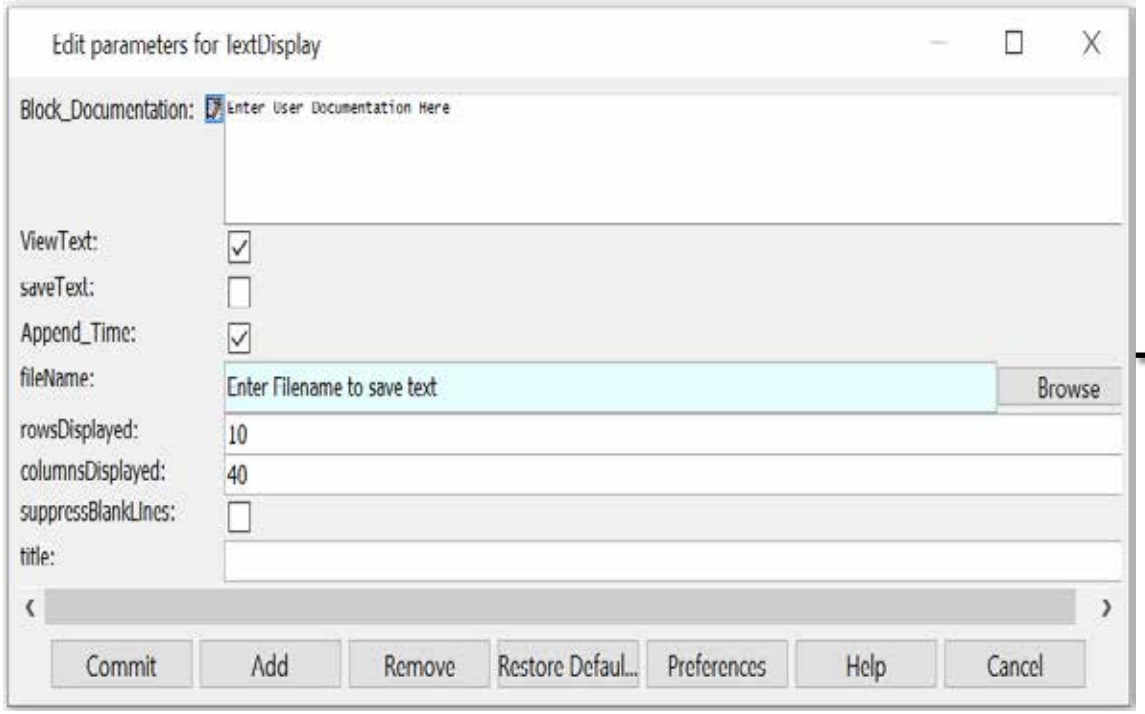
- Output to text the data structure and statistics
- The input type can be of any type.
- Can be set to Save/View from Post Processor
- Cannot be viewed from the Post Processor

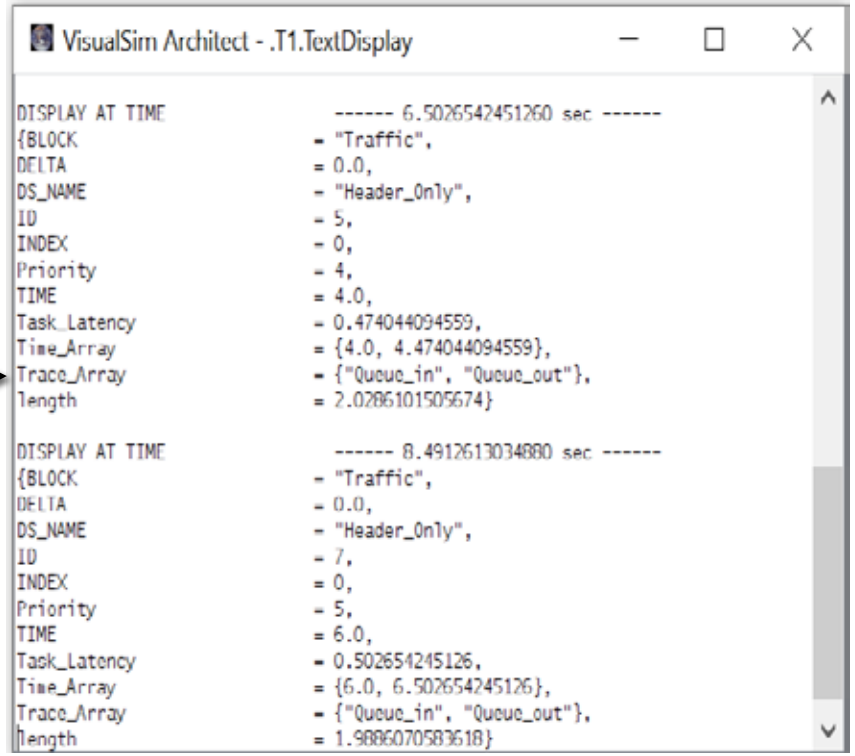






Text Display

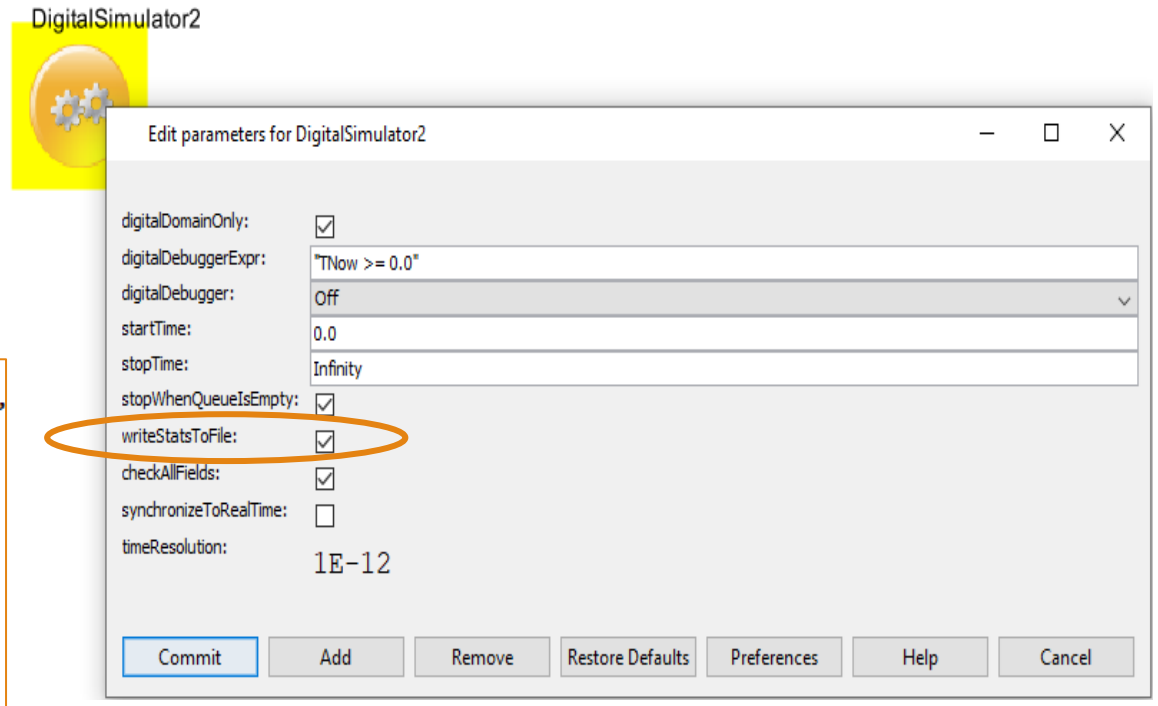




writeStats To File

- Generates Statistics for all the blocks in the model at the end of simulation
- Writes into a Text File in the model directory

```
Queue_Statistics                6.00000000000 sec
{BLOCK                        = "SR_SrExtend_example.SystemResource_Extend",
DELTA                         = 0.0,
DS_NAME                       = "Queue_Common_Stats",
ID                            = 1,
INDEX                         = 0,
Number_Entered                = 7,
Number_Exited                 = 1,
Number_Rejected               = 0,
Occupancy_Max                 = 6.0,
Occupancy_Mean                = 3.7777777777778,
Occupancy_Min                 = 1.0,
Occupancy_StDev               = 1.4740554623802,
Queue_Number                  = 1,
TIME                          = 6.0,
Total_Delay_Max               = 4.0,
Total_Delay_Mean              = 4.0,
Total_Delay_Min               = 4.0,
Total_Delay_StDev             = 0.0,
Utilization_Mean              = 0.0}
```



Behavior Modeling

Processing

Data Flow

- Describe actions based on expressions
- Results stored in DS fields if required by transaction
- Results stored in variables if required elsewhere in model

Control Flow

- Model if-else; while, and case-switch

Virtual Flow

- Move data to other parts with IN or Script name
- Use Mux and Demux to create instruction decodes, protocol switching, broadcast etc.

Delay

- Simple delay before output

- Switching
  - ✓ Control the flow of data through model
- Execution Control
  - ✓ Control simulation based on model results, activity or triggers
  - ✓ Combine blocks with multi port ExpressionList to establish assertions
- Mapping
  - ✓ Connect processing flow with the resources
  - ✓ Send to SystemResource and Processor
  - ✓ Multiple Mappers can send to a resource

Expression List

Sequence of mathematical expressions

Requires one transaction on all input ports to fire the block

Assign values to fields or variables

- Execution starts when data arrives on the input ports
- Data at each port is identified by the port name
- Queued if multiple values arrive at one port

Usage

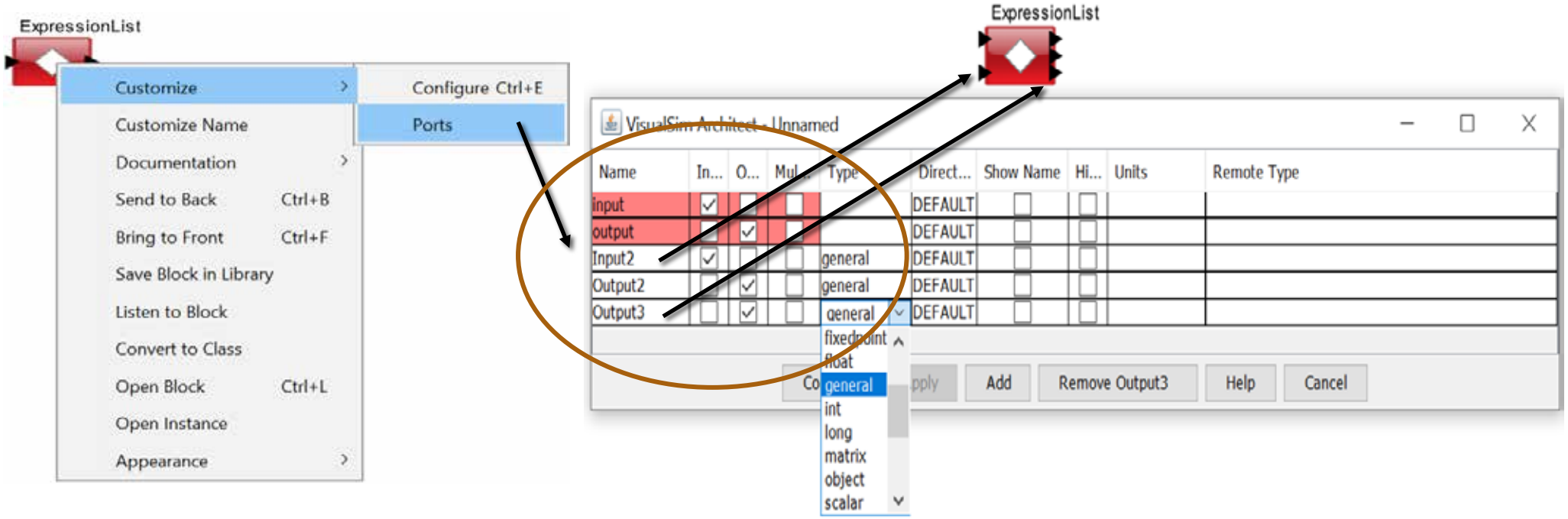
- input.field_name = value

Output

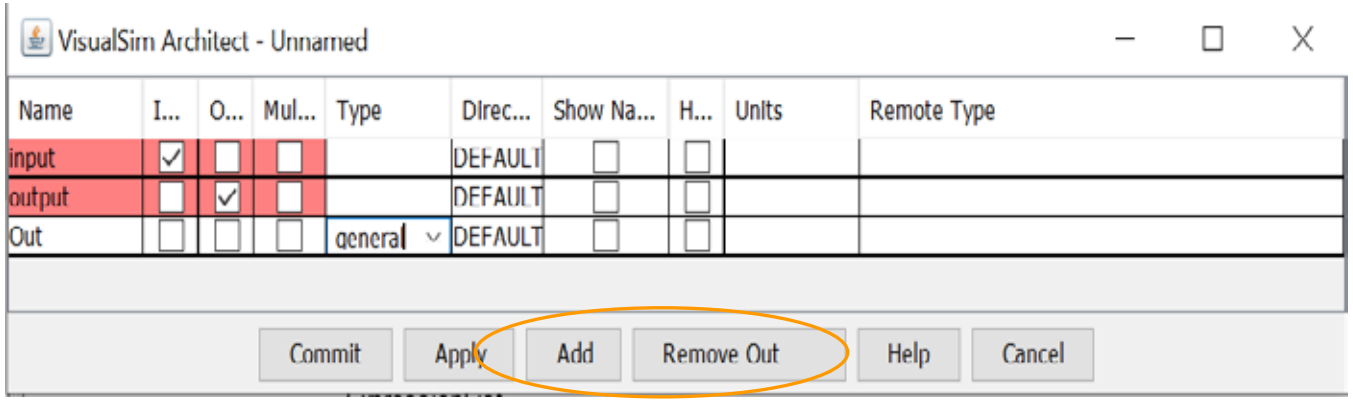
- Condition can use any expression containing fields, variables and logical operators

Expression List (Cont.)

Create multiple input ports and output ports
Avoid setting data types unless a restriction is required



Expression List (Cont.)

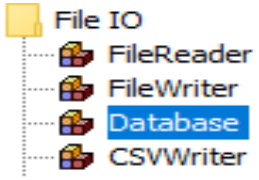


To add or remove ports

Database

Database

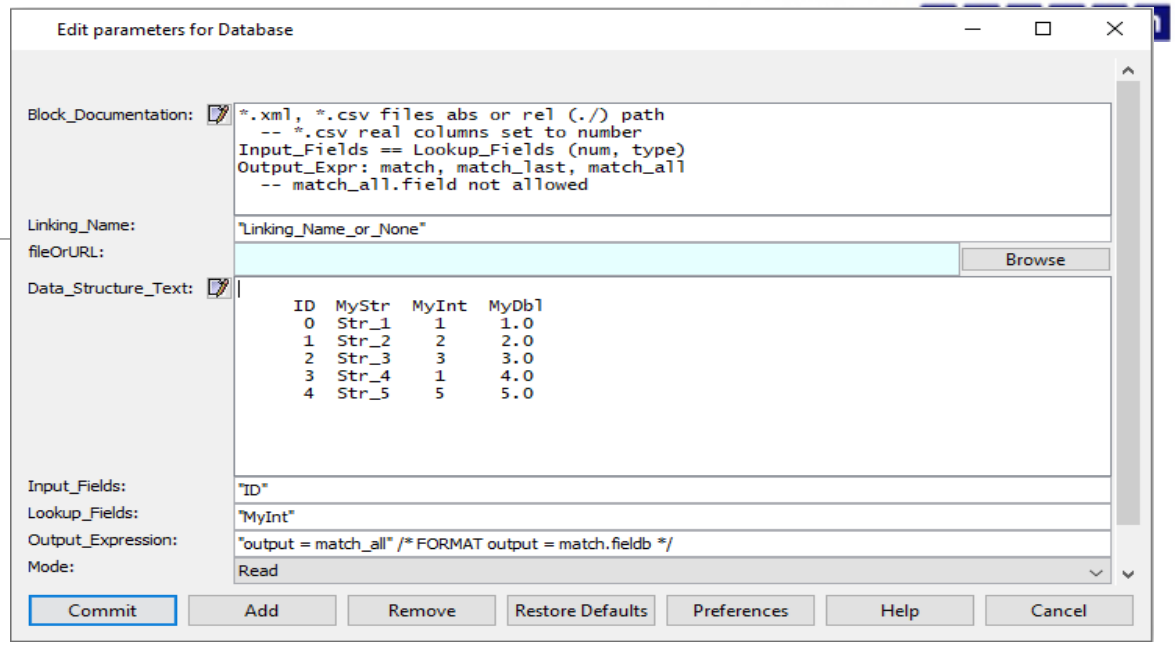
- Lookup table for searches
  - Used as Routing Table
- Main features
  - Read
  - Write
  - Remove

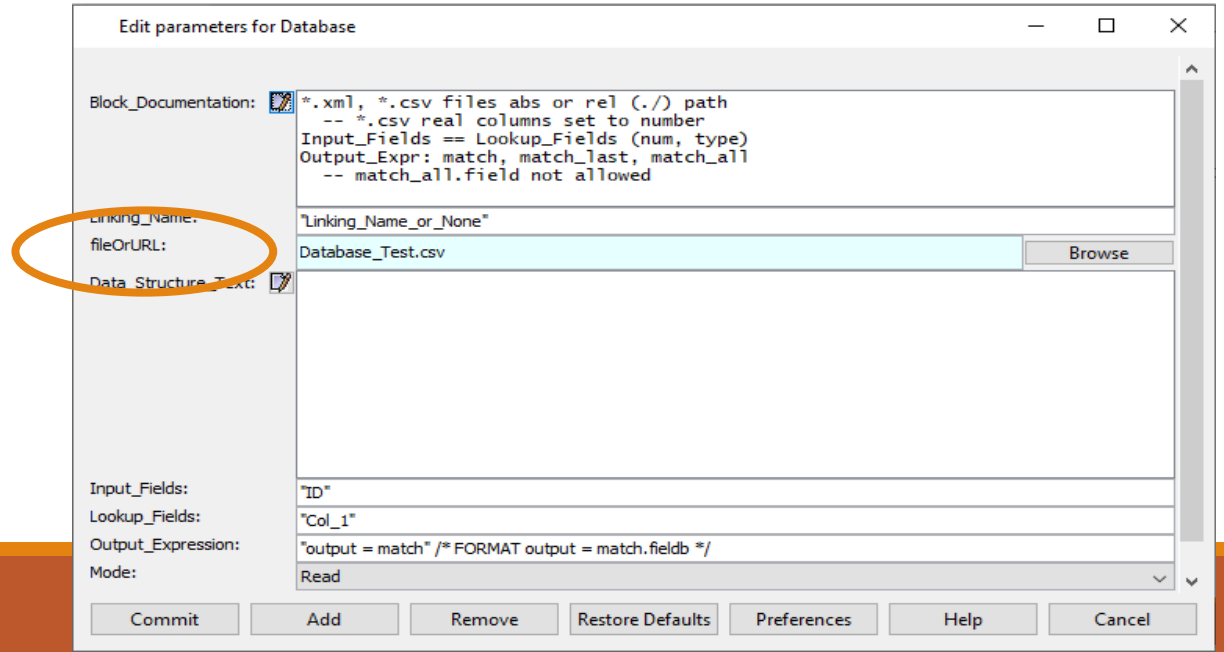


dest	source	sequence	hop
"Scheduler_1"	"Scheduler_1"	0	5
"Scheduler_1"	"Scheduler_2"	0	6
"Scheduler_2"	"Scheduler_3"	0	7
"Scheduler_3"	"Scheduler_1"	0	8

- Database content can be defined in
  - ✓ Data_Structure_Text window
  - ✓ FileorURL
    - ✓ Text File
    - ✓ CSV file
    - ✓ XML file

- Linking name enables multiple Database blocks to share one database content
  - One block maintains the values
  - Other blocks use extern in the Data_Structure_Text





Operation

- Parameters
  - ✓ Input Fields : List of fields in incoming Data Structure
  - ✓ Lookup Fields : List of Column names in the Database
  - ✓ Output Expression : Used to define the match type and value
- Values in Input fields are matched with values in the Lookup Fields in order
  - The number of fields in the input and lookup must match
- According to the match type in the output Expression parameter the data is sent on the output port
  - ✓ Match – Reads the first row match from the database
  - ✓ Match_all – Reads all the rows that matches from the database
  - ✓ Particular value – eg : match.field_a

Read Mode

- FindsSends out the matched row as a Data Structure

- Two types of Matches possible:
  - ✓ Match
  - ✓ Match all

Write Mode

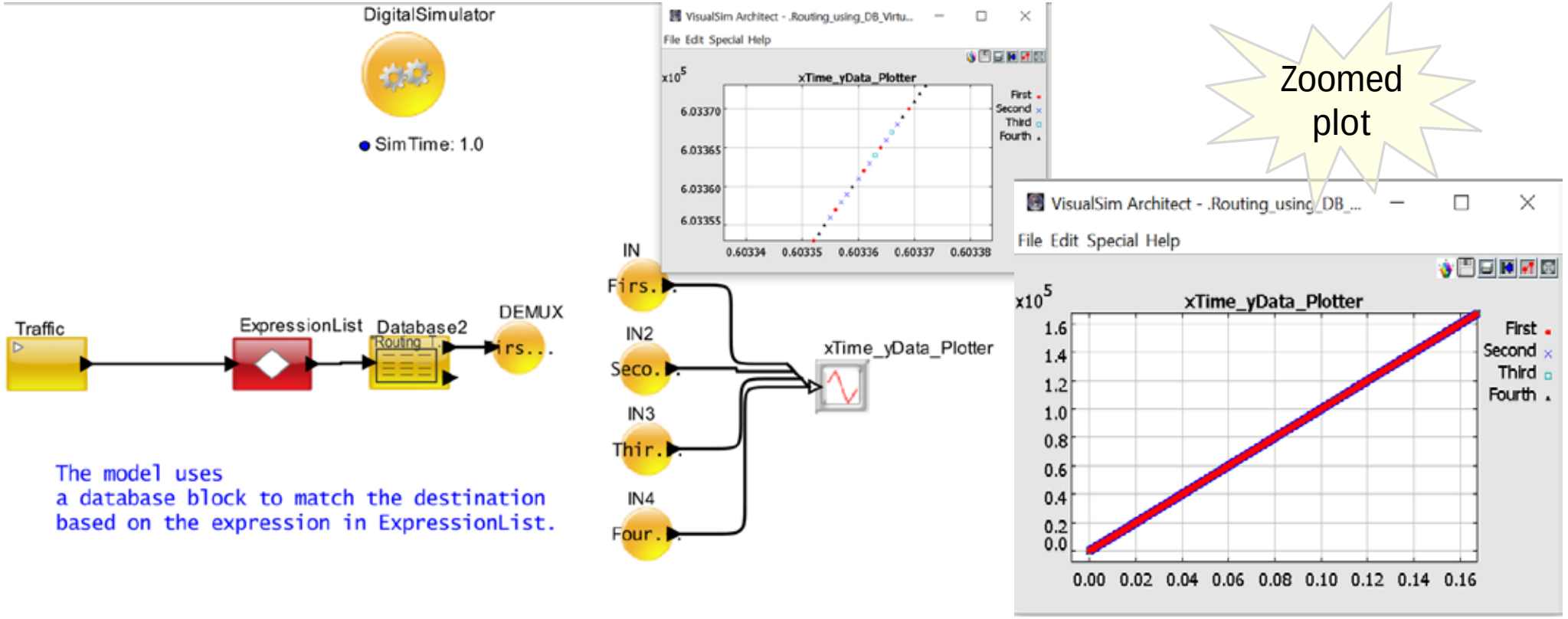
- Replace the matching row with the values in the input Data Structure
- If no match found, append at the end of the table
- Match_all is not possible

Remove Mode

- If a row matches, remove that row

- Match_all is possible

Demo for database model



Return

Database – Virtual Concept

- Content of the Database can be accessed as an array of data structures from any ExpressionList or Script block using **RegEX** functions
- Multiple Databases can reference a single Database

Database Access Using RegEX

- getResourceActivity( )
  - ✓ Gives the current queue length of the Resources under the named column
- getNextResource( )
  - ✓ Get the first resource (Processor, SystemResource and SystemResource_Extend) in the listed column that is not currently processing any transaction. If none of the resources are free, then the function returns "none"
- getColumn( )
  - ✓ Get an Array of Column entries from Database block, based on Column Name
- getRows( )
  - ✓ Returns an array of data structures for each row that match the value in the named column.
- getCell( )

  Usefull if we only want a particular value rather than an array of values

dest	source	sequence	hop
"Scheduler_1"	"Scheduler_1"	0	5
"Scheduler_1"	"Scheduler_2"	0	6
"Scheduler_2"	"Scheduler_3"	0	7
"Scheduler_3"	"Scheduler_1"	0	8

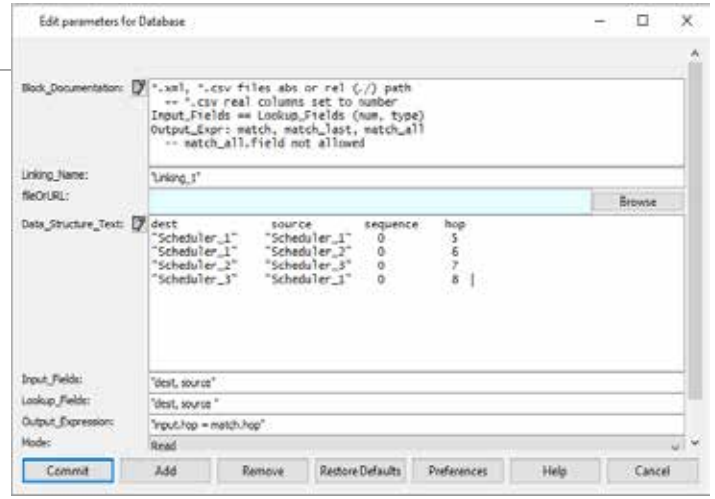

```
input.Get_Column        = getColumn("Linking_1","dest")
input.Get_Next_Resource = getNextResource("Linking_1","dest")
input.Res_Activity      = getResourceActivity("Linking_1","dest")
input.Get_Rows          = getRows("Linking_1","Scheduler_1","dest")
input.Get_Rows_1        = getRows("Linking_1","Scheduler_1","dest","Scheduler_3","source")
```

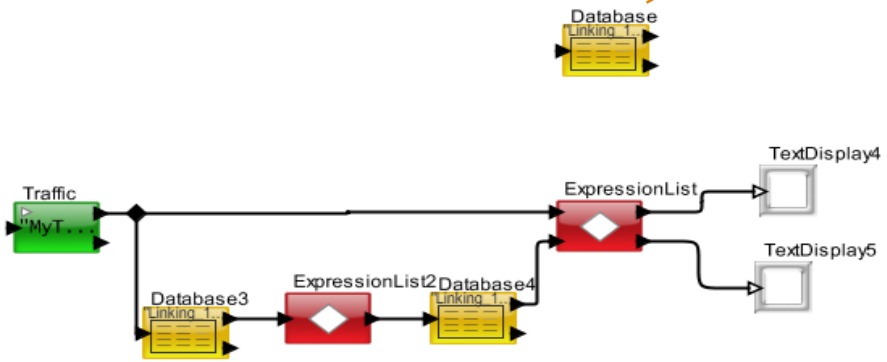
RegEX Results

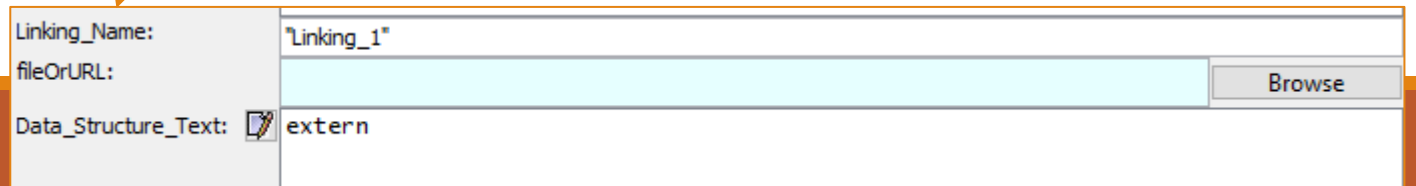
```
DISPLAY AT TIME                 ------ 1.00000000000 sec ------
{BLOCK                        = "Traffic3",
DELTA                         = 0.0,
DS_NAME                       = "Header_Only",
Get_Column                    = {"Scheduler_1", "Scheduler_1", "Scheduler_2", "Scheduler_3"}
Get_Next_Resource             = "Scheduler_2",
Get_Rows                      = {{BLOCK                        = "Database3",
DELTA                         = 0.0,
DS_NAME                       = "DS_Database3",
ID                            = 0,
INDEX                         = 0,
TIME                          = 0.0,
dest                          = "Scheduler_1",
hop                           = 5,
sequence                      = 0,
source                        = "Scheduler_1"}, {BLOCK                        = "Database3",
DELTA                         = 0.0,
DS_NAME                       = "DS_Database3",
ID                            = 1,
INDEX                         = 0,
TIME                          = 0.0,
dest                          = "Scheduler_1",
hop                           = 6,
sequence                      = 0,
source                        = "Scheduler_2"}},
Get_Rows_1                    = {"none"},
ID                            = 1,
INDEX                         = 0,
Res_Activity                  = {Scheduler_1 = 9, Scheduler_2 = 0, Scheduler_3 = 0},
TIME                          = 1.0}
```

Database Reference

- The content of the single Database can be accessed by multiple Databases using the Keyword **Extern**
- All these database blocks read/write/remove this single database
- The content is instantly updated in all the Database blocks.







FSM

- **Controller**
  - ✓ FSM has states and transitions
  - ✓ Controls the processing of linked hierarchical blocks in the same Window
  - ✓ Must contain the FSM Simulator and use the FSM_Controller
  - ✓ Refinement of states and transitions is the firing of one or more of these hierarchical block

- **Hierarchical**
  - ✓ Functions as a hybrid simulation domain
  - ✓ Depends on top-level simulator to control time and data flow
  - ✓ State and Transition actions are defined as block diagrams/ FSM
  - ✓ Multiple refinements can be applied to each state or transition

Concept of Virtual Connections

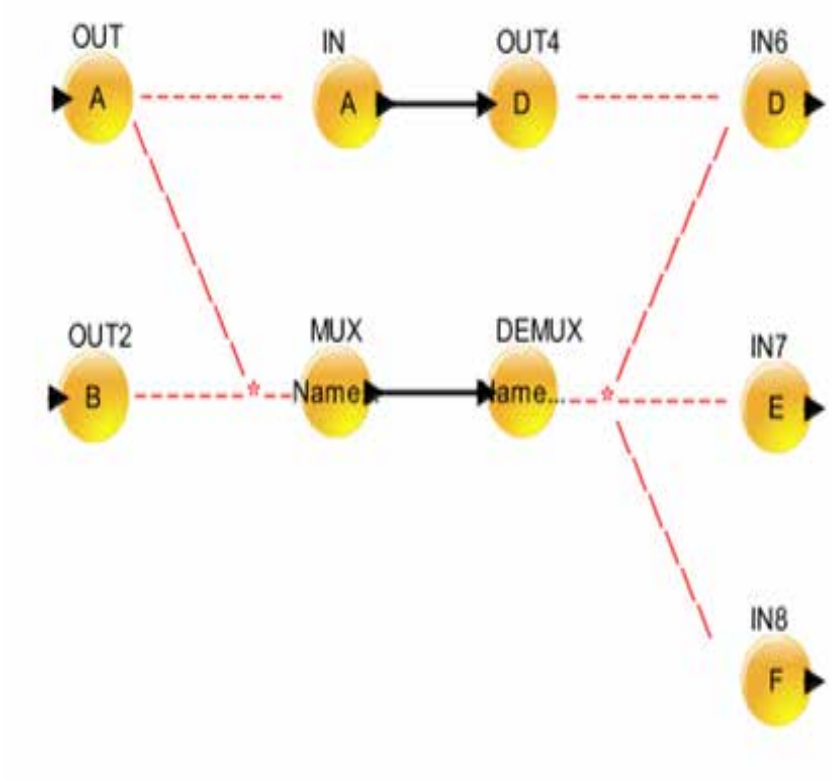
Virtual Connections

- Connections made using Names and not wires
- Multi-function
  - ✓ Mux and demux for routing and decisions
- Names accessed in memories or fields
- Different Virtual Connections in VisualSim
  - ✓ IN and OUT
  - ✓ Mapper to System Resource
  - ✓ Mapper to Processor
  - ✓ Script block to Script block
  - ✓ Virtual Concept in Database block

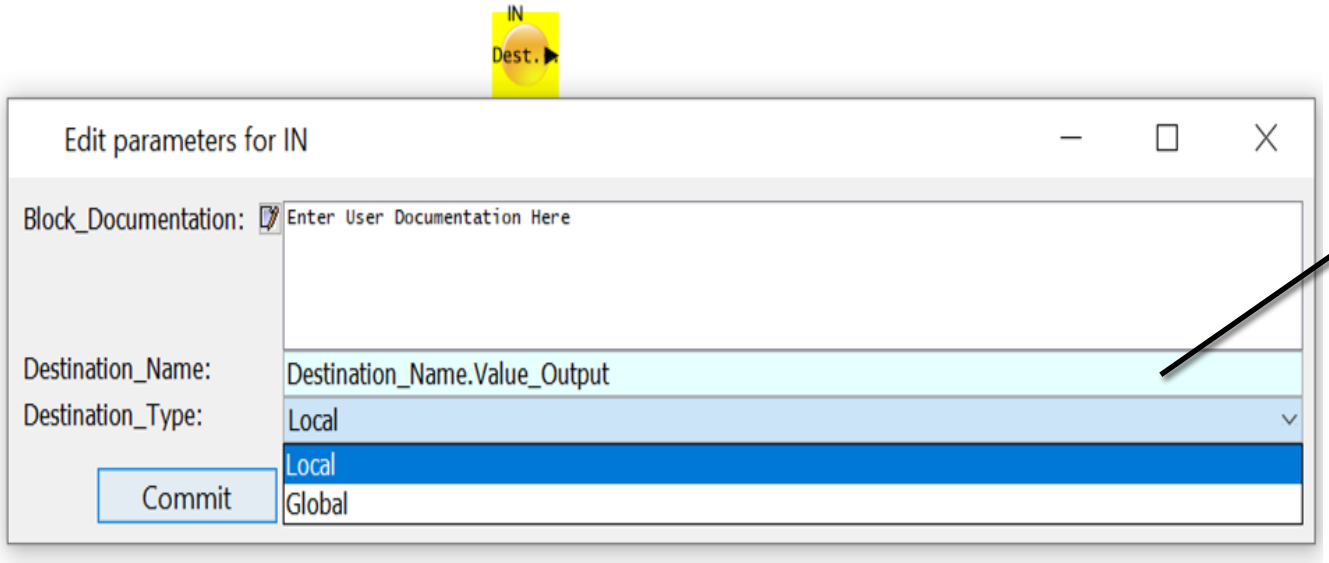
IN, OUT, MUX, DEMUX (Bubble Blocks)

- The **OUT** block accepts Data Structures or token arriving on the input port and transmits it as a virtual connection to:
  - ✓ 'IN' and 'MUX' that match the Destination_Name.
  - ✓ 'NODE', Script block with the same name.

- The **IN** block accepts incoming Data Structures or tokens from any OUT/MUX/uEngine/Virtual_Machine blocks and sends a value on the output port.



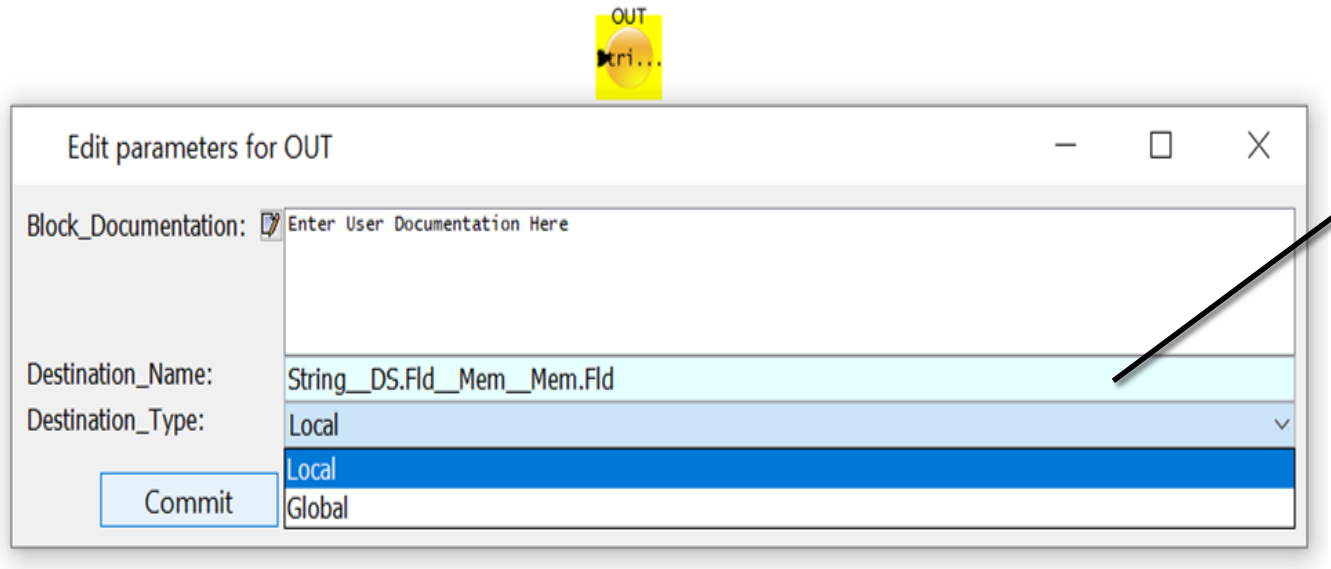
In



Destination.Value means you can send out a DS, DS field, variable or a parameter.
If the field has only Destination, send incoming value.

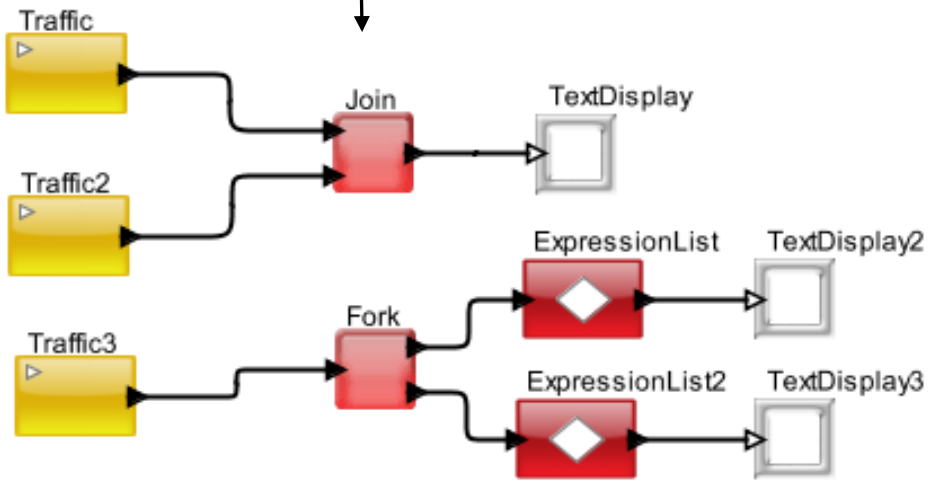
Return

Out



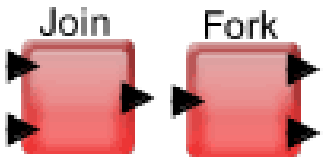
Routes the data structure to other blocks.
Destination.Value means you can send out a DS, DS field, variable or a parameter.
If the field has only Destination, send incoming value.

Join and Fork



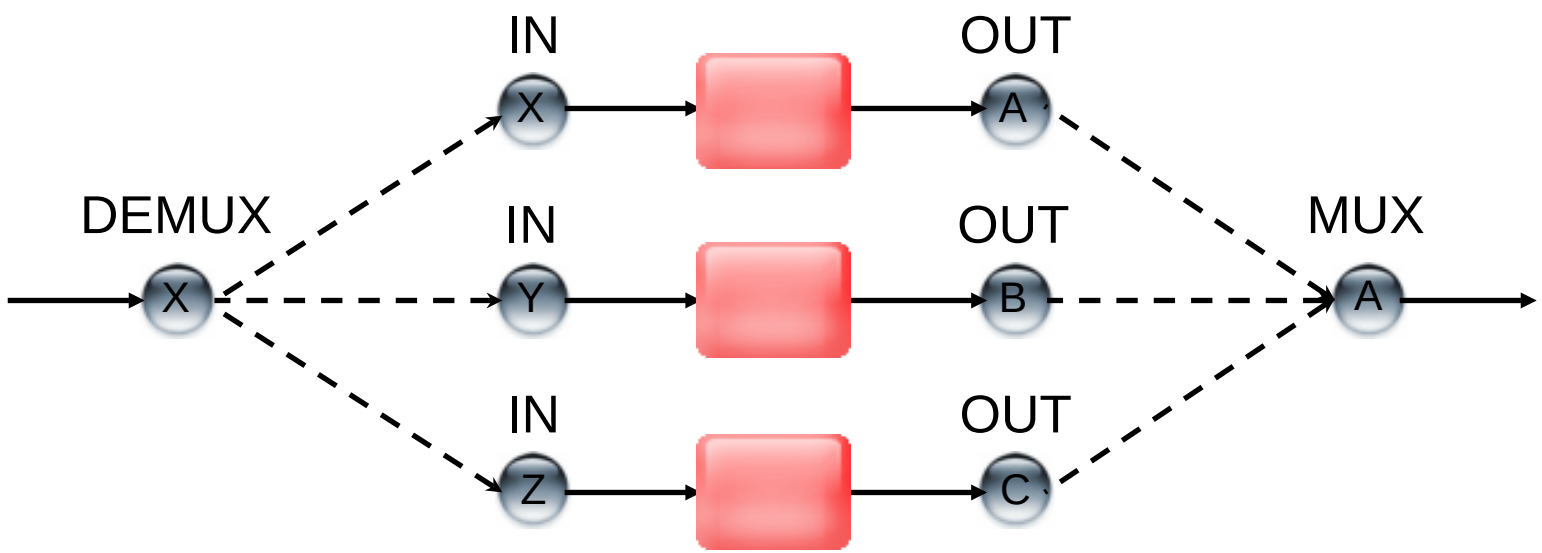
Separate inputs from any output

Control execution of two concurrent flow



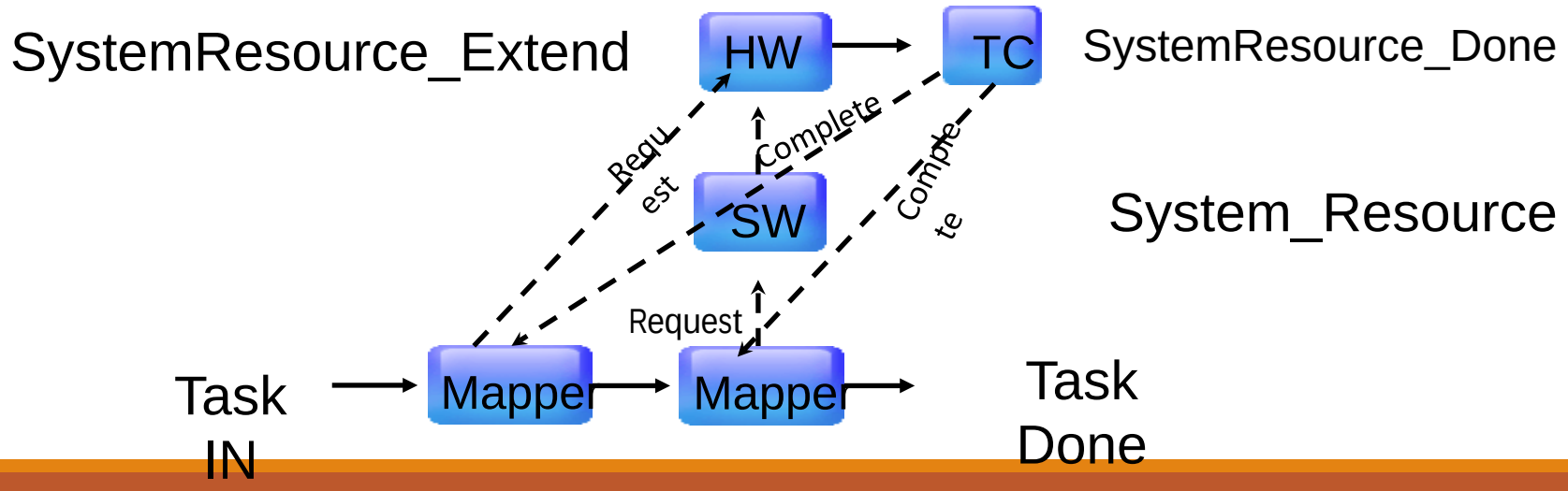
Virtual Flow Modeling

IN, OUT, MUX, DEMUX



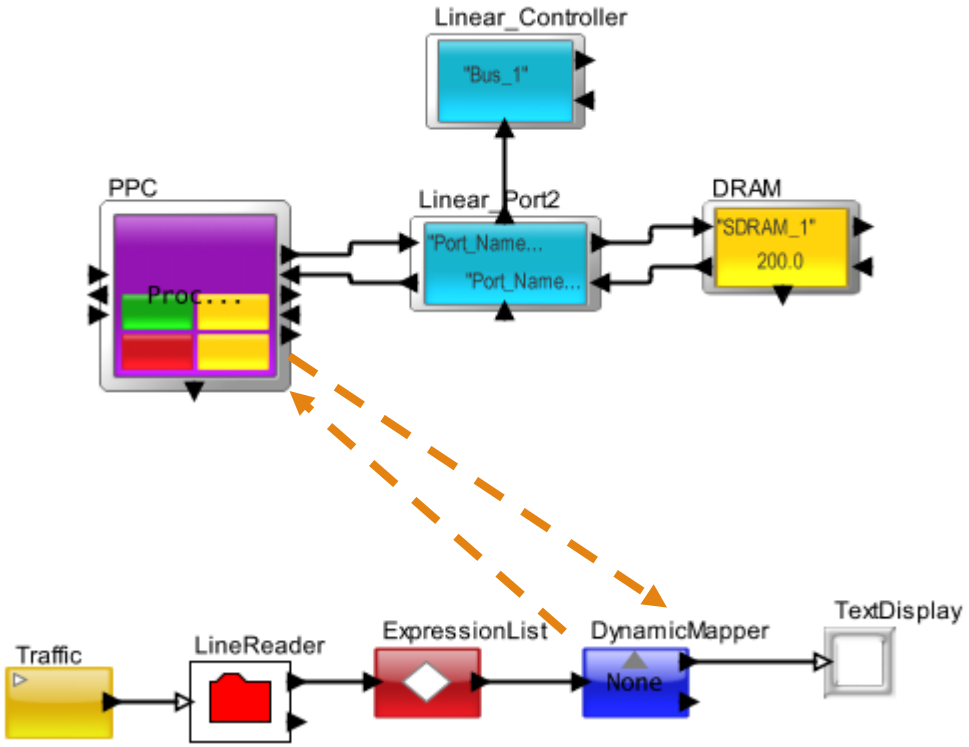
Mapper to System Resource

- Mapper blocks define the connectivity between the behavior flow and the architecture flow, and within the architecture flow using a named connection
- The block takes the incoming Data Structure and send it to the Scheduler virtually
- This block can send a request to either the SystemResource or SystemResource_Extend.



Mapper to Processor

- This block enables the mapping of behavior, task or function on a target processor, SystemResource or SystemResource_Extend.



Script To Script

- Virtual_Machine supports both local and global virtual input/output.
- The SEND function will send the values to the output ports, virtual connections, other Virtual Machine blocks or a LABEL.

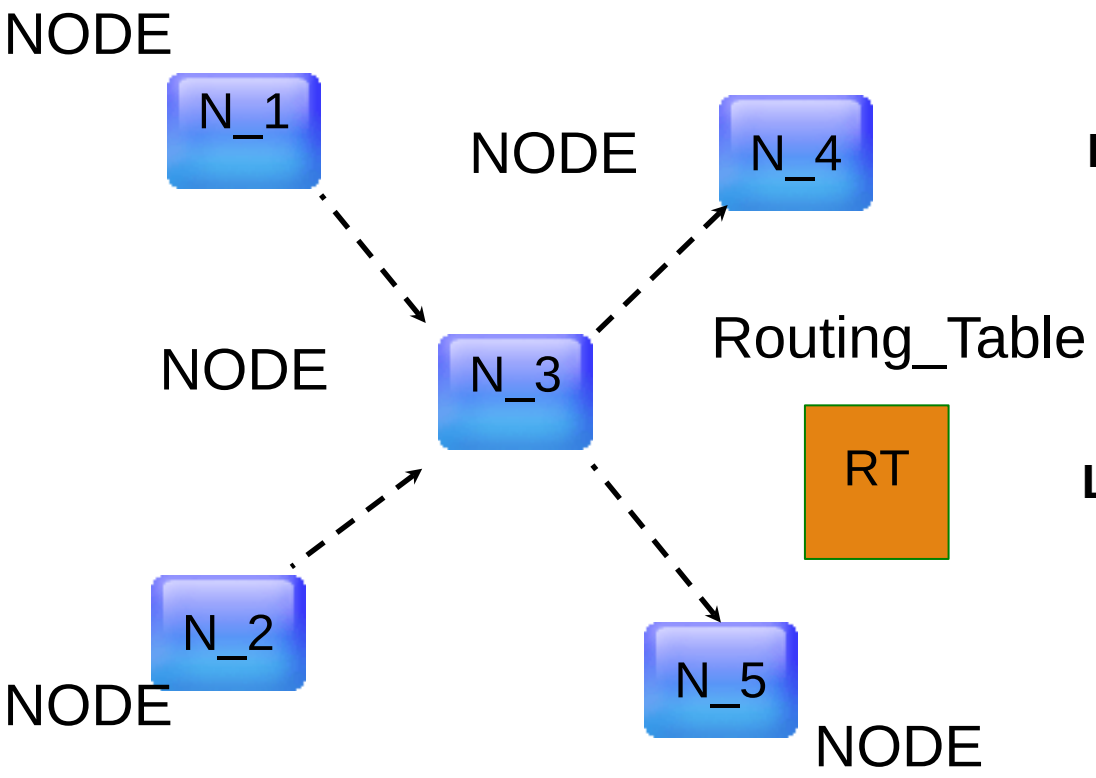
Database – Virtual Concept

- The content of the Database can be accessed as an array of data structures from any ExpressionList or Script blocks.

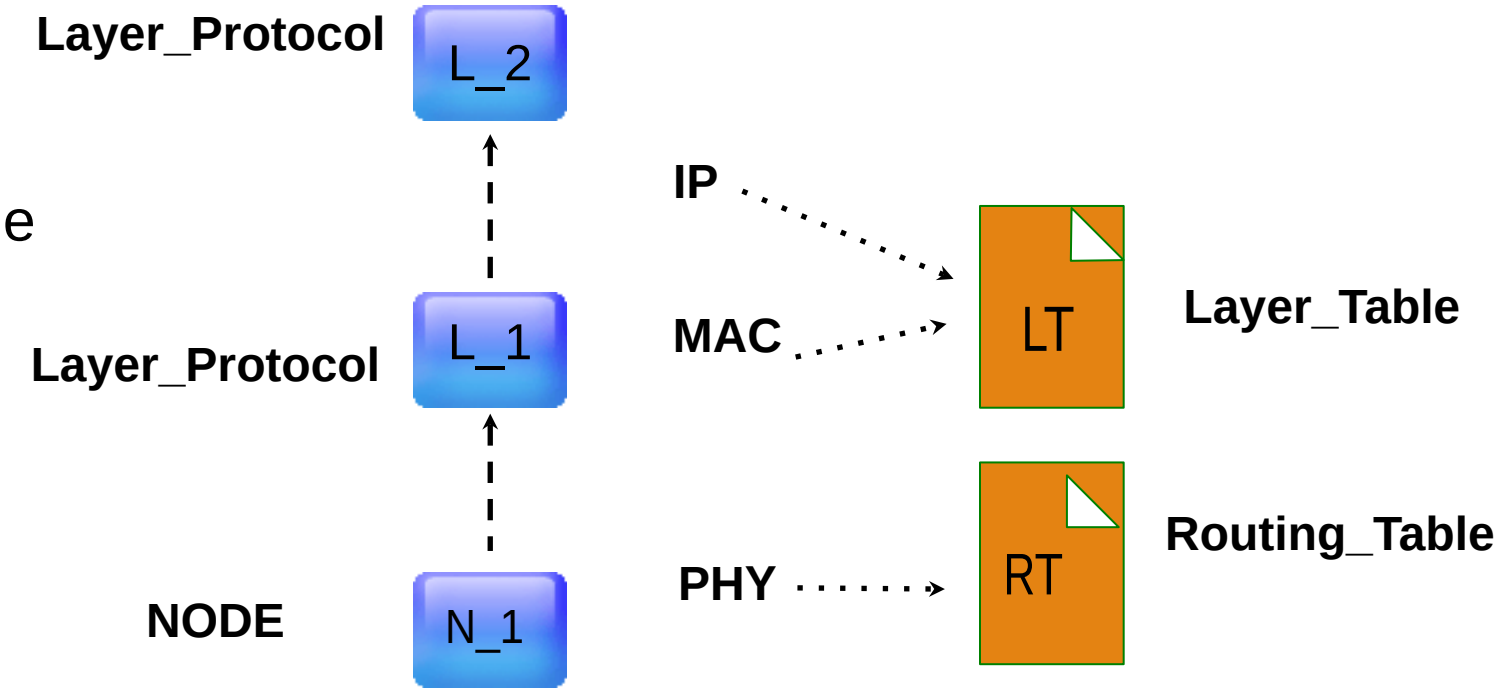
dest	source	sequence	hop
"Scheduler_1"	"Scheduler_1"	0	5
"Scheduler_1"	"Scheduler_2"	0	6
"Scheduler_2"	"Scheduler_3"	0	7
"Scheduler_3"	"Scheduler_1"	0	8

Virtual Connection Modeling

Networking: NODEs



Networking: Protocol Layer



Utility Functions

- readAllVirtual
  - ✓List of all Virtual Connections (IN, OUT, MUX)
  - ✓Names of SystemResources, Queues, Servers, script and Smart_Controller.
- Used to identify the location of a connection

RegEx

RegEx

- Collection of Mathematical, Logical, Statistical and Algorithm-Specific Functions
- Popular RegEx are used with Array, Queues and Schedulers, Data Structure, Power and Networking
- Usage
  - ✓Parameters
  - ✓Processing, ExpressionList, Script and Smart Controller for defining logic and decisions
  - ✓Can combine parameters, variable and data structure fields

Applications

- Flow Control
  - Example: Application Demo->Systems->Flow Control
- Determining status of a hardware resource before transmitting
  - RegEx function->Queues and Schedulers->getDeviceStatus
- Selecting the next available resource among a pool
  - Example: Application Demo->Automotive->Detailed SH4 Model-> Look at the Spread Spectrum Mapper.

Script to SystemResource Blocks

Block_Name: “MyVM”

input

```
done=scheduleTask(token,”MyVM”,”MySched”,time,tnow,priority,id,mutex)
if (port_name == “virtual”)
{
   <my function>
}
```

output

Block_Name: “MySched”

SystemResource or
SystemResource_Extend
Blocks

Currently, triggers completion of Scheduled Event

getBlockStatus() to Queue and Server

getBlockStatus(String block_name_, String name_, int index_)

- block_name_ = Queue_Name
- name_ = “length”, “copy”, “take”, or “stats”
- index_ = Index of Queue (one based indexing)

Queue_Name + _Length

- One can obtain the length of a Resource Queue directly from a memory.

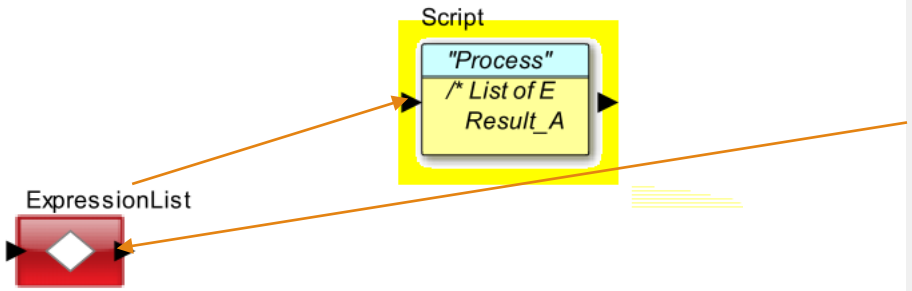
Virtual
Use in ExpressionList and Script blocks

```
virtual(input.destPort,input.Priority)
```

Send data to another block virtually

FORMAT

virtual(<Destination>, <Data>)



Expression_List:

```
/* Template to enter multiple RegEx lines*/
Result_A = virtual("Process",input)
```

newArray

Editor for Expression_List of .newArray.Script2

File Edit Help

```
/*
 * Add logic here
 */
dbName        = "DB"
taskArr       = getColumn(dbName,"Task_Name")
/*
 * calculate the length of the taskArr
 */
taskArrLength    = taskArr.length()
/*
 * create a flag array, each index for each Task
 */
taskFlagArr = newArray(taskArrLength,false)
LABEL:BEGIN
```

```
Expression    (5):   taskArr      = getColumn(dbName,"Task_Name")
  Result         :   {"GW_Entry", "Rx_Sw_Scan", "Rx_Sw"}

Expression    (9):   taskArrLength    = taskArr.length()
  Result         :   3

Expression   (13):   taskFlagArr = newArray(taskArrLength,false)
  Result         :   {false, false, false}
```

FORMAT

newArray(length, value)

Create a new array with the type being the default value

length()

FORMAT

<Array>. length( )

length() function will give us the length of the array.

Editor for Expression_List of .length.Script2

File Edit Help

```
/*
 * Add logic here
 */
dbName          = "DB"
taskArr         = getColumn(dbName,"Task_Name")
/*
 * calculate the length of the taskArr
 */
taskArrLength   = taskArr.length()
LABEL:BEGIN
```

```
Expression    (5):   taskArr         = getColumn(dbName,"Task_Name")
  Result         :   {"GW_Entry", "Rx_Sw_Scan", "Rx_Sw"}

Expression    (9):   taskArrLength   = taskArr.length()
  Result         :   3
```

Append()

Editor for Expression_List of .length.Script2

File Edit Help

```
/*
 * Add logic here
 */
idArr            = {}
LABEL:BEGIN
idArr.append(port_token.ID)
SEND(output,port_token)
```

FORMAT

<Array>. append(value)

Here, we append the incoming packet ID value to an array called idArr.
Append at the end of the array

```
Expression    (6):        idArr.append(port_token.ID)
  Result        :         {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}
```

min()

Editor for Expression_List of .min.Script

File Edit Help

```
1  /*
2   * Add logic here
3   */
4  dbName      = "DB"
5  timeArr     = getColumn(dbName,"Time")
6  minTime     = timeArr.min()
7  SEND(output,minTime)
8  LABEL:BEGIN
```

FORMAT

<Array>. min()

NOTE
Array has to be of type integer or double
Get the minimum value

```
59 Expression    (5):   timeArr      = getColumn(dbName,"Time")
60   Result         :   {1.0E-5, 6.85E-5, 2.8E-6}
61
62 Expression    (6):   minTime      = timeArr.min()
63   Result         :   2.8E-6
```

mini()

Editor for Expression_List of .mini.Script

File Edit Help

```
/*
 * Add logic here
 */
dbName      = "DB"
timeArr     = getColumn(dbName,"Time")
minIndx     = timeArr.mini()
SEND(output,minIndx)
LABEL:BEGIN
```

FORMAT

<Array>. mini()

NOTE
Array has to be of type integer or double
Gets the matched index

```
Expression    (5):   timeArr     = getColumn(dbName,"Time")
  Result        :   {1.0E-5, 6.85E-5, 2.8E-6}
                        0       1       2
Expression    (6):   minIndx     = timeArr.mini()
  Result        :   2
```

find

```
lookUpTable      = "Routing_Table_"+GW_ID
priorityTable    = "Priority_Level_Mapping_"+GW_ID
rxChannelArr     = getColumn(lookUpTable,"Source")
txChannelArr     = getColumn(lookUpTable,"Destination")
lookupPeriodArr  = getColumn(lookUpTable,"Period")
canIDArr         = getColumn(lookUpTable,"CAN_ID")
priChannelArr    = getColumn(priorityTable,"Priority_Level")
priPeriodArr     = getColumn(priorityTable,"Period")
priIDArr         = getColumn(priorityTable,"ID")            /* all column of CAN-ID range from Priority Table */
id_arr           = {}
time_arr         = {}
db_name          = "DB_"+GW_ID
task_name        = "Forwarding"
delay            = (getCell (db_name, task_name, "Time"))
id               = getCell (db_name, task_name, "ID")
target           = getCell (db_name, task_name, "Dest")
LABEL:BEGIN
port_token.Delay          = delay
port_token.ID             = id
port_token.Resource_Name     = target
id_index     = find(canIDArr,port_token.id)
if(id_index.length() == 0)  {            /* if this condition is satisfied, then that means,
                                            the incoming meessage is a non-relay message*/
   SEND(trash,port_token)
   GTO(END)
}
```

```
ch                          = 2,
dlc                         = 1,
id                          = "0x220",
overhead                    = 61,
payload_list                = "112",
sec                         = 15.2552611,
type                        = "FD",
ukey                        = 124}

*** New Thread (input) ****

Expression  (18):      port_token.Delay                = delay
  Result        :      1.1335714285714E-6

Expression  (19):      port_token.ID                   = id
  Result        :      1

Expression  (20):      port_token.Resource_Name        = target
  Result        :      "CPU_GW_1"

Expression  (21):      id_index        = find(canIDArr,port_token.id)
  Result        :      {0}
```

```
Expression    (6):       canIDArr                = getColumn(lookUpTable,"CAN_ID")
  Result        :       {"0x220", "0x100", "0x320", "0x221", "0x1C0", "0x101", "0x222", "0x321", "0x1C1", "0x223", "0x102"
```

FORMAT

find(<Array>, value)

Get index of all matched values in the array
Output is in array format

search

```
>> a = {1,3,4,5,6,6,6,6}
{1, 3, 4, 5, 6, 6, 6, 6}
>> b = a.search(6)
4
```

FORMAT

<Array>.search(value)

Get first index matching the value in an array

removeHead()

```
>> a = {1,3,4,5,6,6,6,6}
{1, 3, 4, 5, 6, 6, 6, 6}
>> b = a.removeHead
{3, 4, 5, 6, 6, 6, 6}
```

FORMAT

<Array>.removeHead()

indexOf

```
>> v = {"str1","str2","str3"}
{"str1", "str2", "str3"}
>> b = v.indexOf("str1")
{0, -1, -1}
```

FORMAT

Variable.indexOf(value)

NOTE
Variable can be a string, array of string

If String array is used, output also will be an array

substring

```
>> v = {"abc1","bcd2","cde3"}
{"abc1", "bcd2", "cde3"}
>> b = v.substring(1,2)
{"b", "c", "d"}
```

FORMAT

```
<String Variable>.substring(len1,len2)
```

Get the number of values (len1) start from position (len2)

eval

```
>> a = "0x010"
"0x010"
>> b = eval(a)
16
```

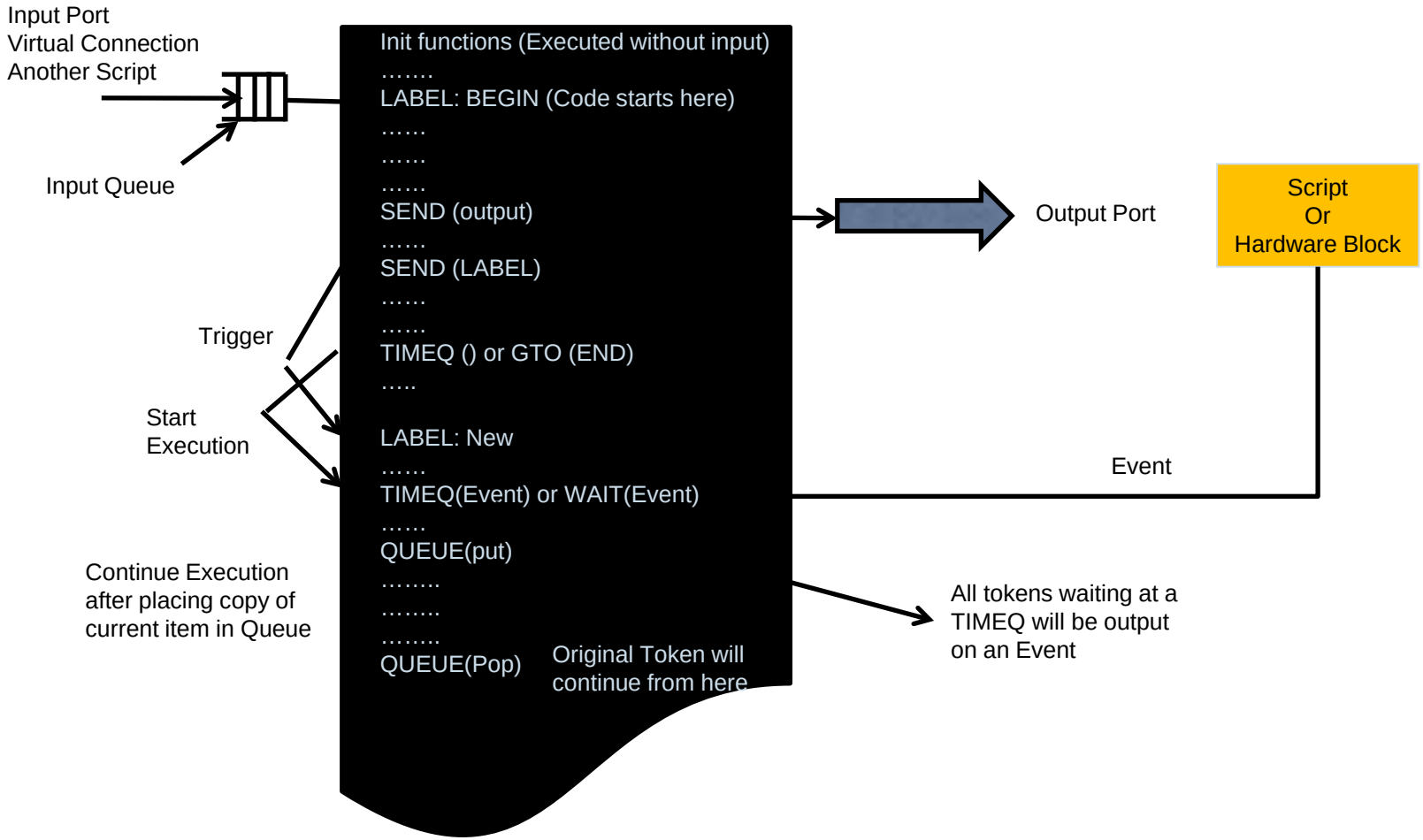
Here, by using eval we got the decimal value of the Hexadecimal equivalent

FORMAT

eval(<string>)

Script Language

Script Flow Diagram



Script Sequence of Operation

Current executing data structure is called- port_token

All code prior to LABEL:BEGIN runs at init time

Block can have multiple input and multiple output ports

When data arrives at any input ports, DS is stored in input queue in the order of arrival with portname

- No reordering for priority is done
- Virtual connections to LABELs have higher priority

Code executes similar to C

SEND can send to LABEL in same script, output port, Script block or IN

Script Sequence of Operation

When TIMEQ is reached, current DS is queued and new DS from input queue starts executing

When time changes and a DS pops out of a TIMEQ, it starts executing from the next line

When DS is stored in a Queue, it continues to execute from the next line.

When a DS is pop from the Queue, the pop DS becomes the port_token

When a WAIT is encountered, no other port_token can execute

TIMEQ and WAIT can be triggered to pop either using a Delay (provided along with the DS) or using an EVENT from any where in the model (name provided along with DS)

Script Functions

Keywords

if, if-else, else, else-if

single line if-else

While

for

LABEL

SWITCH/CASE

CALL/ RETURN

Reserved words

port_token

port_name

TNow

TStop

TResolution

local_memory

local_memory_name

Address

Methods

QUEUE

TIMEQ

SEND

PLOT

WAIT

EVENT

GTO

Script Parameters

STANDARD

- Block_Name
- Single_Cycle
- Breakpoint

HIDDEN PARAMETER

- Max_Queue_Length
- Number_of_Queues
- Add_Scheduler_Times_to_DS
- Maximum_Loops
- Read_File
- Save_Files
- Profile_File
- Listen_to_File
- Path

List of Methods

```
GTO  (3)
WAIT (time*)
SEND (port,token*)

QUEUE(Queue_Name,Token,Priority,Cmd)
 Cmd = Put,Pop,Length,Copy,Take,Stats

TIMEQ(Queue_Name,Token,Priority,Cmd,Time*)
 Cmd = Block,Non_block,Length,Copy,Take,Stats

PLOT (Plot_Name,Dest,Color,Offset,Value*)
Dest = output port or virtual connection
```

*Denotes RegEx can be used.

Grey entries used by conditional statements, red for Script only.

QUEUE, TIMEQ place result in “port_token”.

“Length” sets a block variable “length”.

If Cmd is “Set”, then Priority indicates the position.

GTO Function

```
GTO (relative address*)
GTO (LABEL*)
```

*Denotes RegEx can be used.

LABEL Function

```
LABEL: MyLabel
```

Note: MyLabel can be a fixed value or parameter of the model.
It cannot be a Variable.

WAIT Function

```
WAIT (time*)
WAIT(Event Name)
```

*Denotes RegEx can be used.

WAIT is a blocking wait, does not allow other inputs to execute.

Event is triggered from anywhere in the model.

Can be used to trigger at a particular time or dependency

If value is integer or long, then aligns to Clock boundary, else delay

SEND Function

```
SEND (port, token*)
SEND (virtual_connection, token*)
SEND (LABEL, token*)
```

*Denotes RegEx can be used.
SEND to LABEL creates a new thread

QUEUE Function

QUEUE (name, token, priority, command)
Addition functions are:

QUEUE (“MyQueueName”, “length”) ->Gets the length
QUEUE (“MyQueueName”, “stats”) -> Generates stats
QUEUE (“MyQueueName”, “get”)
QUEUE (“MyQueueName”, “pop”) ->removes head of Queue
QUEUE (“MyQueueName”, “copy”) ->get first item in Queue.
QUEUE (“MyQueueName”, “put”)

Single dimension queue and length is determined by the Hidden parameter
Length and stats require a variable on the LHS
Get and pop will write to port_token
Values for name, token, priority, command supports DS.field notation.
RegEx expressions for the arguments are not supported.

TIMEQ Function

TIMEQ (name,token,priority,delay)
Eg: TIMEQ (“MyTimeqName”, port_token, 0, 1.0)

Additional Functions:
TIMEQ (“MyTimeqName”, “length”)
TIMEQ (“MyTimeqName”, “stats”)
TIMEQ (“MyTimeqName”, “copy”)

name, token, priority, delay supports DS.field notation.
RegEx expressions for the arguments are not supported.

PLOT Function

```
PLOT (name,dest,color,offset,value)
```

name, dest, color, offset, value supports DS.field notation.
RegEx expressions for the arguments are not supported

If-Else Conditional Statements

```
if      (port_name == “input”)
{
   SEND (output, port_token)
}
else if (port_name == “input2”)
{
   SEND (output2,port_token)
}
else
{
   SEND (reject,port_token)
}
```

port_name and port_token are Script/Smart_Controller identifiers for port name source + active token.

While Conditional Statement

```
while (MyFlag)
{
   if (Queue_Length > 0)
   {
      MyFlag = false
      SEND (output, port_token)
   }
}
```

MyFlag is an instance variable defined in self_start.
Queue_Length is a memory location being set outside this while loop.
If Queue_Length > 0, MyFlag will be set false, SEND to output and drop out of while loop.

For Conditional Statement

```
for(idx=0;idx<10;idx=idx+1)
{
    port_token.length=irand(idx,20)
    SEND(output,port_token)
}
```

The for loop runs until idx gets the value 10. For each run a random integer value between idx and 20 is generated and is assigned to a Data structure field called length. Then we use SEND() function to send the data structure out.

SWITCH, CASE Statement

```
SWITCH (port_name)
{
   CASE: input
      SEND (output, port_token)
   GTO (END)
   CASE: DEFAULT
      SEND (output2, port_token)
   GTO (END)
}
```

SWITCH and CASE are spelt as all caps
Last CASE needs to be DEFAULT.
Case can flow through to next case if no GTO()

CALL, RETURN Statements

```
<Normal Flow>
CALL (MySubr)
SEND (output, port_token)
GTO (END)

LABEL: MySubr
   port_token.INDEX = 0
   port_token.TIME  = TNow
RETURN
```

CALL, RETURN all caps.
RETURN statement should have the same port_token when
CALL was executed, else will not RETURN correctly.
CALL will stop the current flow to execute this subroutine
When done it will return to the next line
Port_token must be of type data structure, else a error will be reported.

How to Read or Write a DS Field or Variable

Basic: Field or variable name fixed

input

```
port_token.MyField = MyVariable + port_token.MyField
WAIT(1.0)
SEND(output, Token)
```

output

Advanced: Calculate field or variable name

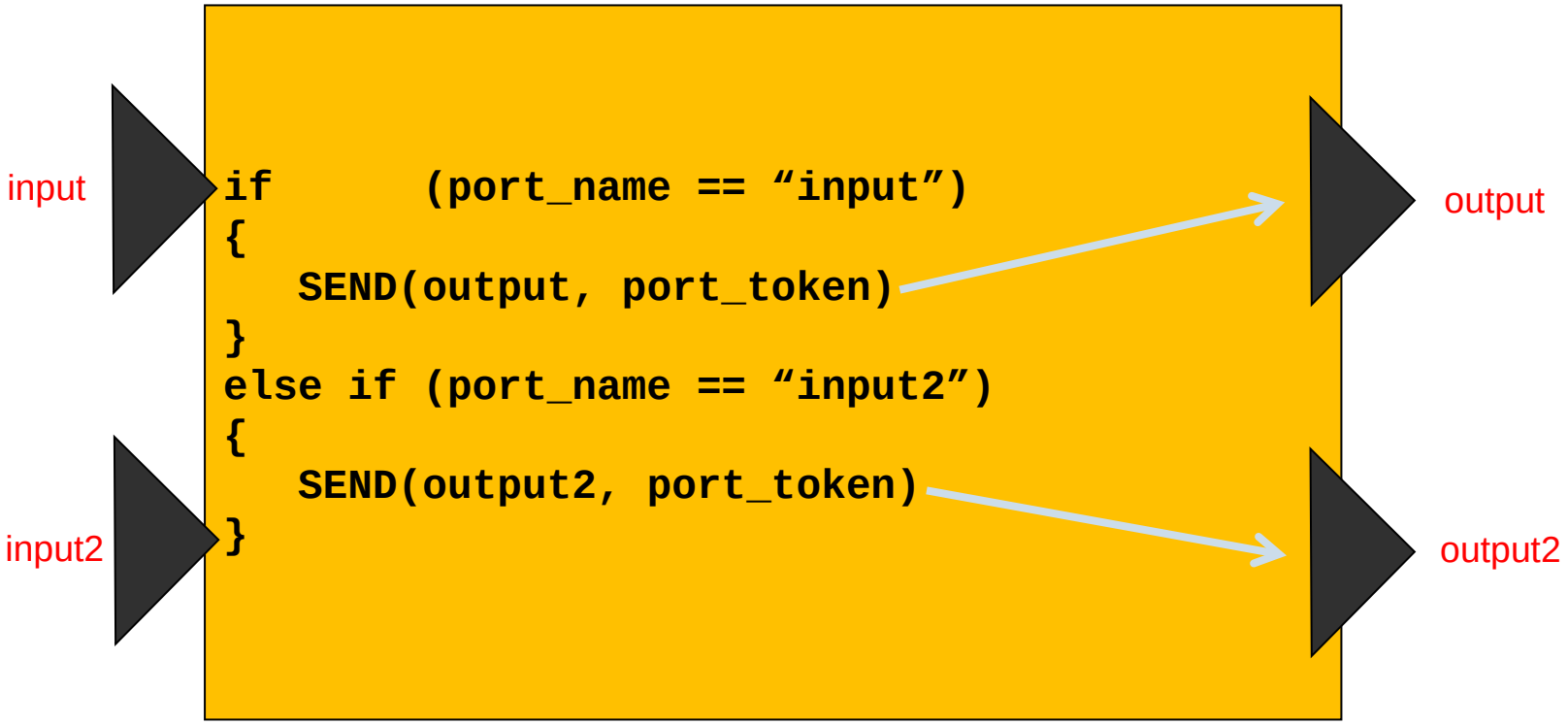
input

```
MyFieldToken = readField(port_token,”MyField”)
MyToken = readMemory(”MyVariable”)+ MyFieldToken
MyToken = writeMemory(”MyVariable”, MyToken)
WAIT(1.0)
SEND(output, Token)
```

output

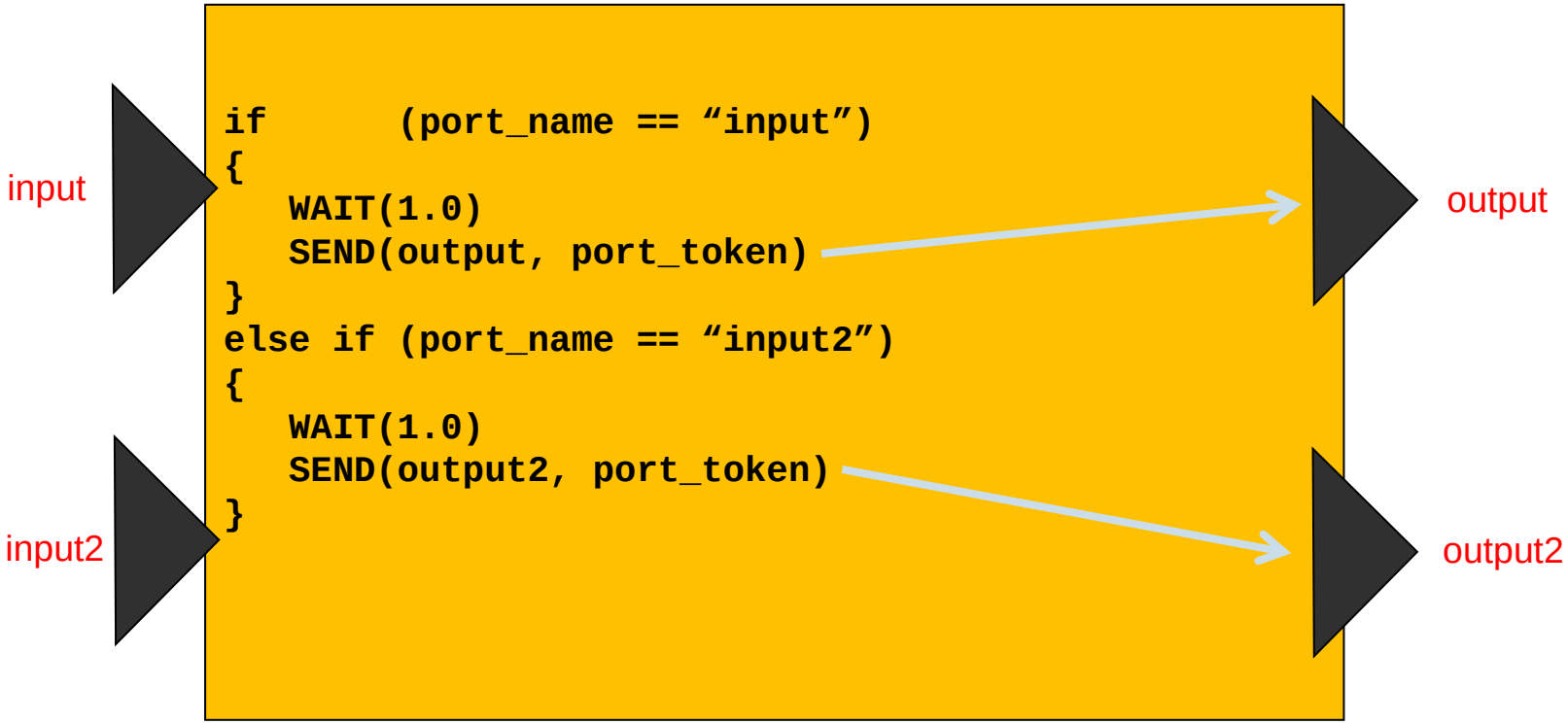
Variable is a **global** variable.

How to use “port_name” and“port_token”



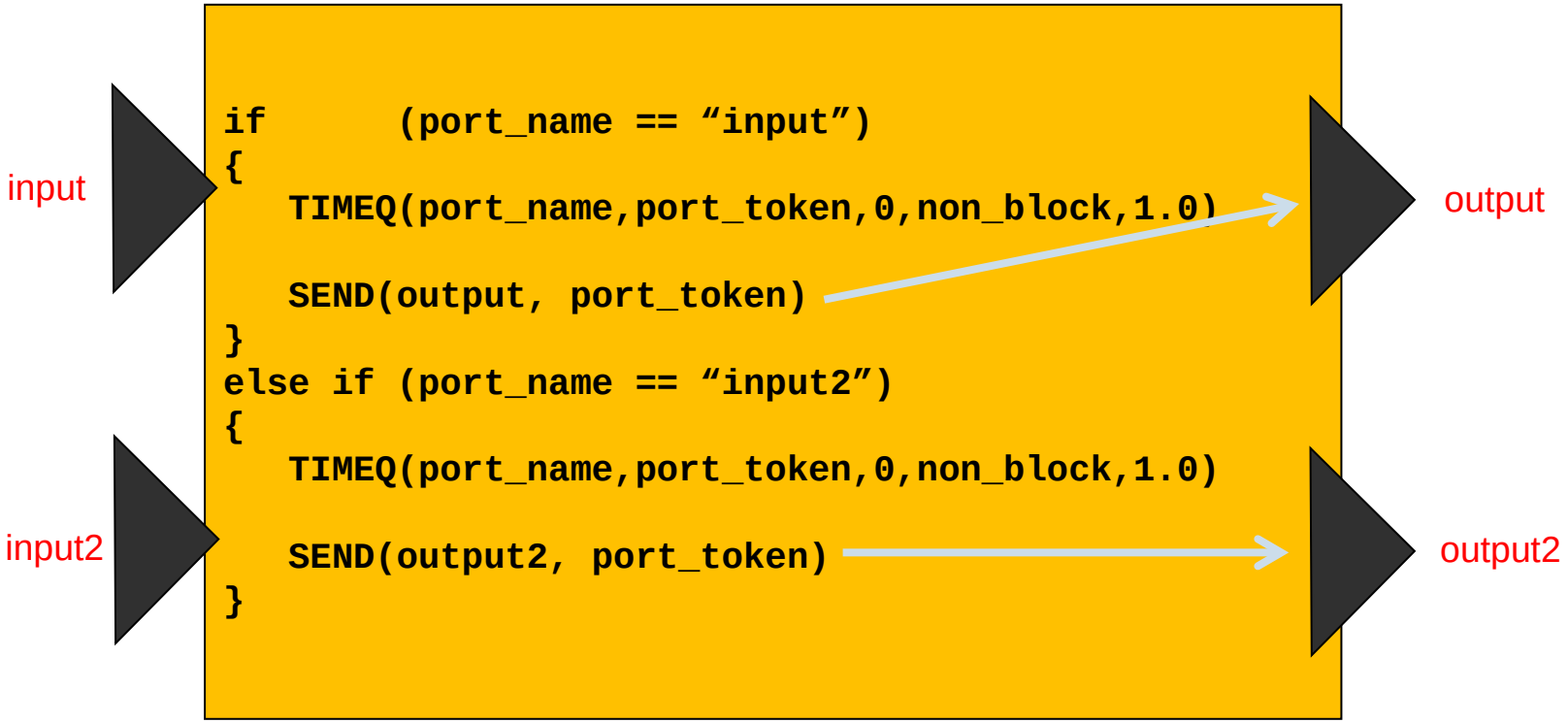
port_name == “virtual” for virtual input, can use field of data structure for further selection.

WAIT as a blocking Switch

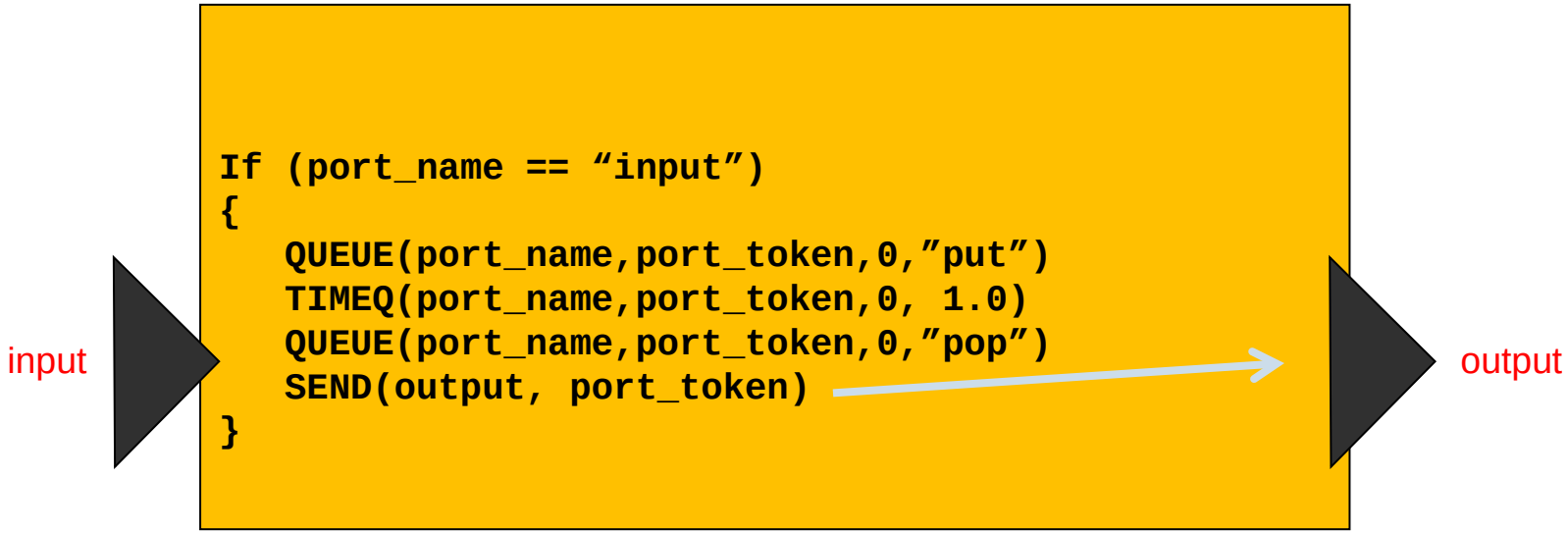


Script or Smart_Controller has a common queue for input Transactions, and allows blocking operation if two Transactions arrive at the same time.

TIMEQ as a non-blocking Switch



How to use QUEUE




```
If (port_name == “input”)
{
   QUEUE(port_name,port_token,0,”put”)
   TIMEQ(port_name,port_token,0, 1.0)
   QUEUE(port_name,port_token,0,”pop”)
   SEND(output, port_token)
}
```

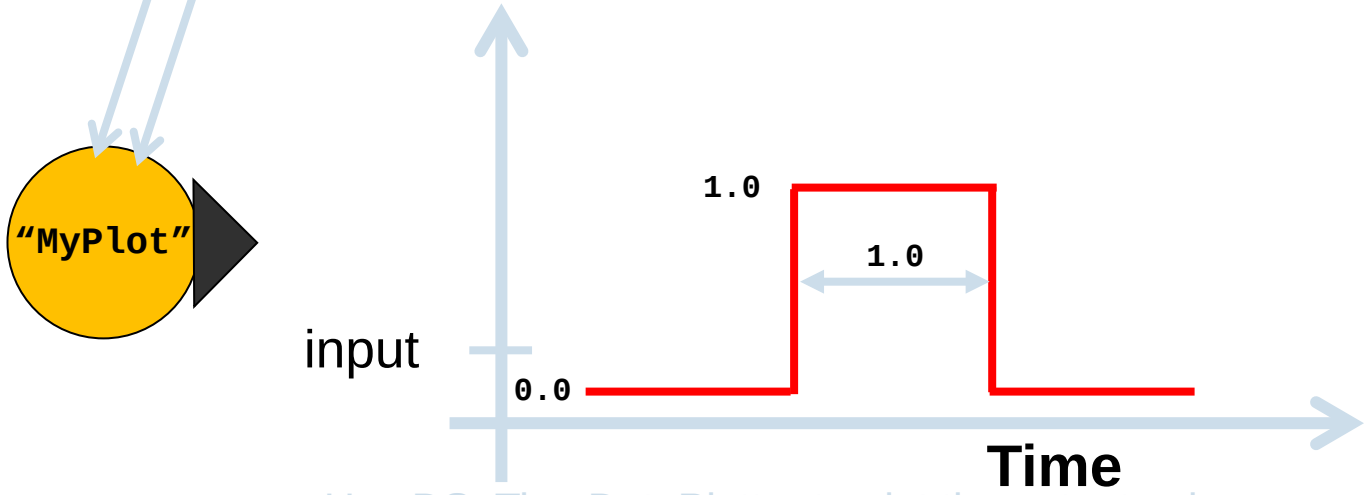
The above QUEUE will keep statistics for Transactions that arrive while the delay executes. This models a simple delay, yet keeps track of active Transactions. TIMEQ is also sufficient.

Script (only) How to PLOT a Transaction

input

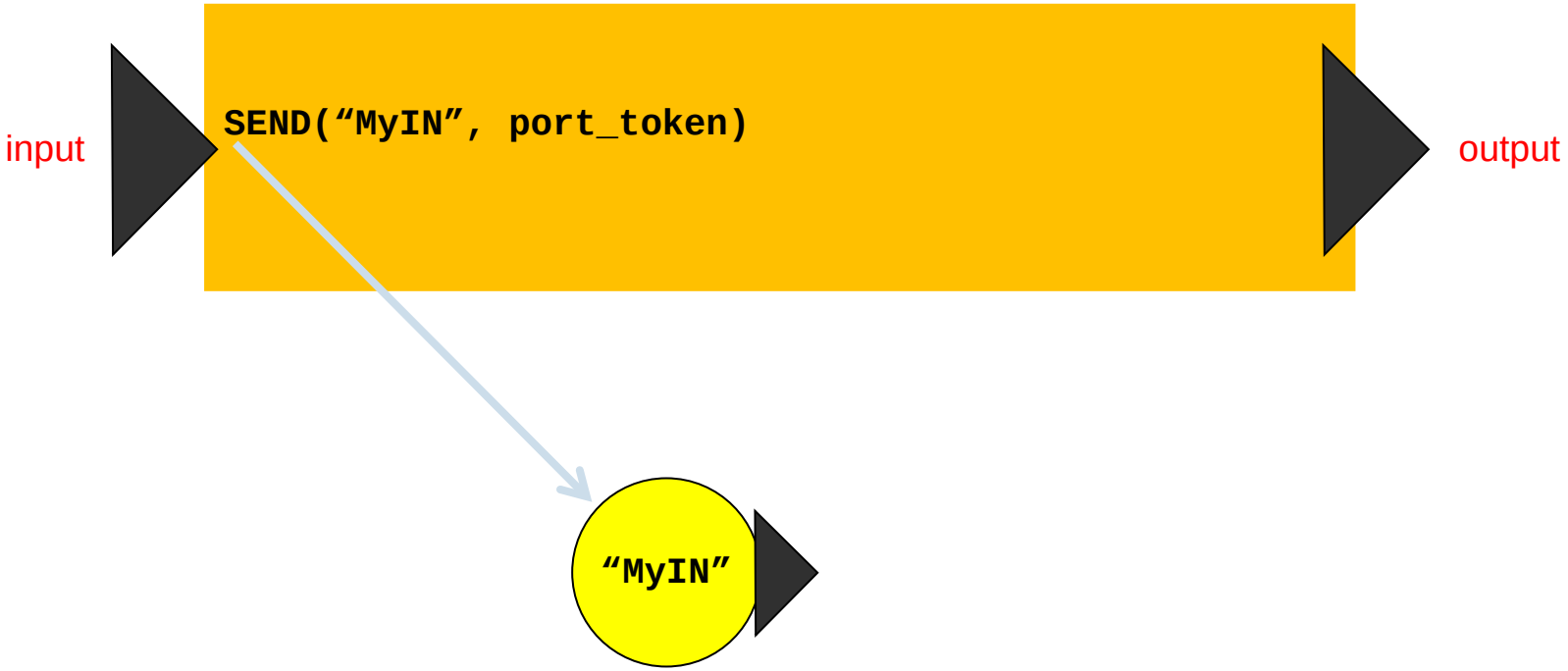
```
if (port_name == “input”)
{
   PLOT(port_name,”MyPlot”,”red”,0,1.0)
   WAIT(1.0)
   SEND(output, port_token)
   PLOT(port_name,”MyPlot”,”red”,0,0.0)
}
```

output



Use DS_TimeDataPlotter to plot the output values

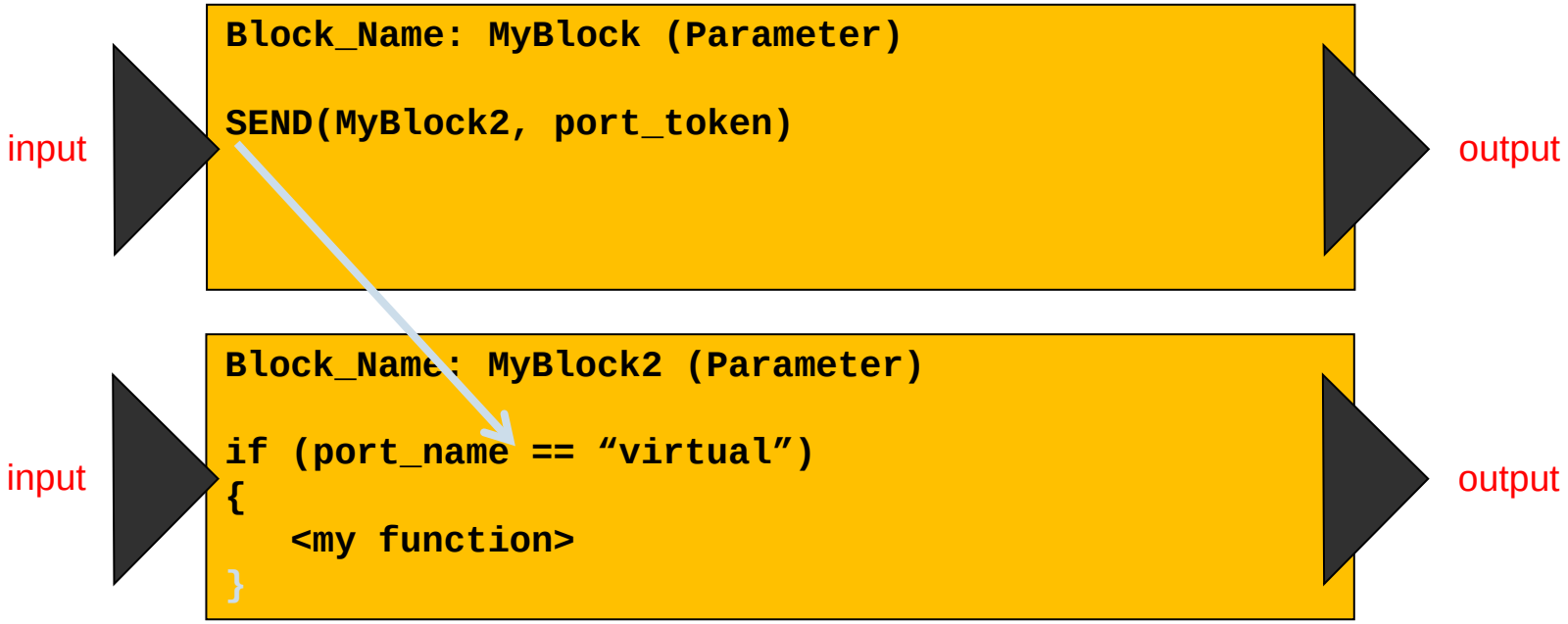
Script, SC to Virtual Connection Block



Note: SC shorthand for Smart_Controller.

IN or DEMUX Virtual Connection Blocks

Script, SC to Script, SC Communications



Note: SC shorthand for Smart_Controller.
Virtual input gets higher priority in the input queue

“virtual” is keyword, DS can contain further information

Script to SystemResource Blocks

Block_Name: “MyVM”

input

```
done=scheduleTask(token,”MyVM”,”MySched”,time,tnow,priority,id,mutex)
if (port_name == “virtual”)
{
   <my function>
}
```

output

Block_Name: “MySched”

SystemResource or
SystemResource_Extend
Blocks

Currently, triggers completion of Scheduled Event

RegEx: getBlockStatus() to Queue

getBlockStatus(String block_name_, String name_, int index_)

- block_name_ = Queue_Name
- name_ = “length”, “copy”, “take”, or “stats”
- index_ = Index of Queue (one based indexing)

Queue_Name + _Length

- One can obtain the length of a Resource Queue directly from a memory.

Queue, Smart_Controller Example

input

Queue

```
Max_Queue_Length: Param
Number_of_Queues: Param
Initial_Queue_State: “First_Token_Enqueue”
Queue_Reject_Mechanism: “Incoming_Token_Rejected”
Queue_Type: “FIFO”
```

output

pop

input

```
while (true) {
   Select              = 1
   WAIT  (1.0E-05)
   while (Select       <= NumQs && Deficit > 0) {
      if (getBlockStatus(Smart_Resource_Name,"length",Select) > 0) {
         token          = getBlockStatus(Smart_Resource_Name,"copy",Select)
         WAIT ((token.Size * 1.0E-06) / Controller_Speed_Mhz)
         SEND (pop,Select)
         Index          = Select - 1
         Thru(Index)    = Thru(Index) + token.Size
         Deficit        = Deficit     - token.Size
         if (Deficit   <= 0 || (TNow  - Const_Time) > Period) {
            Const_Time  = TNow
            Deficit     = 128
         }
      }
      Select            = Select + 1
   }
}
```

output

Script - Traffic Generator Example

input

```
LABEL: BEGIN
   Token                = newToken("Processor_DS")
   Token.ID             = 1
   WAIT (Start_Time)
   Token.TIME           = TNow
   SEND (output,Token)    /* Send out first DS, Start_Time */
   while (true)
   {
      WAIT(Mean_Time)
      Token.TIME        = TNow
      Token.ID          = Token.ID + 1
      SEND(output,Token) /* Send out mean time DSs        */
   }
```

output

Note: Assume Start_Time and Mean_Time are the parameters of the block. Double click the block, add name and value, where value could be a window level parameter.

Script, SC- DS input, Set Fields, DS output

input

```
Count                    = 0
LABEL: BEGIN
If (port_token.containsRecordTokenLabel("MyField"))
{
   Count                 = Count + 1
   port_token.MyField = Count
   port_token.TIME       = TNow
   SEND(output,Token)
}
else
{
   throwMyException("No Field Named MyField.")
}
```

output

Note: Assumes MyField is an existing field with an integer type.

Script, SC- Search Array Example

input

```
MyArr                    = {“one”,”two”,”three”}
LABEL: BEGIN
   Search_Name           = port_token.MyName
   Search_Result         = search(MyArr,Search_Name,0)
   if (Search_Result != -1)
   {
      SEND(output,Token)
   }
   else
   {
      throwMyException(“No Search_Name: ” + Search_Name)
   }
```

output

Note: search (Array_Name, Matching_Value, Starting_Index) will find the first match or return -1.

search (Array_Name, Matching_Value) will return an array of matching indexes, if none found will return {} (empty array with length == 0).

Script, SC- How to obtain Statistics of Queues

```
Time     = TStop
TIMEQ (“MyStats”, port_token, 0, non_block, TStop)
Stats_DS = getBlockStatus(Block_Name,"Input_Queue","stats")
SEND (stats_out,Stats_DS)
LABEL: BEGIN
 TIMEQ (“MyInput”, port_token, 0, non_block, 1.0)
 SEND(output,port_token)



/* Note: If the block has a WAIT, when the getBlockStatus
         executes, then the WAIT can block the stats
         from processing.  In this case, it may be
         better to have a separate block collect the
         statistics, using the same getBlockStatus()
         function.  */
```

input

output

stats_out

Note : This block obtains the statistics of the “Input_Queue” at the end of the simulation (TStop) and sends to the port named stats_out. QUEUE and TIMEQ named queues would be accessed in the same manner.

Script, SC- Statistics Output

```
DISPLAY AT TIME   ------ 20.00000000000 sec ------
{BLOCK                    = "Stats_Example.Input_Queue",
DELTA                     = 0.0,
DS_NAME                   = "Queue_Common_Stats",
ID                  = 1,
INDEX                     = 0,
Number_Entered            = 20,
Number_Exited             = 19,
Number_Rejected           = 0,
Occupancy_Max             = 1.0,
Occupancy_Mean            = 0.5128205128205,
Occupancy_Min             = 0.0,
Occupancy_StDev           = 0.4998356074261,
Queue_Number              = 1,
TIME                = 20.0,
Total_Delay_Max           = 0.0,
Total_Delay_Mean          = 0.0,
Total_Delay_Min           = 0.0,
Total_Delay_StDev         = 0.0,
Utilization_Mean          = 0.0}
```

Script, SC- How to Debug

input

```
Debug      = false
LABEL: BEGIN
port_token = port_token  /* Use “Listen to Block” */
Debug ? sendToCommandLine(“Incoming DS: “ + port_token.toString()) : 0
SEND (output, port_token)
```

```
/* sendToCommandLine() can send specific information
   to the command line.  Another approach is to set
   the memory equal to itself and “listen to block”
   to see the execution.  The script itself may want
   to have a “debug” mode to see DSs entering or
   exiting the block, listing only key fields.  */
```

output

Note: One can also listen to port to see DS’s entering or exiting.
Use the Single Cycle and Breakpoint options

Resources

Resources

- Consume time or quantity
- Can be distributed, shared or dedicated
- Timing: cycle or event
- Level of details
  - ✓**Abstract**: Delays, Quantity and Buffering (Focus of this Chapter)
  - ✓**Detailed**: Processor, Memory and RTOS

Resources- Definition

- Resource is an element required by an entity
  - ✓ Channels
    - Zchannel blocks
  - ✓ Queuing Resource
    - Queues
  - ✓ Quantity-Based Resource
    - Resource_QS_Allocate
  - ✓ Time-Based Resource
    - Server
  - ✓ Scheduler Resource
    - SystemResource
    - SystemResource_Extend with SystemResource_Done

Selection of Resource

To hold a data structure until it receives the permission to transmit

- Queue

To model a delay or a single processing stage in a flow

- Server

To define multiple flows or applications and link them to a single processing unit

- SystemResource

To create a complete system definition of bus, cache and processor with multiple application flows

- SystemResource_Extend

To define interfaces, networks or wires

- Channel

To define a quantity that can be broken down into ether collection or indexed

- Quantity_Shared

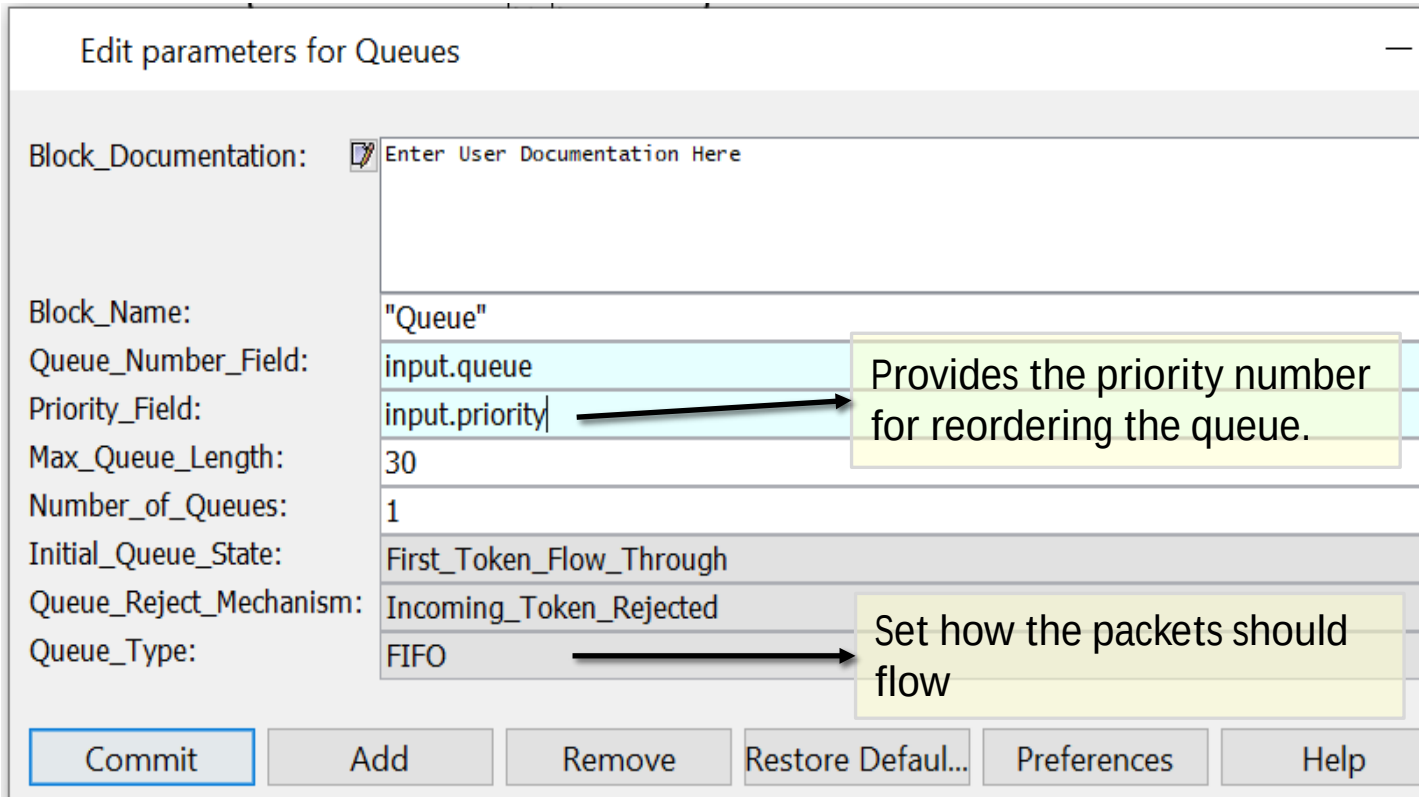
If you want data to be sent out in a particular order at different times

- Pipeline

To model single queue with multiple processing resources that picks the next available

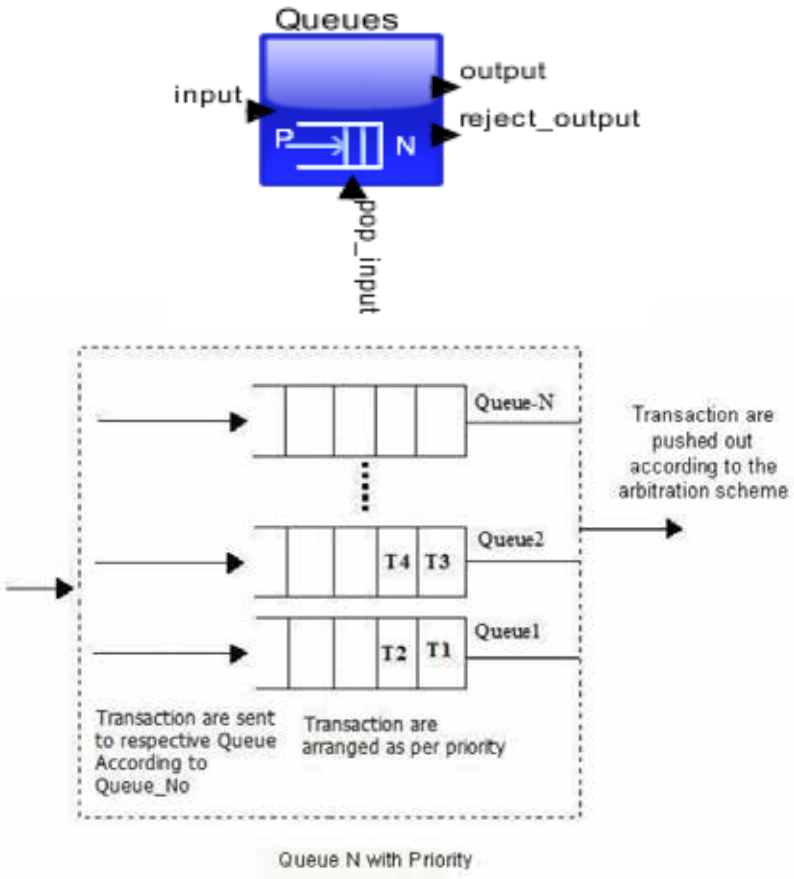
- Use Server_N

Queue



Provides the priority number for reordering the queue.

Set how the packets should flow



Queue Operation

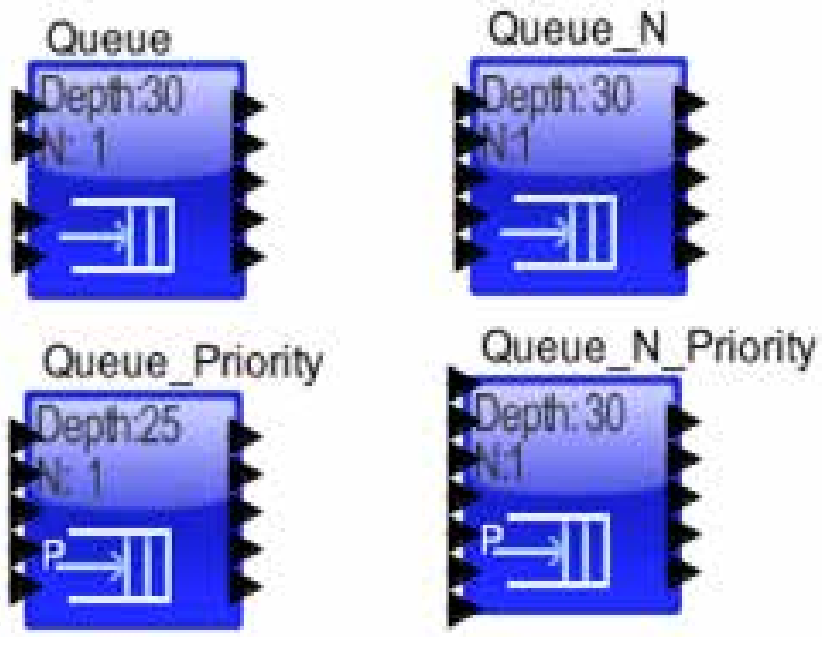
- Data Structures are queued based on priority from high to low number
- Data Structures in the queue are arranged based on FIFO or LIFO setting
- Number_of_Queues defines the number of parallel queues contained by a single Queue block
- Queue Number Field selects the queue to place
- To pop a packet
  - From the head of a queue, Queue_Number must be sent to pop_input port.
  - Any position in the Queue, {Queue_Number, position) must be sent to the pop_input port
- When Maximum_Queue_Length is reached, packets are Rejected based on Rejection_Mechanism and sent to Reject_output
- Based on initial Queue State parameter,
  - Enqueue: First Transaction can be enqueued and wait for the pop
  - First_Packet_Flow_Through: First transaction send without pop. After first packet, head of queue sent if prior was acknowledged with pop

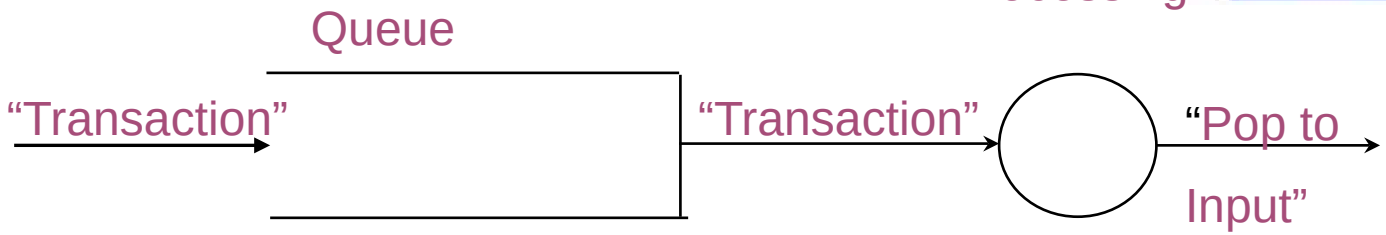
EventQueue (Deprecated)

Queue can be associated with dimension (Queue_N) and priority (Queue_Priority)

- ✓ Priority determines queue reordering and reject mechanism
- ✓ Dimension specifies number of parallel queues within this block

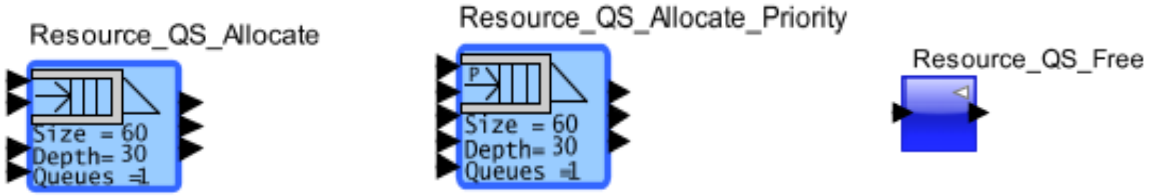
Requires a pop to send DS on the output port
Delay not predictable in advance
Statistics generation is provided on lowest ports

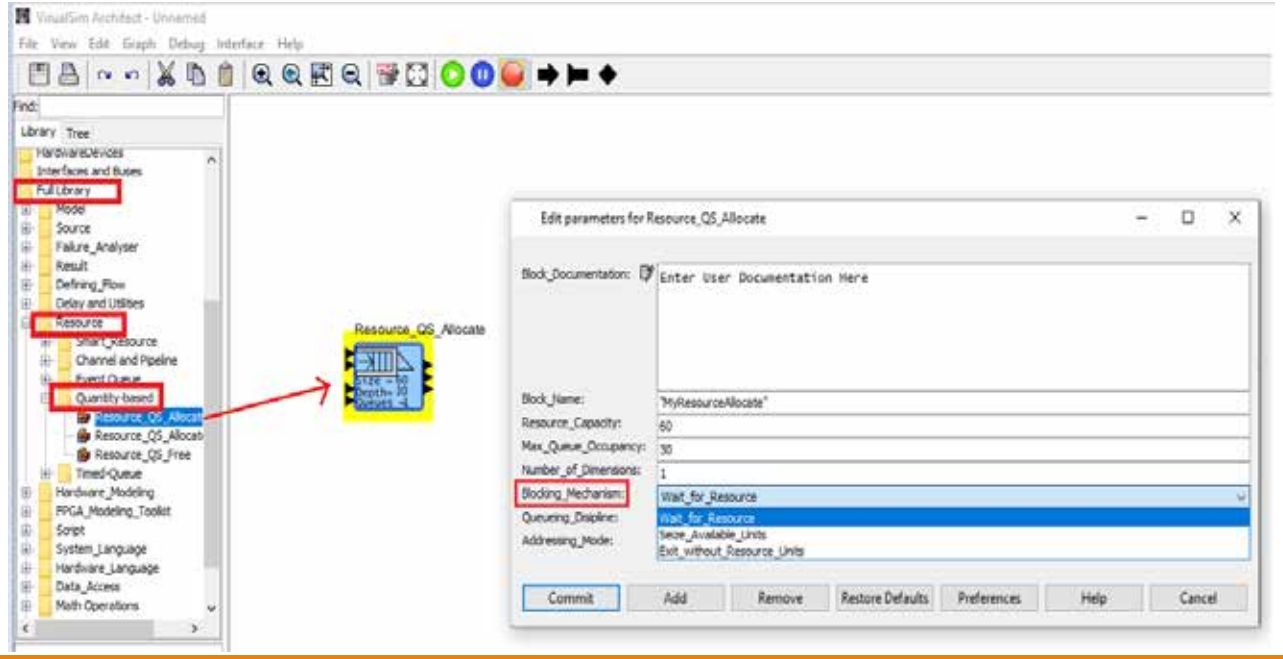




Quantity Based Resources

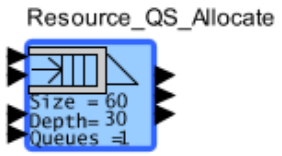
- Passive Resources
- Resource units represent a quantity of items that must be possessed before a transaction (DS) can continue
- **Location : Full Library -> Resources - > Quantity-Based**
- Queues requests if sufficient quantity of resource not available
- Can index quantity and select contiguous or distributed
- Queueing Discipline - Enqueues input as FIFO or LIFO



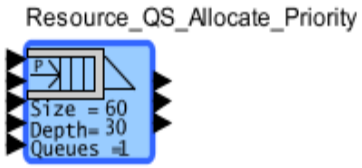


Types of Quantity Based Resources

- Resource_QS_Allocate
  - Consumes quantity of resources
  - Contains a multiple queue with no reordering



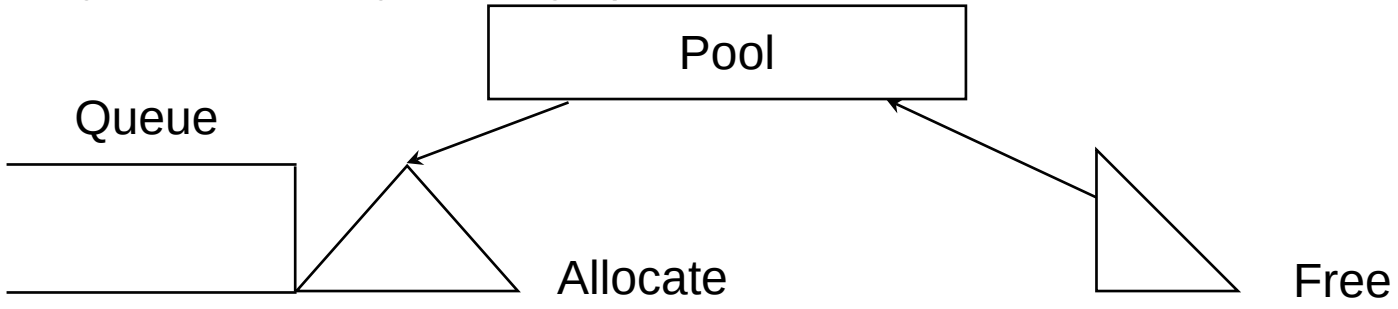
- Resource_QS_Allocate_Priority
  - Consumes quantity of resources
  - Multiple queues with reordering



- Resource_QS_Free
  - Free the resources allocated by Resource_QS_Allocate

Allocate and Free

- Allocate
  - ✓ This is where the transaction attributes are set
  - ✓ The input values specify transaction attributes
  - ✓ If all inputs are enabled, requests for *n* units of resource are made
  - ✓ When resource units granted to a transaction, outputs are enabled
- Free
  - ✓ Resource units are held for arbitrary time (DS delayed in model)
  - ✓ They can be freed by enabling inputs of Free block



Examples

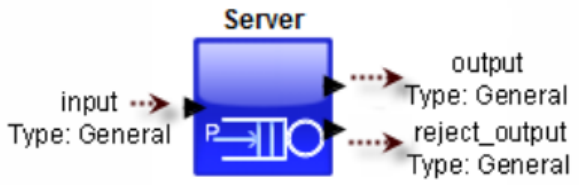
- Pages of memory in a computer
  - ✓ A process requests x pages, waits if it can't get them immediately
  - ✓ Once granted the requested number of pages, the process holds the memory an indeterminate amount of time
  - ✓ At a later time, the process returns the memory to a “pool”, where free pages are stored
  - ✓ Upon being freed, pages may be allocated to other waiting processes
- Bus arbitration in hardware
  - ✓ Pool initially consists of one token
  - ✓ Each component which requires the use of the shared bus must request a token
  - ✓ Once token is obtained, mutual exclusive access to the bus is assured
  - ✓ Bus is freed when token is placed back into the resource

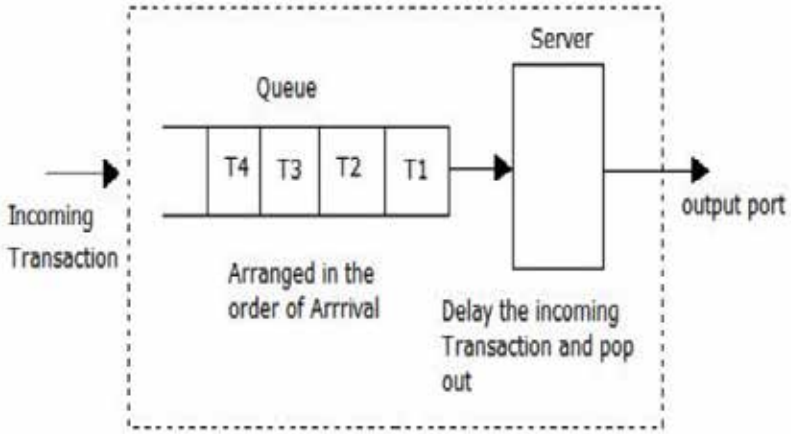
Addressing mode

- Non-indexed resource units--default
  - ✓ Token Pool
  - ✓ Indistinguishable items
- Indexed resource units
  - ✓ Distinguishable items
    - ✓ Request contiguous block of resource units. Examples:
      - Pages of memory
      - Virtual circuit numbers in a network
  - ✓ Integer address (1st = 0)

Server

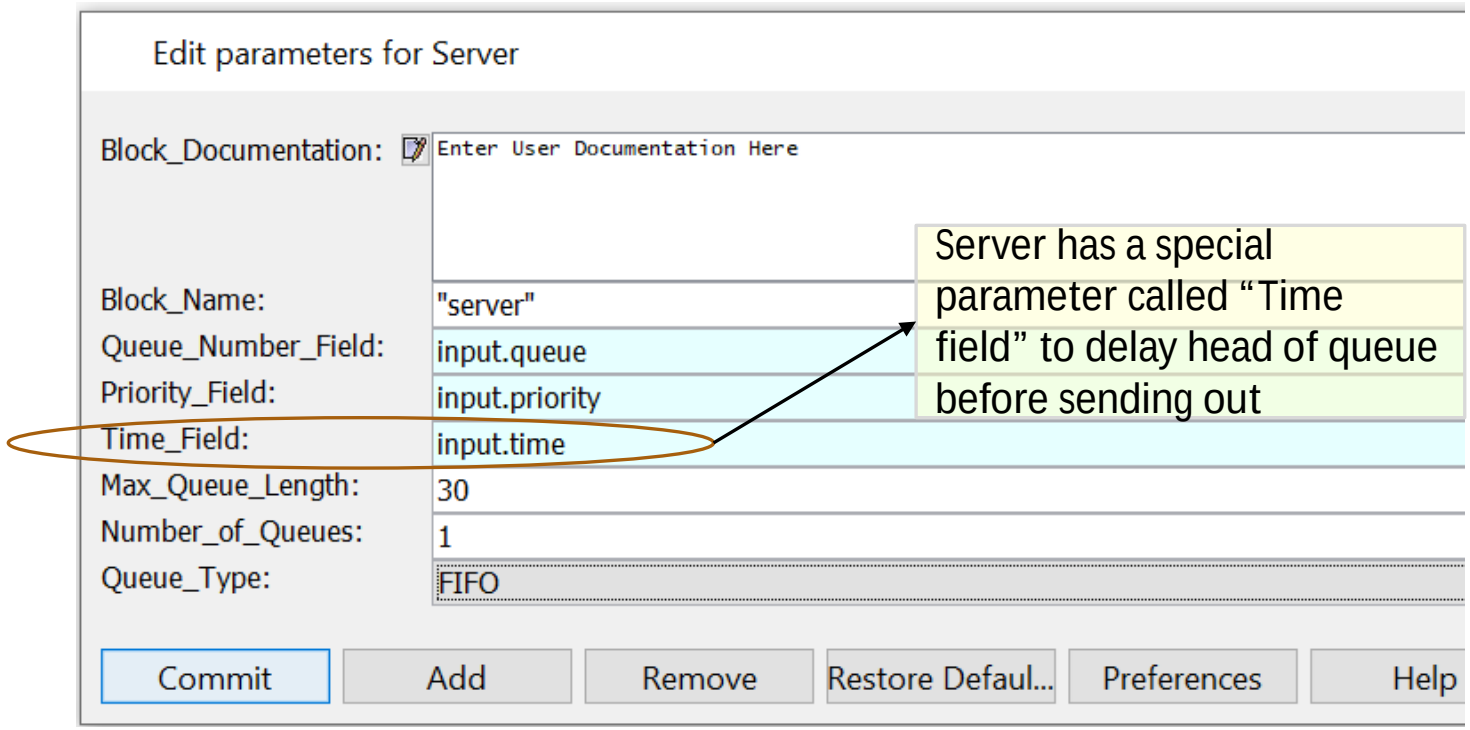
- Define multiple queues + time delay
  - Active Resource
  - DataStructures queued in FIFO or LIFO order
- Processing time is known in advance
  - Provided along with the transaction to this block.
- SLOT
  - Special operation mechanism
  - Models any slot-based architecture such as multiple virtual RTOS, TDMA etc.





Basic Timed Queue

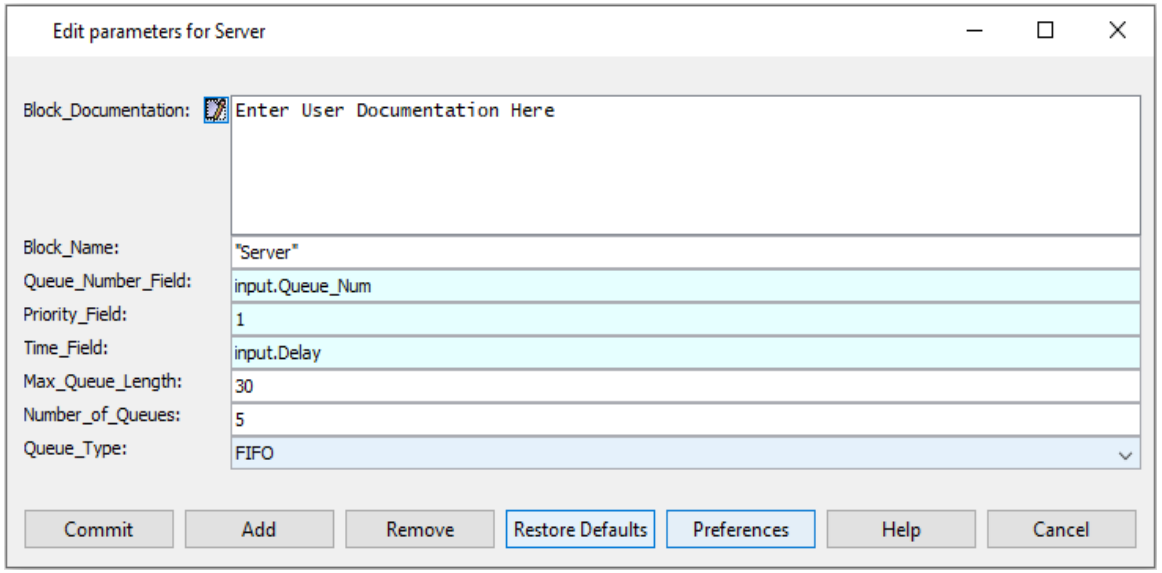
Server



Server has a special parameter called "Time field" to delay head of queue before sending out

Operation

- *Queue_Number_Field* selects the queue
- Queue is reordered based on *Priority field*
- Queue data in FIFO or LIFO based on *Queue_Type*
- Delayed by *Time_Field* value at head of queue and sent out
- DS sent to *reject_output* when Max_Queue_Length reached

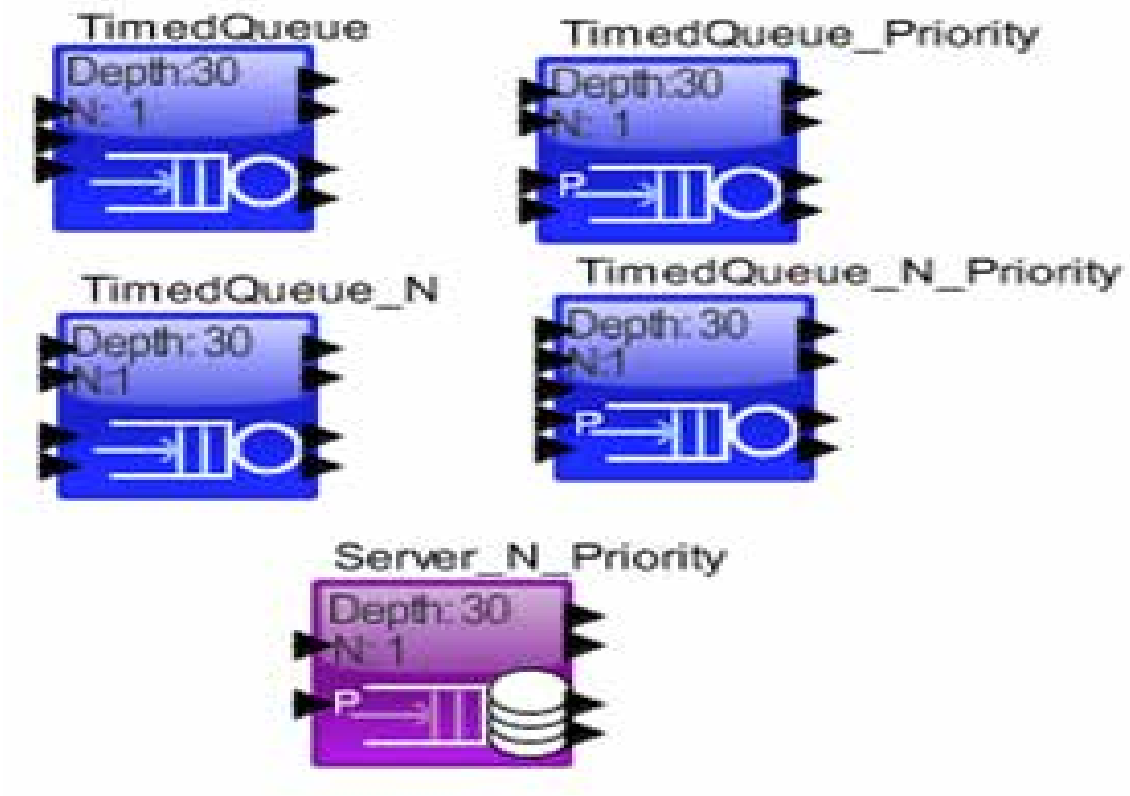


Time- based Resources

1 Queue- 1 Resource

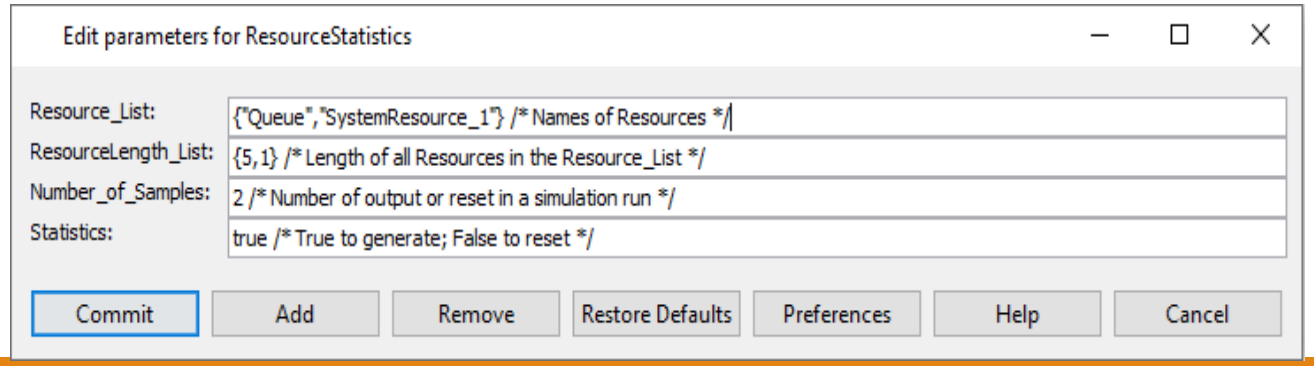
Multiple instances
1 Queue- 1 Resource

1 Queue- |
Multiple Resource

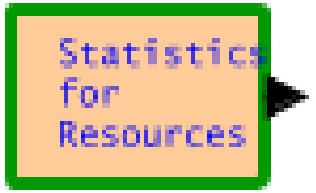


Resource Statistics

- Generated statistics using
  - ✓ Resource Statistics
  - ✓ RegEx Function- getBlockStatus
  - ✓ Array Lookup- Queue(Name, length)
- ResourceStatistics bloc
  - Supports System_Resource, System_Resource_Extend, Queue, Server, channel and Quantity Shared Blocks
    - ✓ Buffer Occupancy, delay, utilization, Number of Transactions entered, exited, rejected
- Array Lookup
  - ✓ eg: *Length_A = Queue_Length(1)* -> This gets the length of Queue Number 1



ResourceStatistics




```
DISPLAY AT TIME                    ------ 100.00000000000 sec --
{BLOCK                           = "Resource_Statistics.Queue",
DELTA                            = 0.0,
DS_NAME                          = "Queue_Common_Stats",
ID                               = 6,
INDEX                            = 0,
Number_Entered                   = 199,
Number_Exited                    = 12,
Number_Rejected                  = 157,
Occupancy_Max                    = 30.0,
Occupancy_Mean                   = 20.0754716981132,
Occupancy_Min                    = 0.0,
Occupancy_StDev                  = 10.1402412546265,
Queue_Number                     = 1,
TIME                             = 100.0,
Total_Delay_Max                  = 91.0182813545,
Total_Delay_Mean                 = 26.2911000907667,
Total_Delay_Min                  = 0.0,
Total_Delay_StDev                = 23.8057259325954,
Utilization_Mean                 = 0.0}
```

Resource Statistics

Statistics Name	Value	Explanation
Number_Entered	100	Number of transactions entering the queue
Number_Exited	25	Number of transactions that left the queue
Number_Rejected	10	Number of transactions rejected and output to reject port
Queue_Number	1	Queue Number. Queue number start at 1.
Occupancy_Min	4.0	Minimum queue size during the simulation
Occupancy_Mean	8.0	Mean/Average queue size during the simulation
Occupancy_StDev	3.0	Standard Deviation from the Mean queue size during the simulation
Occupancy_Max	25	Maximum queue size consumed during the simulation
Total_Delay_Min	1.3	In seconds. Least time through the queue+server among all transactions
Total_Delay_Mean	1.3	In seconds. Mean/Average time through the queue+server among all transactions
Total_Delay_StDev	1.3	In seconds. Standard Deviation from the Mean time through the queue+server among all transactions
Total_Delay_Max	1.3	In seconds. Maximum time through the queue+server among all transactions
Utilization_Mean	10.0	Mean utilization of the server portion only. Queue utilization not considered

Statistics & Debugging

Applies to Queue, Server, Server_N, SystemResource and Channel
Generate using:

- ResourceStatistics block
- Name + "_" + Length(queue_index))
- RegEx

```
DISPLAY AT TIME                    ------ 110.00100 us ---
{A_Bytes                        = 268,
A_Field                         = 5.0E-5,
A_Priority                      = 1,
A_Queue                         = 1,
BLOCK                           = "Traffic",
DELTA                           = 0.0,
DS_NAME                         = "Header_Only",
EndTime                         = 1.0,
ID                              = 2,
INDEX                           = 0,
TIME                            = 6.0E-5,
Task_Latency                    = 5.0001E-5,
Throughput_Array                = {5.359892802144E6},
Time_Array                      = {6.0E-5, 1.10001E-4},
Trace_Array                     = {"Bus_in", "Bus_out"}}
```

```
DISPLAY AT TIME                    ------ 50.00000000000 sec ---
{{BLOCK                         = "Resource_Statistics.Server"
DELTA                           = 0.0,
DS_NAME                         = "Queue_Common_Stats",
ID                              = 1,
INDEX                           = 0,
Number_Entered                  = 28,
Number_Exited                   = 27,
Number_Rejected                 = 0,
Occupancy_Max                   = 3.0,
Occupancy_Mean                  = 0.8363636363636,
Occupancy_Min                   = 0.0,
Occupancy_StDev                 = 0.7327069701676,
Queue_Number                    = 1,
TIME                            = 50.0,
Total_Delay_Max                 = 2.5,
Total_Delay_Mean                = 1.2301235472111,
Total_Delay_Min                 = 1.1,
Total_Delay_StDev               = 0.3281892988381,
Utilization_Mean                = 62.9309355286132}}
```

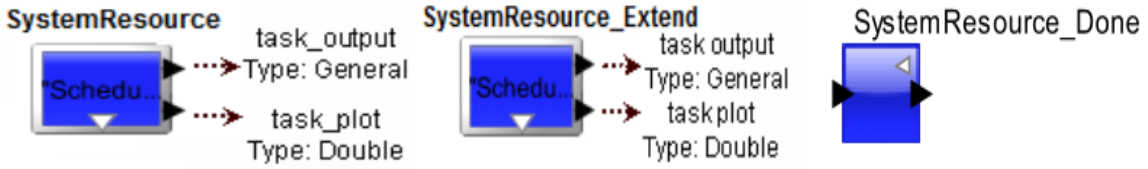
Concept of System Resource

Concept
- Split operation into two parts
- Behavior or mapper
- Resource (similar to Server)

Blocks
- Behavior: Mapper, SoftwareMapper, DynamicMapper
- Architecture: SystemResource_Extend, SystemResource
- Notify: SystemResource_Done

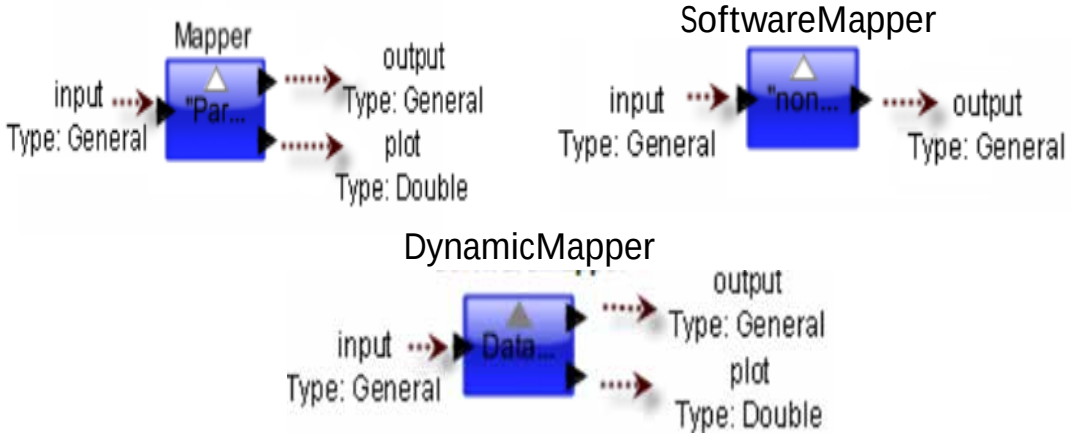
Multiple concurrent requests
- Send from Mapper (Behavior) to the SystemResource with the delay information
- Can be static or dynamic reference
- Scheduler: First Come-First Serve, Round-Robin, Preemption, Non-Locking

SystemResource_Done block
- Release appropriate SystemResource_Extend block by signaling the completion of an external task

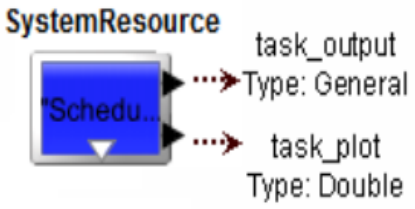


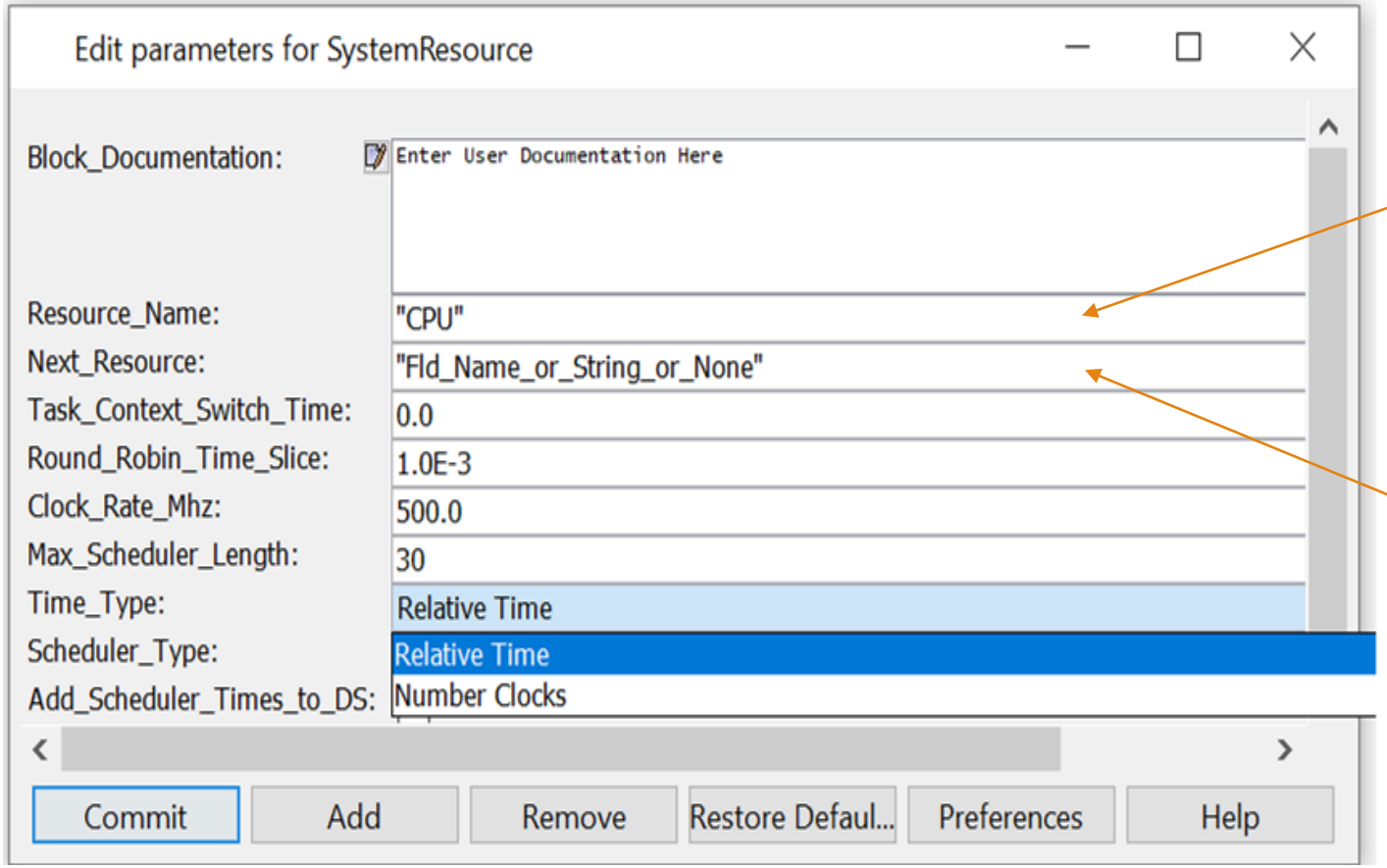
Architecture

Behavior



System Resource

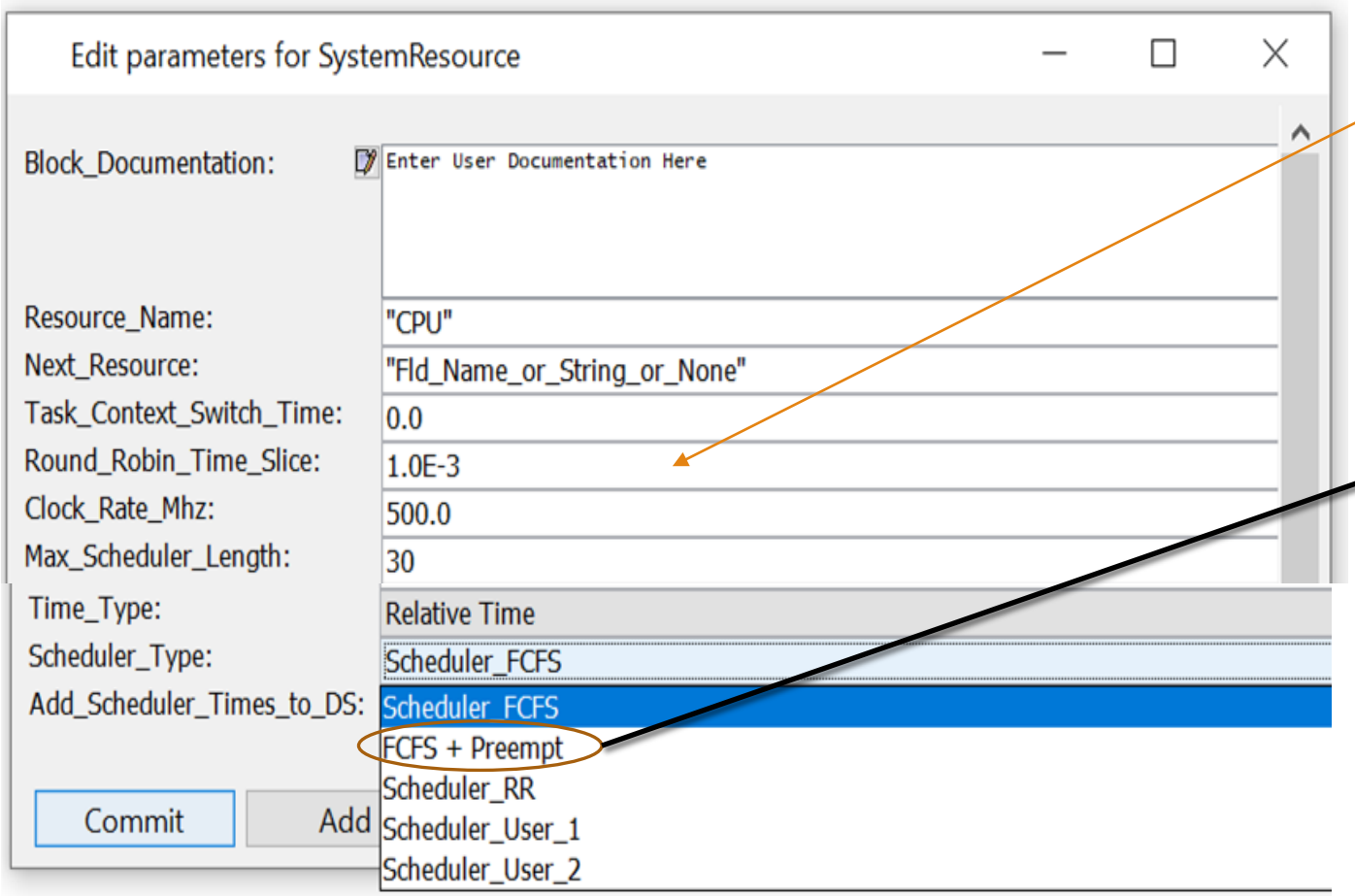




This is the name of this SystemResource block and is used by Mappers, RegEx and other SystemResource block to call this block to execute a transaction.

Next_Resource is the name of the next hierarchical System Resource, which can be SystemResource or SystemResource_Extend block.

System Resource (Cont.)



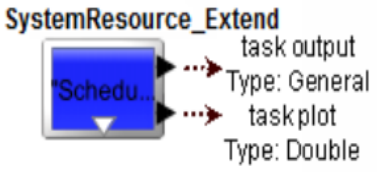
Time the scheduler will devote to each task for Round Robin

Application comparisons

Features	SystemResource	SystemResource_Extend
Preemption	Yes	No
Hierarchical	Yes	No
Extended Task Processing	No	Yes
Non-Blocking	No	Yes

Set Scheduler type from the range of schedulers

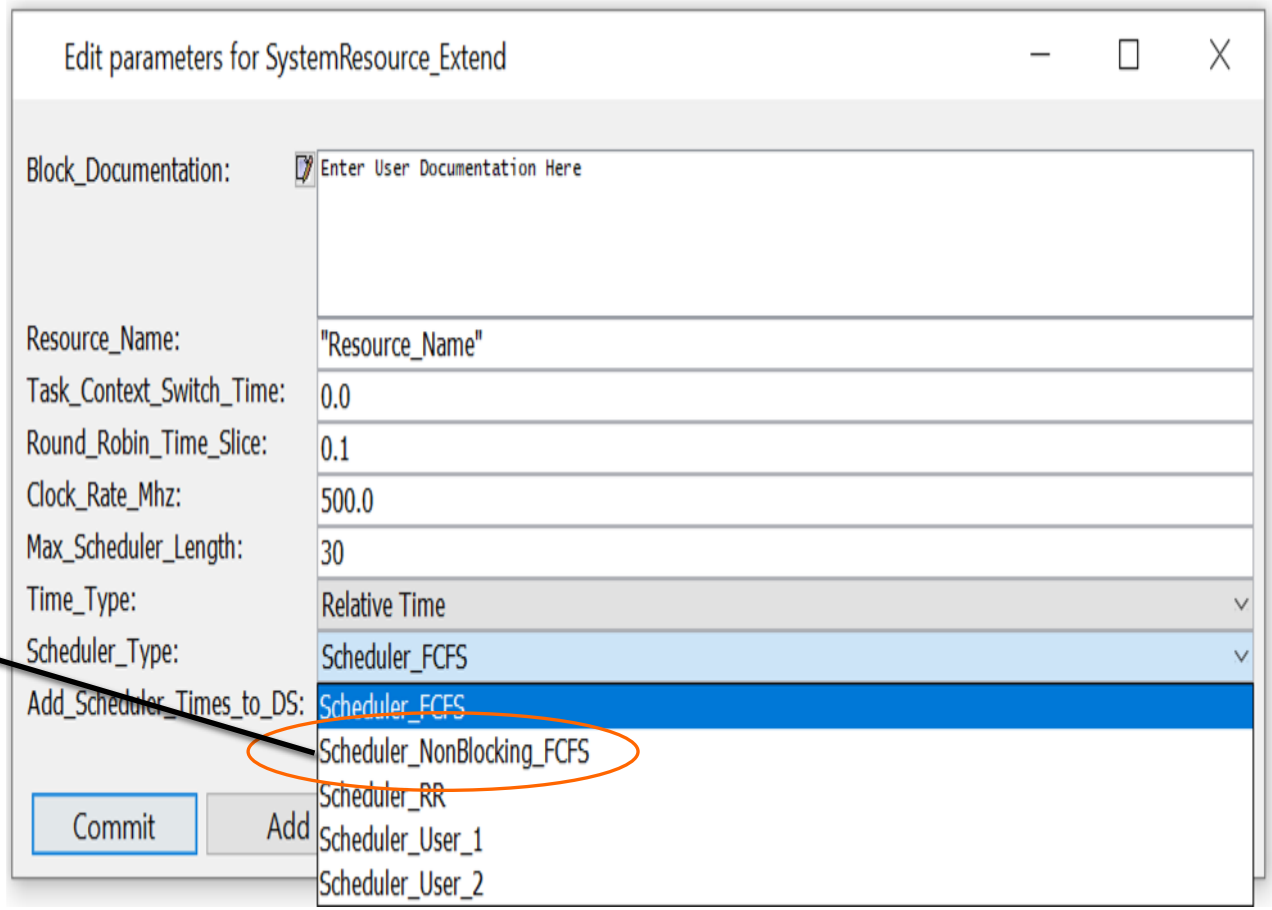
System Resource Extended



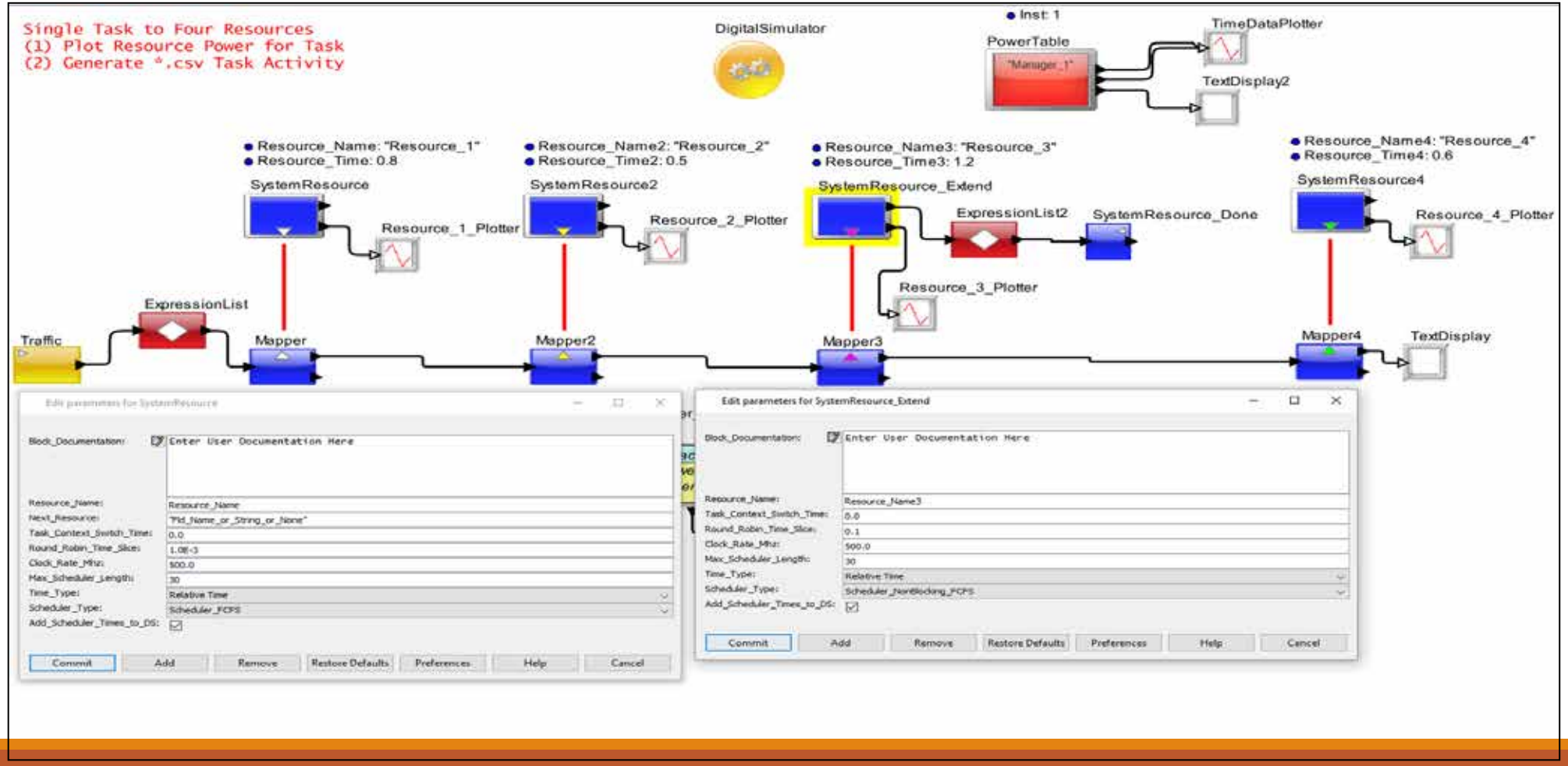
- Double click to configure

Application comparisons

Features	SystemResource	SystemResource_Extend
Preemption	Yes	No
Hierarchical	Yes	No
Extended Task Processing	No	Yes
Non-Blocking	No	Yes

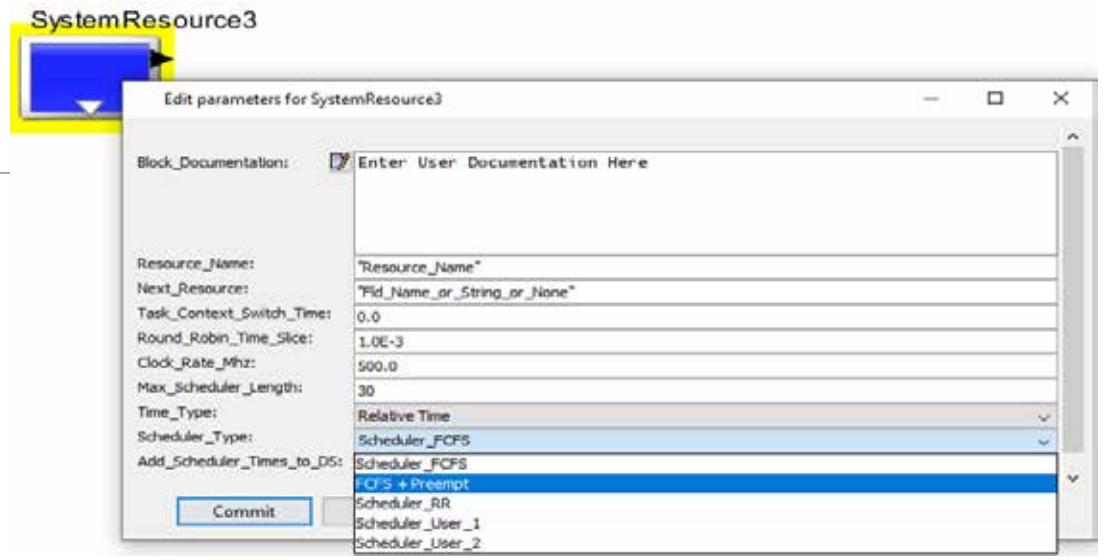


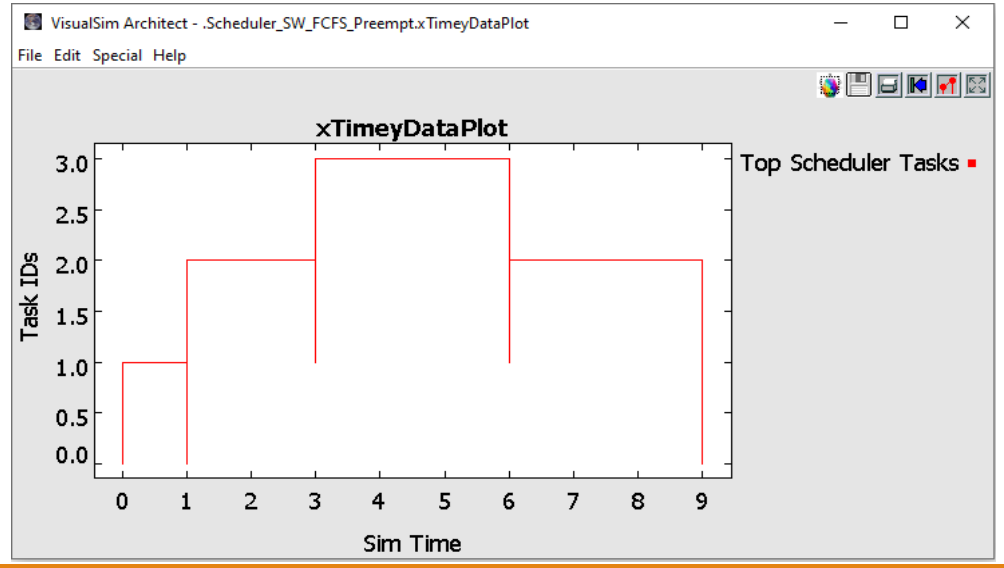
Example Model



Preemption

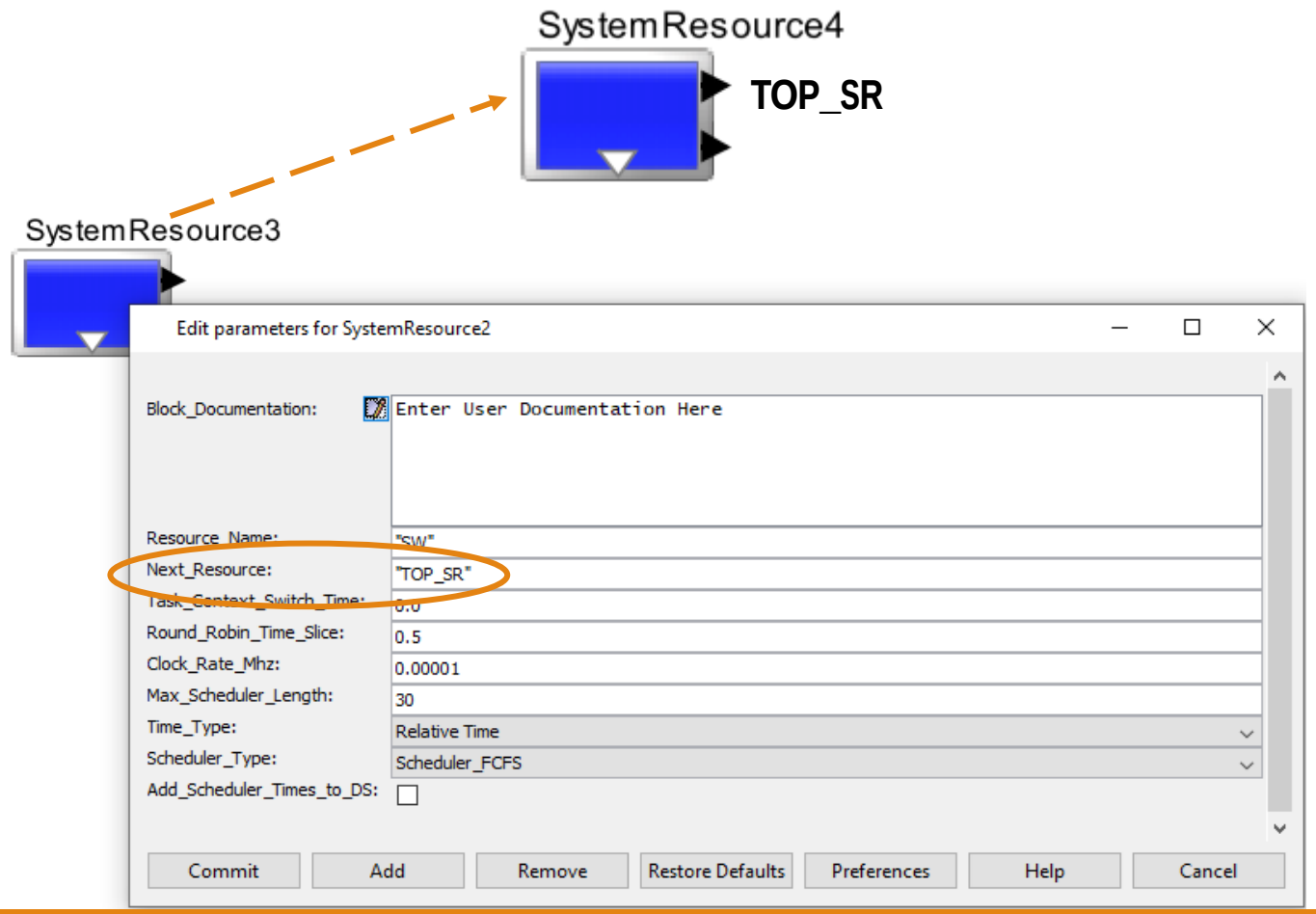
- Possible with System_Resource
- High priority task preempts the currently executing Low Priority task





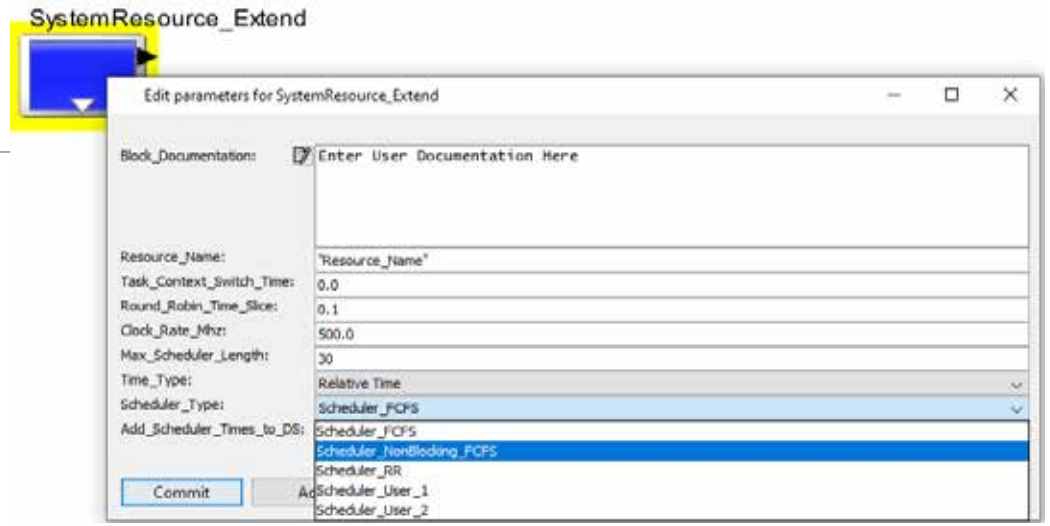
Hierarchical

- Possible with System Resource
- The System Resource can refer to another System Resource or System Resource Extend for actual Processing

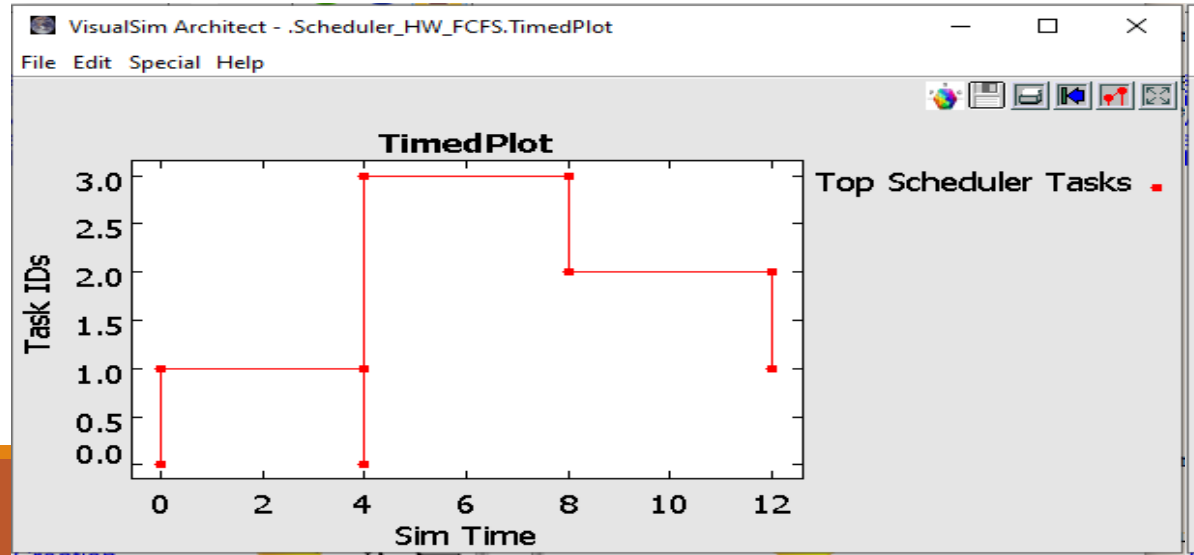


Non Blocking

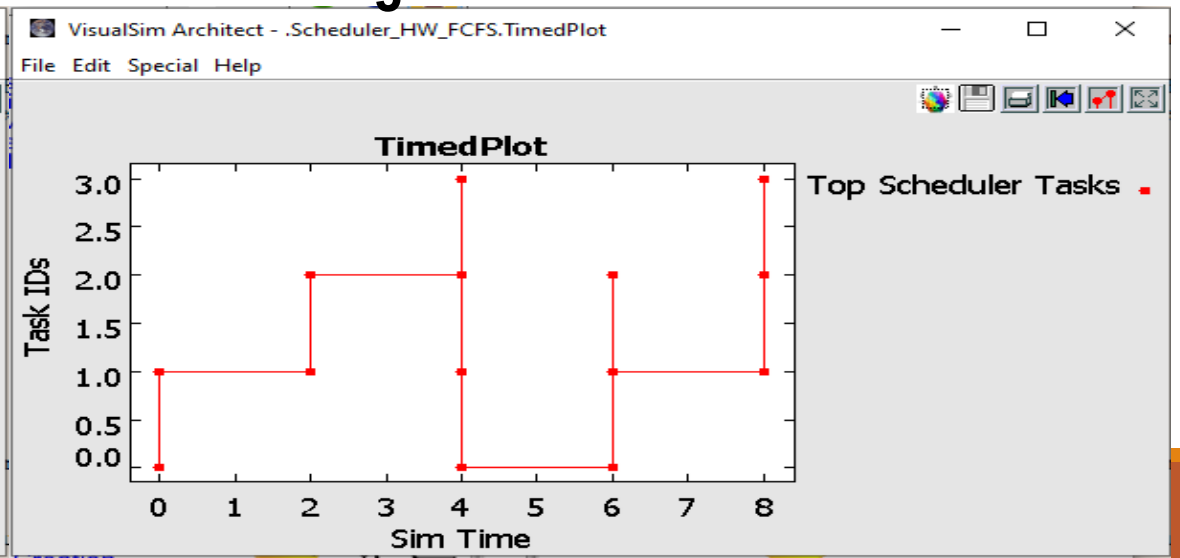
- Possible with System_Resource_Extend
- Multiple Data Structures can be executed between the output and the SystemResource_Done block.



FCFS

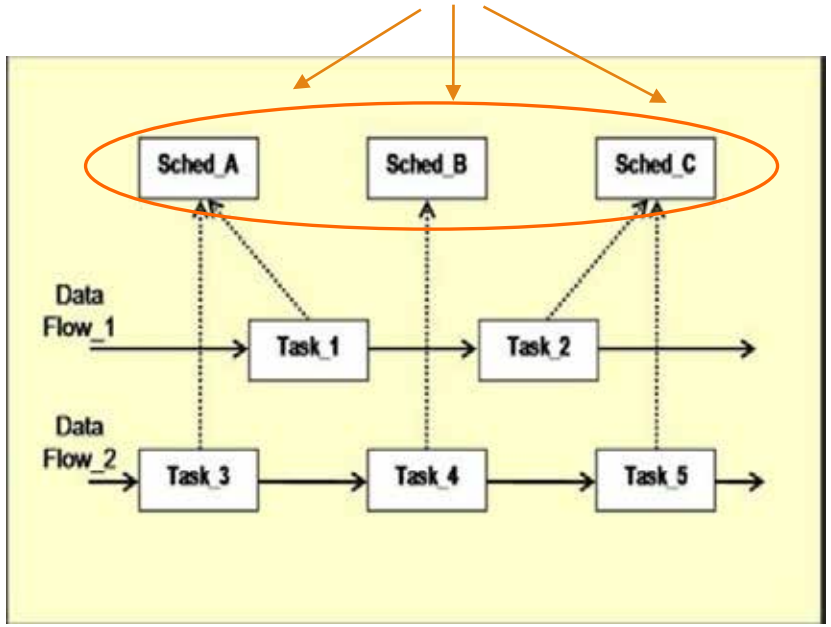


Non- Blocking FCFS



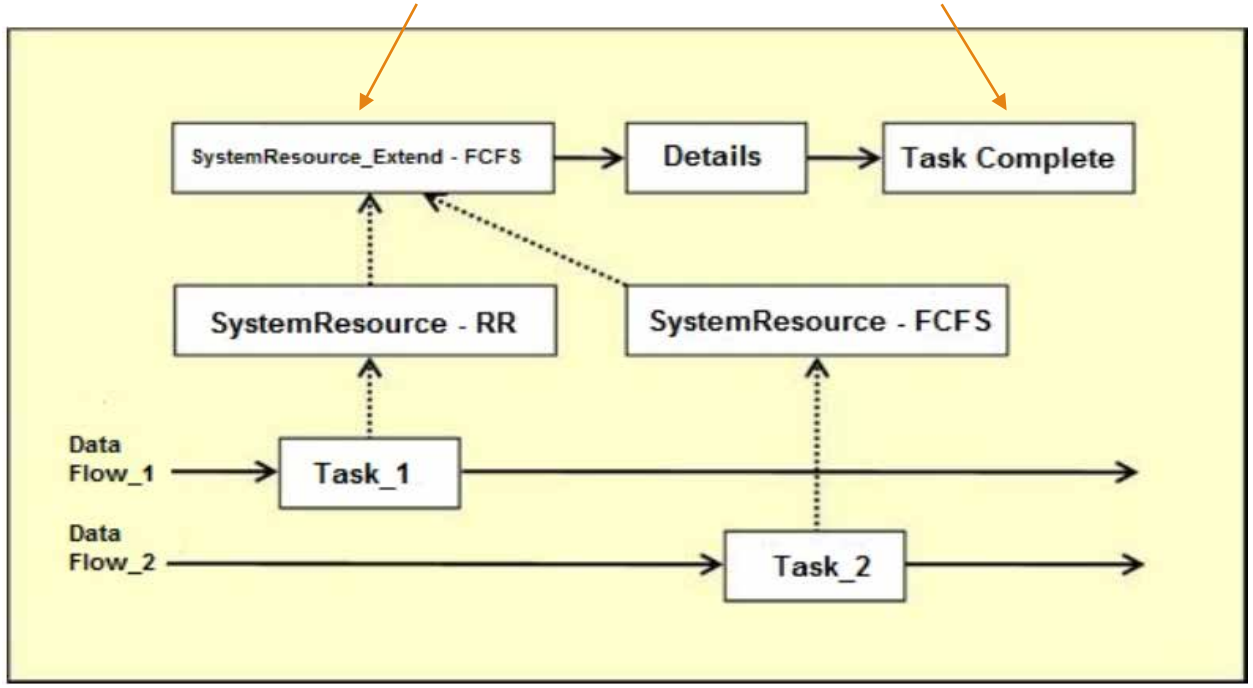
Application Examples of System Resources

System Resource



SystemResource_Extend

SystemResource_Done



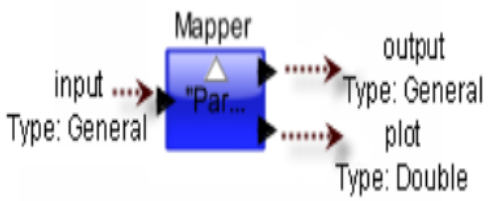
Differences

Features	System Resource	SystemResource_Extend
Preemption	Yes	No
Hierarchical	Yes	No
Extended Task Processing	No	Yes
Non – Blocking	No	Yes

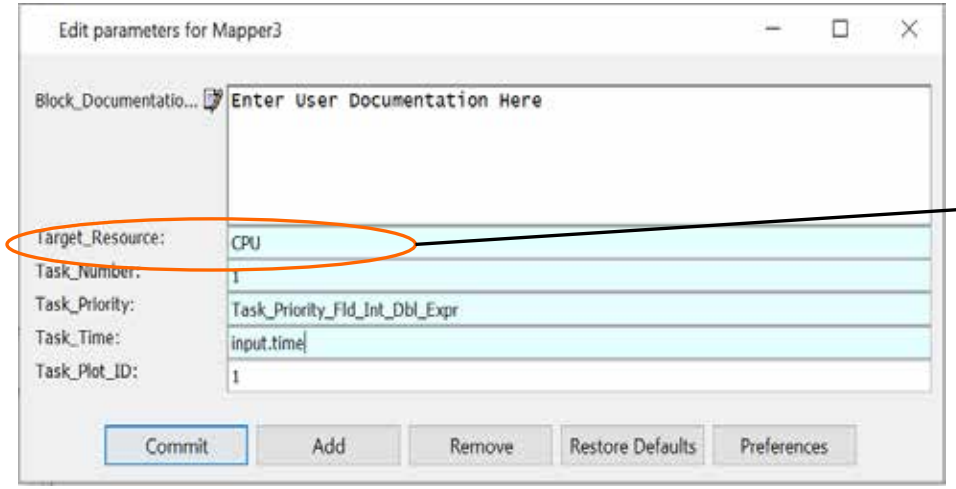
What is Mapper?

Connect behavior flow with architecture resources

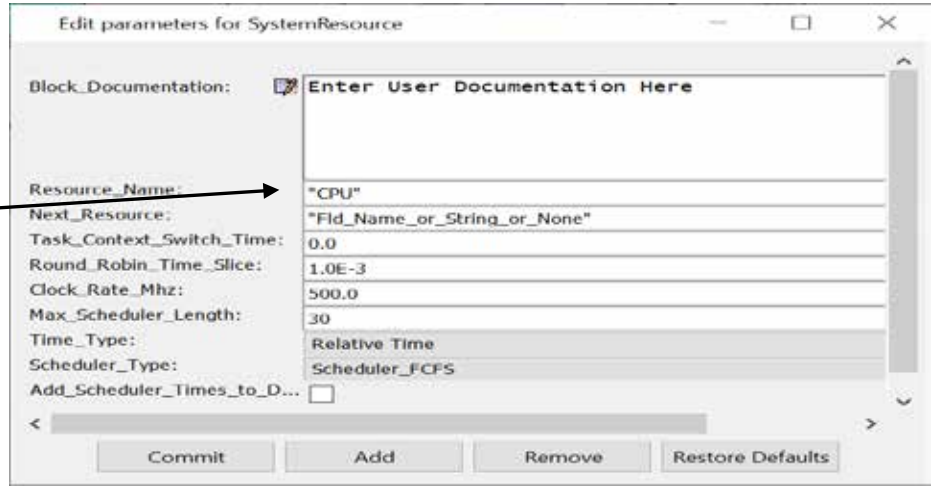
- Takes incoming Data Structure and sends to
  - SystemResource
  - SystemResource_Extend blocks
- Placed in the behavior flow where timed resources required
- Consumes zero time, no queue, no arbitration



Mapper



System Resource

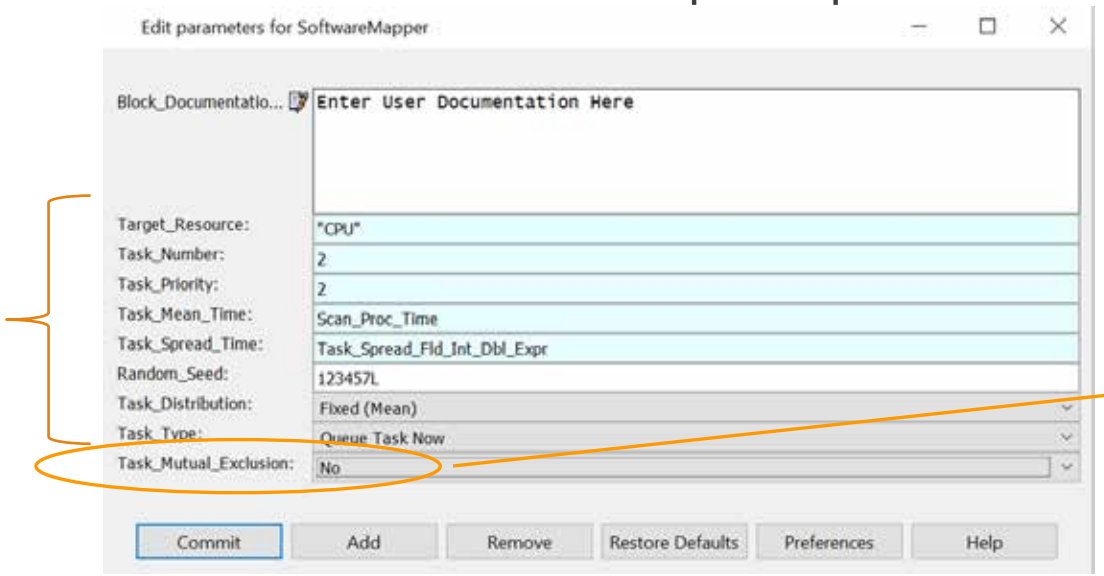


Software Mapper

Hardware or Software Task issuer

- Sends tasks to SystemResource or SystemResource_Extend based on Target_Resource
- Delay at Resource provided by the distribution between Task_Mean_Time, Task_Spread_Time and the Task_Distribution
- Block can either Queue incoming Data Structure or send to SystemResource immediately
- Mutex=true means that the DS cannot be preempted

Attributes to issue the task to System Resource

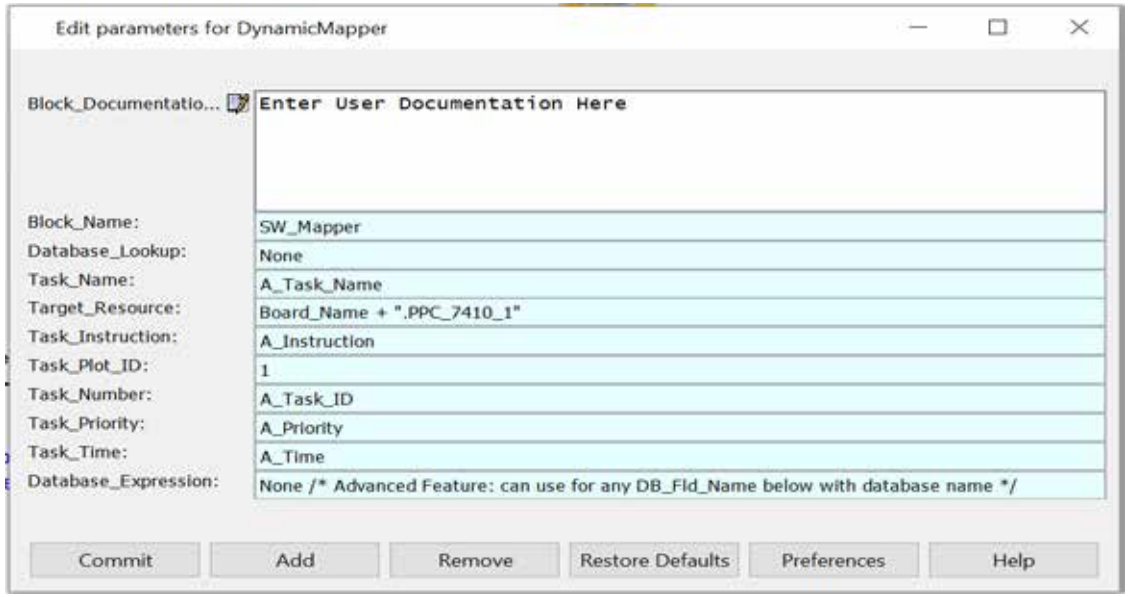


Lock out all other tasks from preempting this Task at the SystemResource

Dynamic Mapper

Mapping of tasks on

- Target processor,
- SystemResource
- SystemResource_Extend

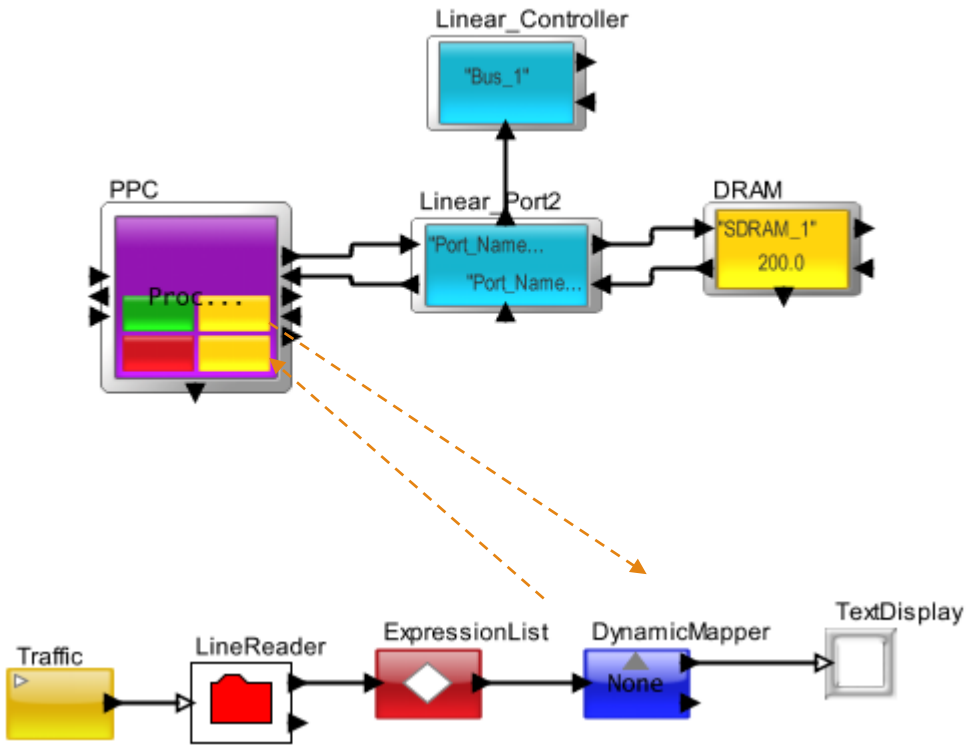


Dynamic Mapper Overview

- This block accepts a data structure on the input and sends this along with the information in block parameters to the target resource (processor or SystemResource).
- When the resource completes the processing, the data structure is returned to this block, which places it on the output port. The Task_Destination determines the target resource.
- The Destination, Instruction, Time and Priority can be accessed from the Database or the field content. If the Database_Lookup is the Linking_Name of a block, then a database is available. The database row to use is matched with the Task_Name from this block.
- If the Database_Lookup value is "None" or default, then no database is available.

Dynamic Mapper to Processor

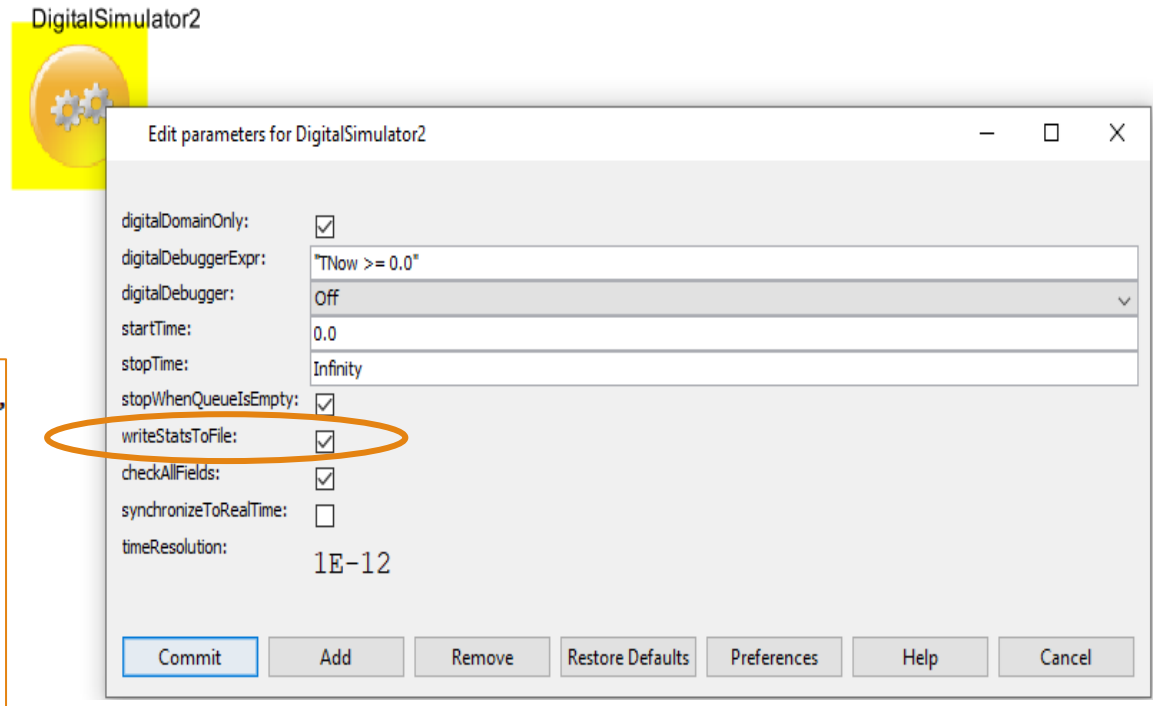
Mapping function to a target processor



writeStats To File

- Generates Statistics for all the blocks in the model at the end of simulation
- Writes into a Text File in the model directory

```
Queue_Statistics                6.00000000000 sec
{BLOCK                        = "SR_SrExtend_example.SystemResource_Extend",
DELTA                         = 0.0,
DS_NAME                       = "Queue_Common_Stats",
ID                            = 1,
INDEX                         = 0,
Number_Entered                = 7,
Number_Exited                 = 1,
Number_Rejected               = 0,
Occupancy_Max                 = 6.0,
Occupancy_Mean                = 3.7777777777778,
Occupancy_Min                 = 1.0,
Occupancy_StDev               = 1.4740554623802,
Queue_Number                  = 1,
TIME                          = 6.0,
Total_Delay_Max               = 4.0,
Total_Delay_Mean              = 4.0,
Total_Delay_Min               = 4.0,
Total_Delay_StDev             = 0.0,
Utilization_Mean              = 0.0}
```



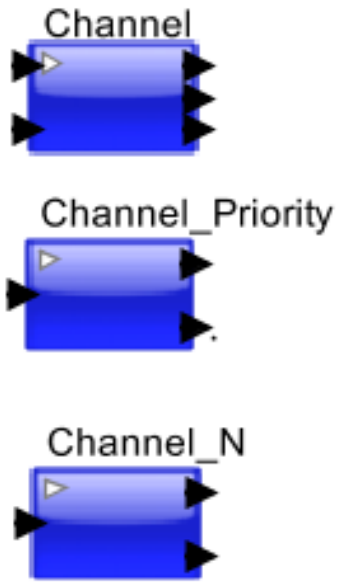
Using Channel Block

Types of System Components

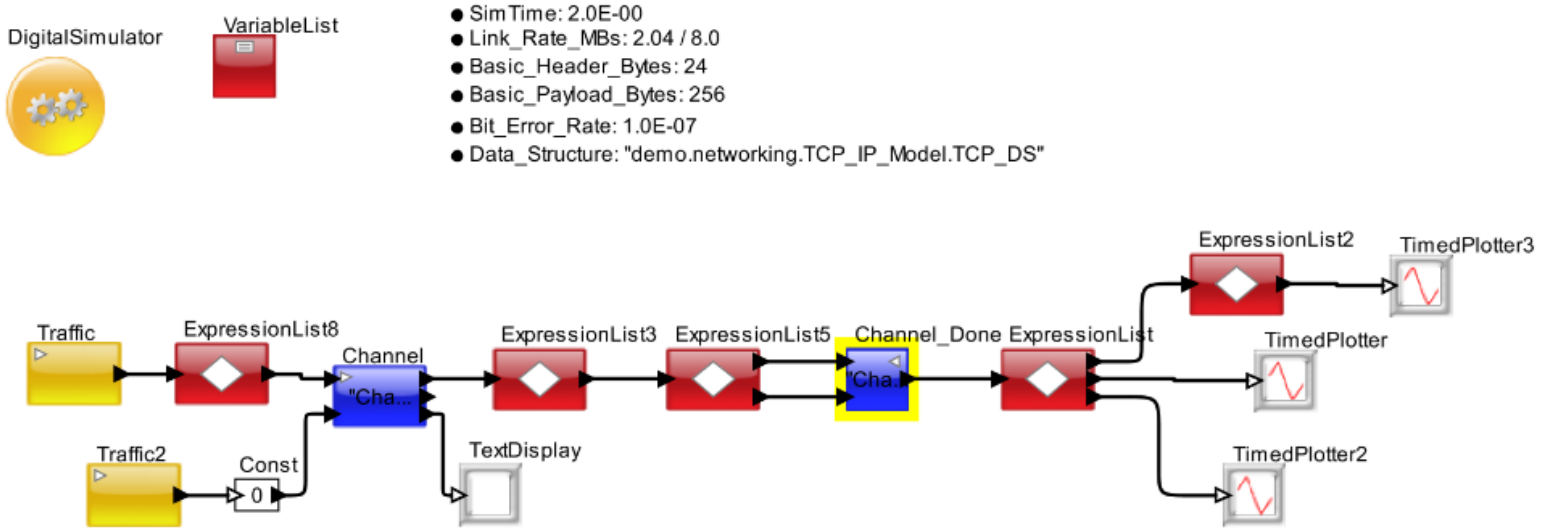
- Entity
  - ✓ Bus
  - ✓ DMA Controllers
  - ✓ Wireless Channel
- Using Channels in place of Timed Queue
  - ✓ Add logic to each Server resource
  - ✓ Contains both 1-to-many and 1-to-1 channel structure
  - ✓ Add more details when access to the channel is provided
  - ✓ Impact of data access or fragmentation
  - ✓ Create multiple channels from a single buffer
  - ✓ Create non-blocking and blocking conditions

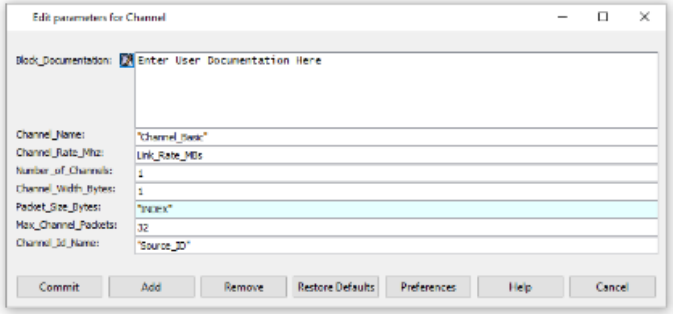
Channel Blocks

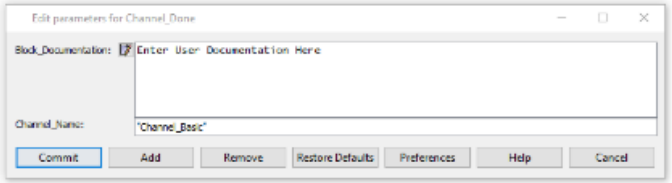
- Channel, Channel_Priority, Channel_N & Channel_Release
  - ✓ Channel_N has a dedicated queue for each Channel
  - ✓ Channel and Channel_Priority have a single Queue
  - ✓ Channel_Priority support priority for ordering the queue only
- Model fixes number of channels
- Queues request and allocates channels as they become available
- Channel Block
  - ✓ Latency= Channels Activity + Channel_Rate * Packet_Size
- Channel_N and Channel_Priority
  - ✓ Latency= Channels Activity
- Channel_Release releases the Channel and transmit next task
  - ✓ Supports retransmission for rejected transactions
- Used for modeling Bus, wireless channels and DMA



Using the Channel Blocks



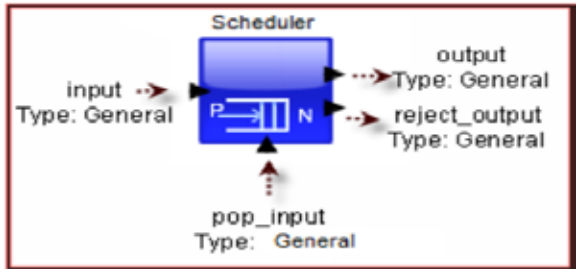


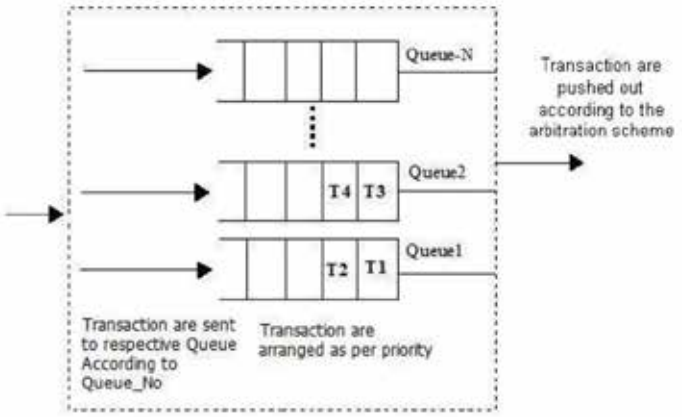


Scheduler

- Reordered based on
  1. Priority

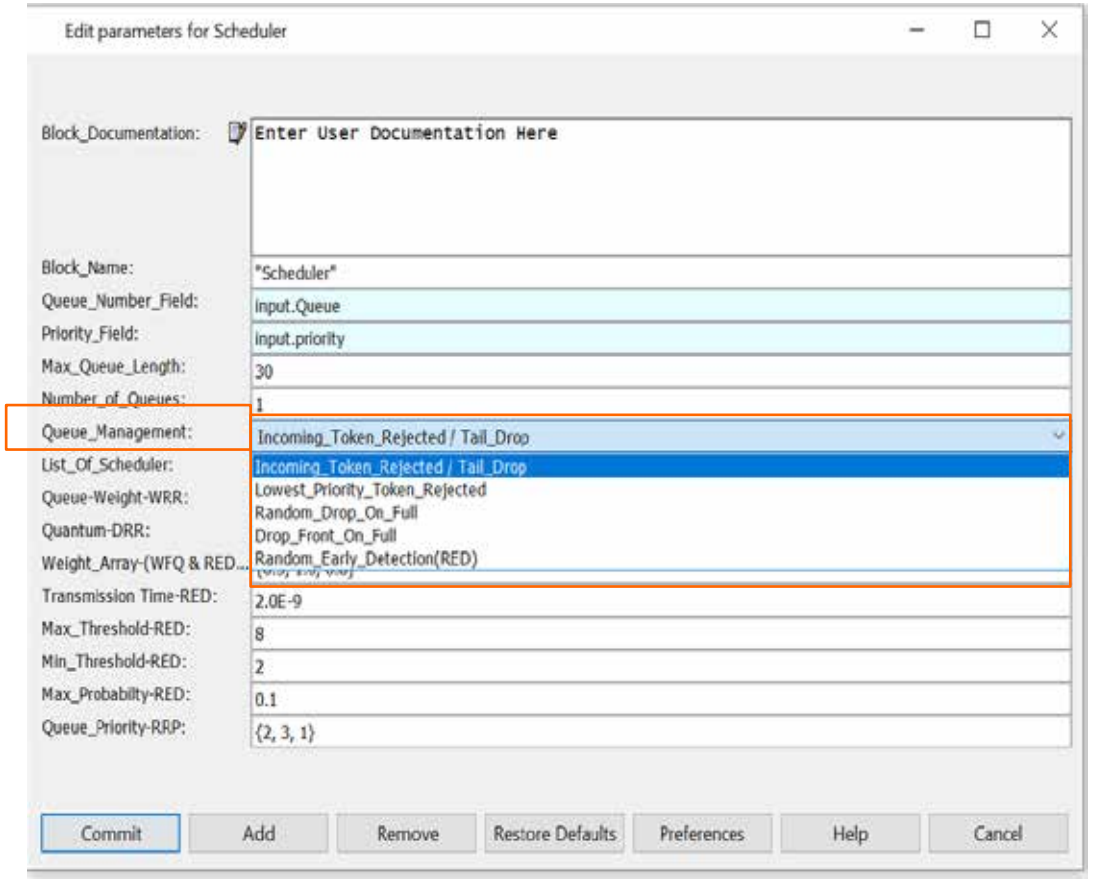
- Queue Management:
  1. FIFO
  2. LIFO
  3. FCFS
  4. Round Robin
  5. Weighted Round Robin
  6. Weighted Fair Queuing
  7. Deficit Round robin
  8. Strict Priority

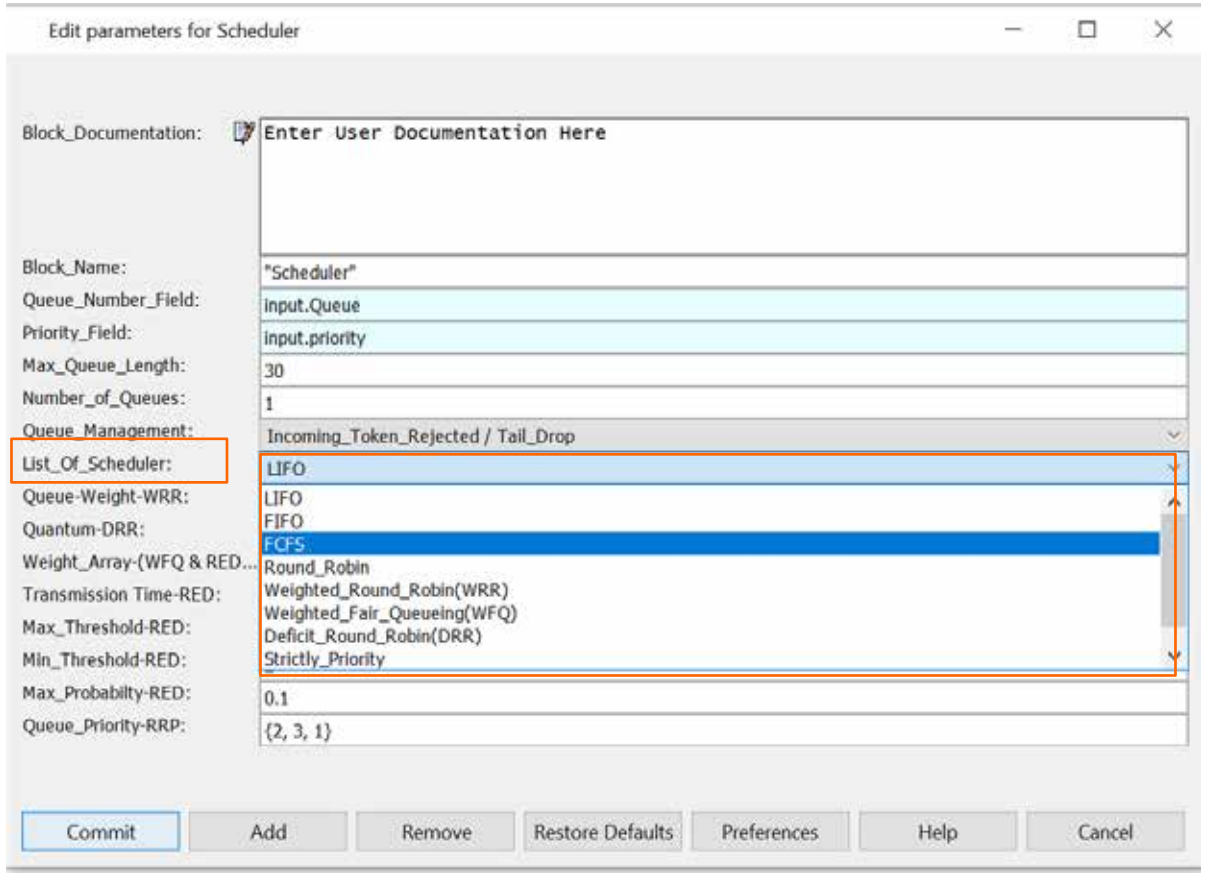




Queue N with Priority

Scheduler





Return

Differences and Usage

- EventQueues and Queues
  - ✓EventQueue requires multiple inputs for store and pop
  - ✓Queues access required data from fields of the Data Structure
  - ✓Queues content and statistics access via RegEx
- TimedQueues and Server
  - ✓Timed Queue requires multiple inputs for store and pop
  - ✓Server access required data from fields of the Data Structure
  - ✓Server has SLOT queue types for multiplexed access
  - ✓Queues content and statistics access via RegEx

Differences and Usage

- Server and Delay
  - ✓Delay schedules each incoming data structure at the Current Time + Delay
  - ✓Delay block does not preserve the order
  - ✓Server block first queues each data structure
  - ✓Server block starts the delay only when the data structure is the head of the queue
  - ✓Server block preserves the order
  - ✓Server block queue is reordered based on Priority
- Server and SystemResource
  - ✓Server block is used when there are a large number of identical devices
    - ✓ Bus with parallel lanes, multiple core, multiple input channels and output channels
  - ✓SystemResource is used when Requesters are distributed in the model
  - ✓There can only be 64 SystemResource in a model
  - ✓SystemResource support Preemption, hierarchical reference and advanced processing

Learn More by Reviewing Training Recordings

Watch Tutorials

- Training Part 1 (54 minutes):
  https://www.youtube.com/watch?v=9JHcLm0w2-4
- Training Part 2 (65 minutes):
  https://www.youtube.com/watch?v=LY-imqaSBwc
- Training Part 3 (42 minutes):
  https://www.youtube.com/watch?v=3H7YaZ0wrwg

Debugging

Debug

- Debugging is the process of:
  a. figuring out where the bug is
  b. figuring out how to fix it.
- Debugging proceeds from the point at which the realization of an error occurs, to finding the earlier point at which the error was introduced.

Types of Debuggers

- Breakpoints
- Stop & Restart
- Trace Tracking
- Animation
- Dynamic Plotters
- Listen to Port
- Listen to Block
- Listen to Simulator
- Digital Debugger
- Error Messages
- Batch Mode Simulation
- Power Timing Diagram
- Variable Dump
- RegEx
- Script Debugging
- Data Structure Fields
- AutoSave
- Logger for Verilog and SystemC
- Plotters & Text Display

Model Construction - Strategy

Create blocks as individual Sub-Models and Hierarchical blocks
- Test each Hierarchical block prior to adding to a big model

Build-in tests using Displays, debug statements

Check flows within a Hierarchical block using
- Animation and the simulation profiler set to Run mode

Check block usage
- Set Digital Simulator profile to Summary mode

Set a Debug flag for each Hierarchical or sub-model.
- Use this to turn on and off debug statement

Setup monitors for key fields.
- Look at a smaller set of output in the Displays and Listen

During and End of Simulation

Error Message

- Identify the block listed in the Error Description and the Block Highlight in the block diagram view
- Review Possible Solution description to resolve

Listen to Block to see if the sequence of execution is correct

Listen to Port to see if the input and output field values are correct

Variable Dump block and see the values of the memory

View Command Line at the end of the simulation for the summary of total Simulator events, synchronous event, asynchronous mix events, time taken and memory used

Follow the Data Flow

If TextDisplays has no output

- Check the block before it and follow up with each prior block
- Listen to Port to check the output values

Start at Hierarchical block levels to identify the right block

- Instance will display the Listen to Port.

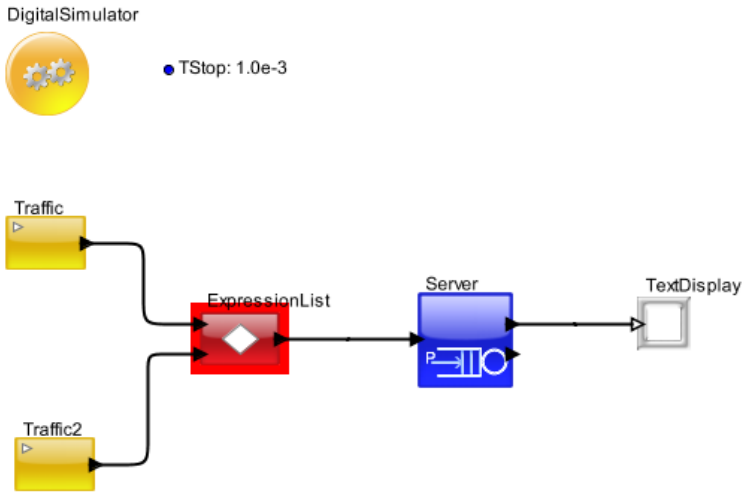
Make sure the transaction is arriving from the correct source and going to expected destination

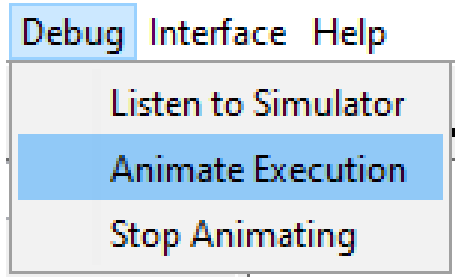
Does the Transaction ID sequence make sense?

Any special Transaction flags set, indicating mode of operation that is inconsistent with current block

Animate Execution

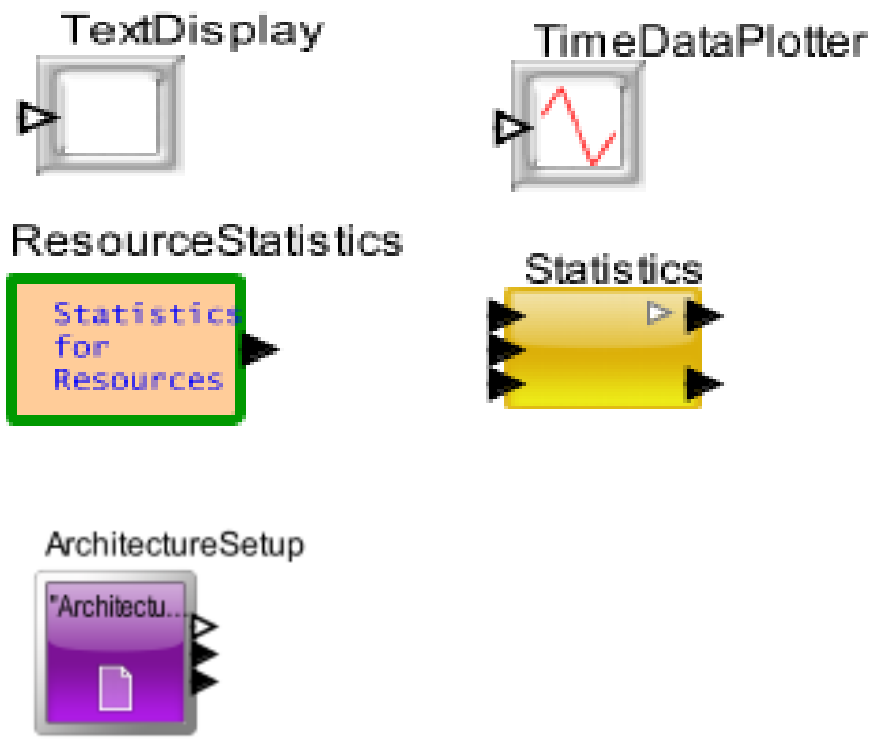
- To view the dynamic operation of the model
- The Executing Block gets highlighted
- The time to highlight a block is in **milliseconds**
- To Start Animation:
  Debug -> Animate Execution
- To Stop Animation:
  Debug -> Stop Animating





Text Display, Plotters, Statistics

- To graphically display and analyze data collected from the simulation.
- Helps to detect any errors in the behavior
- Statistics Generators - Generates statistics of all resources and hardware blocks
- Extracts the appropriate fields in the data structure or the entire object and display them.
- Plotter – Latency, throughput, etc
- Text Display – Entire Data Structure, any value coming in the input port



Statistics to Identify Behavior Errors

Resource statistics

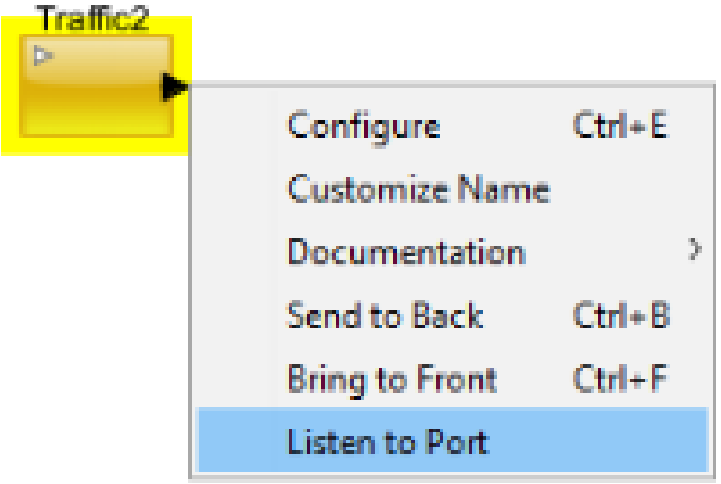
```
DISPLAY AT TIME                    ------ 100.00000000000 sec --
{BLOCK                        = "Resource_Statistics.Queue",
DELTA                         = 0.0,
DS_NAME                       = "Queue_Common_Stats",
ID                            = 6,
INDEX                         = 0,
Number_Entered                = 199,
Number_Exited                 = 12,
Number_Rejected               = 157,
Occupancy_Max                 = 30.0,
Occupancy_Mean                = 20.0754716981132,
Occupancy_Min                 = 0.0,
Occupancy_StDev               = 10.1402412546265,
Queue_Number                  = 1,
TIME                          = 100.0,
Total_Delay_Max               = 91.0182813545,
Total_Delay_Mean              = 26.2911000907667,
Total_Delay_Min               = 0.0,
Total_Delay_StDev             = 23.8057259325954,
Utilization_Mean              = 0.0}
```

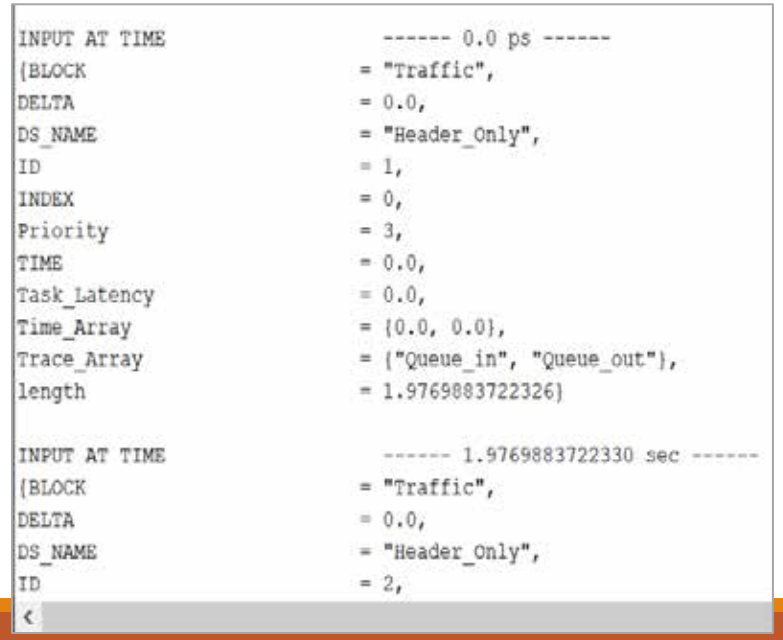
Architecture Setup

```
DISPLAY AT TIME                    ------ 11.42860 us ------
{AHB_Bus_Delay_Max              = 1.8E-8,
AHB_Bus_Delay_Mean              = 1.8E-8,
AHB_Bus_Delay_Min               = 1.8E-8,
AHB_Bus_Delay_StDev             = 0.0,
AHB_Bus_IOs_per_sec_Max         = 524998.6875032812,
AHB_Bus_IOs_per_sec_Mean        = 524998.6875032812,
AHB_Bus_IOs_per_sec_Min         = 524998.6875032812,
AHB_Bus_IOs_per_sec_StDev       = 0.0,
AHB_Bus_Input_Buffer_Occupancy_in_Words_Max       = 8.0,
AHB_Bus_Input_Buffer_Occupancy_in_Words_Mean      = 3.4736842105263,
AHB_Bus_Input_Buffer_Occupancy_in_Words_Min       = 0.0,
AHB_Bus_Input_Buffer_Occupancy_in_Words_StDev     = 3.4084789176194,
AHB_Bus_Preempt_Buffer_Occupancy_in_Words_Max     = 0.0,
AHB_Bus_Preempt_Buffer_Occupancy_in_Words_Mean    = 0.0,
AHB_Bus_Preempt_Buffer_Occupancy_in_Words_Min     = 0.0,
AHB_Bus_Preempt_Buffer_Occupancy_in_Words_StDev           = 0.0,
AHB_Bus_Throughput_MBs_Max      = 8.3999790000525,
AHB_Bus_Throughput_MBs_Mean     = 8.3999790000525,
AHB_Bus_Throughput_MBs_Min      = 8.3999790000525,
AHB_Bus_Throughput_MBs_StDev    = 0.0,
BLOCK                           = ".Processor_Power_model.ArchitectureSetup",
Cache_Delay_Time_Max            = 5.0E-8,
Cache_Delay_Time_Mean           = 5.0E-8,
Cache_Delay_Time_Min            = 5.0E-8,
Cache_Delay_Time_StDev          = 0.0,
Cache_Hit_Ratio_Max             = 100.0,
Cache_Hit_Ratio_Mean            = 100.0,
Cache_Hit_Ratio_Min             = 100.0,
Cache_Hit_Ratio_StDev           = 0.0,
Cache_Memory_Used_By_MAC_ARM9_MB_Max      = 9.6E-5,
Cache_Memory_Used_By_MAC_ARM9_MB_Mean     = 9.6E-5,
Cache_Memory_Used_By_MAC_ARM9_MB_Min      = 9.6E-5,
Cache_Memory_Used_By_MAC_ARM9_MB_StDev    = 0.0,
Cache_Memory_Used_By_Total_MB_Max         = 9.6E-5,
Cache_Memory_Used_By_Total_MB_Mean        = 9.6E-5,
Cache_Memory_Used_By_Total_MB_Min         = 9.6E-5,
```

Listen to Port

- Displays each token passing through the port
- Used to check whether the data is flowing through the particular port
- If no data on the "Listen to Port" window , it indicates that the model did not generate any output from that block.
- If there is no data, then one needs to check the ports, or virtual connections driving this block to see that they are being activated correctly.
- Helps to debug errors in connections, routing and conditional branches.
- Usage
- Right click the required port and select Listen to Port



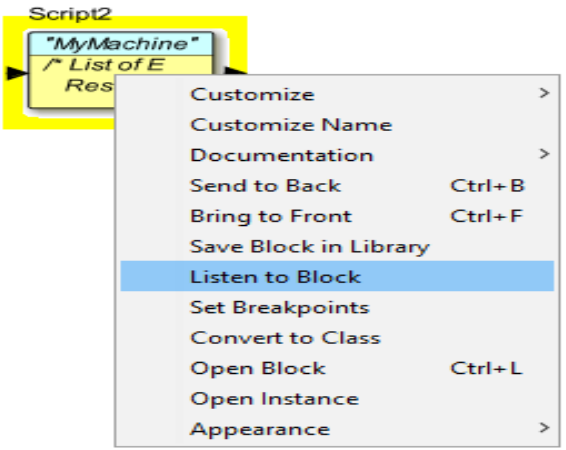


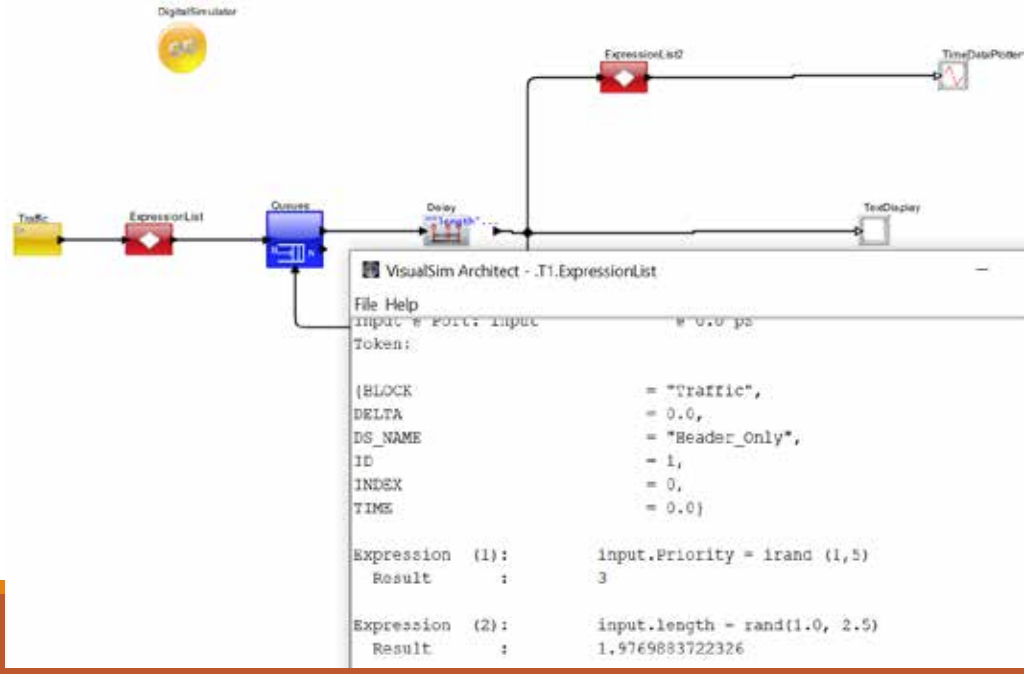
Listen to Block

- Gives more insight into the internal block operation
- Shows the sequence of execution, entry/exit, virtual Send, threads

Support coverage

- All blocks except
  - Hierarchical blocks
  - instantiated hierarchical class
- Provides simulator level information relative to methods being fired, and so on. Simulator information is most useful for evaluating custom blocks in the simulation environment.
- Right click on the block and select Listen to Block

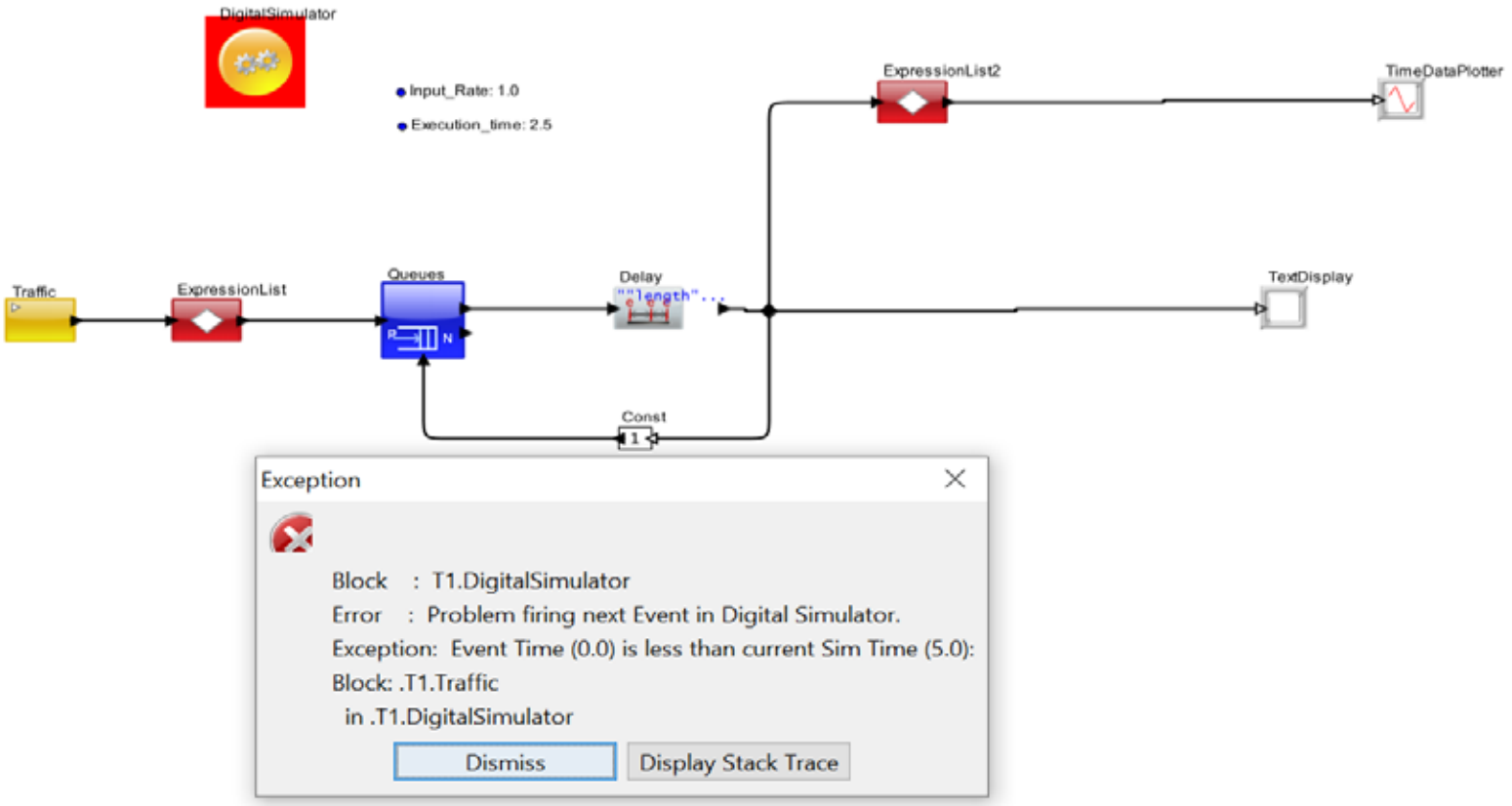




Error Message

Message

- Highlight Error Block
- And Error Message



Fields of the Data Structure

- Each Data Structure (DS) has header fields that provide information as to its source, ID increment, and time created.
  - Traffic field list the originating block
  - ID indicates the sequence of generation
  - TIME is a generation time stamp
- These are valuable clues if a request does not arrive or arrives out of order.

```
DISPLAY AT TIME                    ------ 0.0 ps ------
{BLOCK                           = "Traffic",
DELTA                            = 0.0,
DS_NAME                          = "Header_Only",
ID                               = 1,
INDEX                            = 0,
TIME                             = 0.0}
```

Time and Task Tracer fields

- Set of arrays that are updated with the **name and time stamp** every time the Data Structure enters or departs a Resource or Architecture block.

```
Time_Array              = {0.0, 5.0E-4, 5.0E-4, 1.0E-3},
Time_In_Resource        = 5.0E-4,
Trace_Array             = {"Server_in", "Server_out", "Scheduler_in", "Scheduler_out"}}
```

Tracer

- Enables the capture of execution of multiple Scripts from one location and the content is written to a field.

ScriptTracer

- Helps to check the interaction between scripts and if the sequence of execution of instructions is correct.

- Data is written into a file called **VisualSimTraceLog.txt** which can be seen under **users/user folder/.VisualSim/**

Memory Dump/Variable Dump

- This outputs the current value of all the global and local variables in the model

- The output is a data structure with each field representing one of the memories.

- Full Library ->Model-> Utility-> Checkers-> Variable_Dump

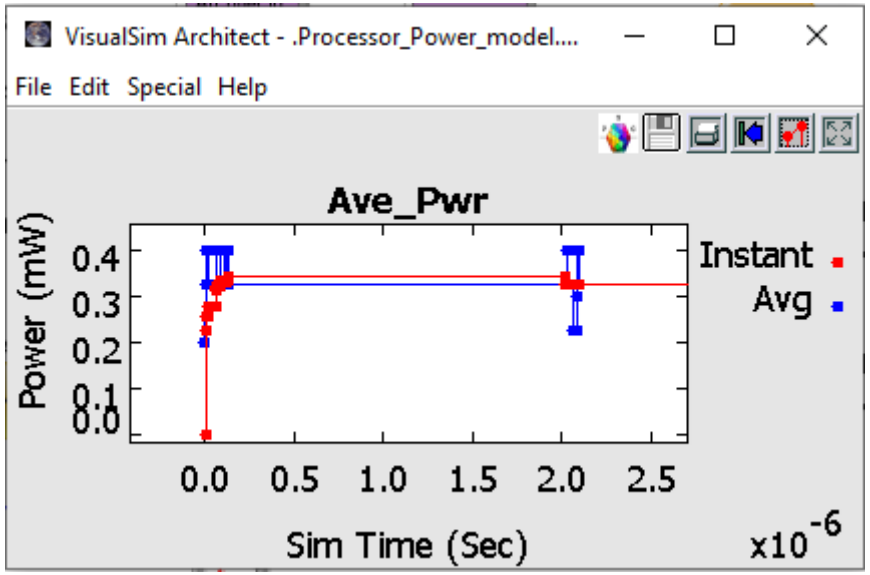
Power Timing Diagram

- Used to analyze the power consumption, battery discharge, dynamic system changes, power state changes of the devices which impacts the system timing.

- Outputs the instantaneous, average and State_Change information of top level and the other powerTable located in hierarchical blocks.

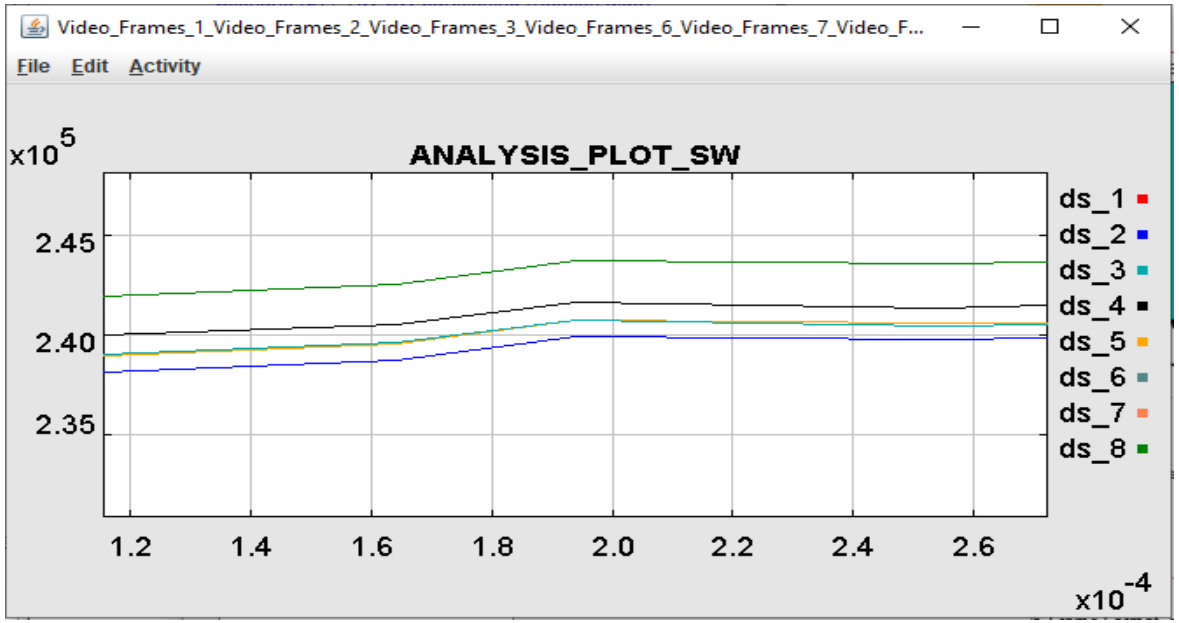
- **Power Statistics and Report**

✓ Instantaneous Power (port)
✓ Average Power consumed (port)
✓ Power Dissipated (port)
✓ Instant (powerCurrent) and total power consumed (powerCumulative) by device



Batch Mode Simulation

- Batch Mode Simulation enables the user to schedule multiple simulation runs with different parameter values.
- The Plot manager is linked to the Post Processor
- The success or failures of the simulation runs are reported on the terminal windows executing the script and in the "Batch_Mode_Results_Summary.txt".



RegEx

- Provides status and visibility into resource and hardware blocks in the execution flow.

✓readAllVirtual( ) - Provides the List of virtual Blocks in the model.

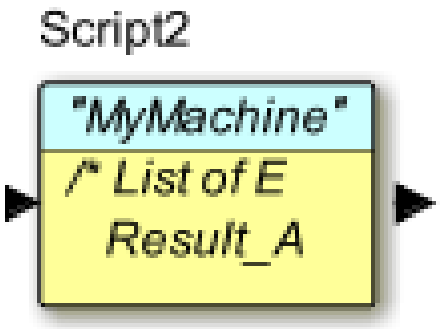
✓readAllMemory( ) - Provides a output of all local and global memories and their current value.

✓getBlockStatus( ) - The statistics for the blocks are generated using the getblockStatus RegEx function with the type, length, stats etc.,

✓getResourceActivity( ) - Used to access the information in the database block. It returns an array of the current queue length of the resources listed in the named column.

Script Debugging

- Breakpoints
- Check variables
- Profiling
- Single cycle step-through
- Script Profiler
- sendToCommandLine(“MyMessage”) added to script
- Build-in DEBUG messages for subsystem
  - ✓Entering, Exiting Subsystem
  - ✓Executing each major block
  - ✓Unexpected states
  - ✓One line per transaction, easy to read



Script Profiler

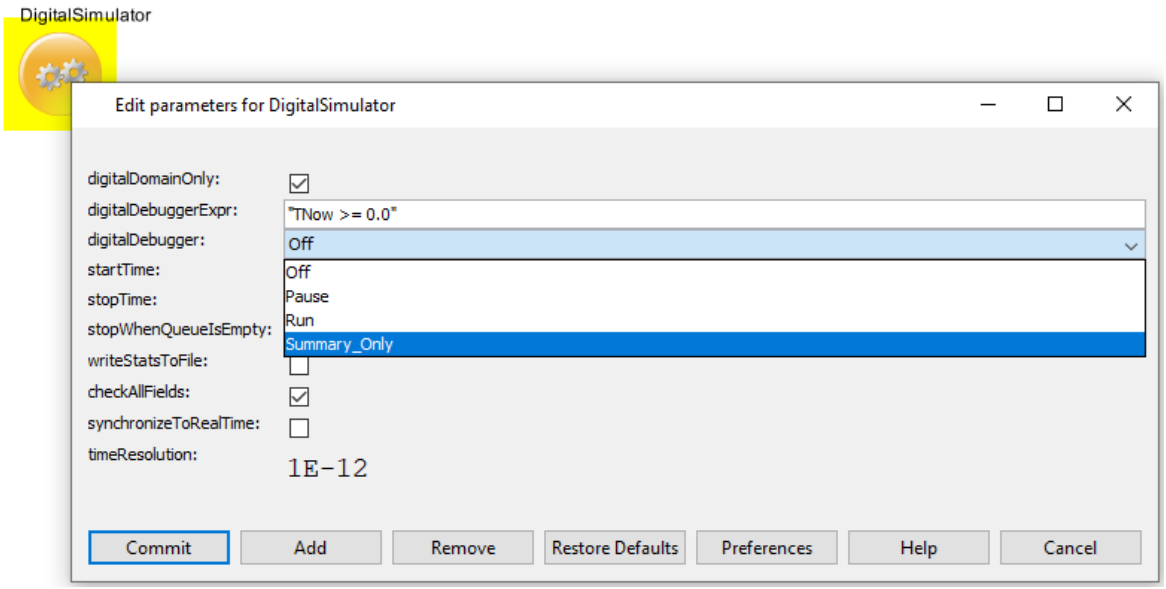
- Keeps track of the number of times a statement executes and the average time the statement took to execute down to the nano-second level.
- Referred to as Code Coverage in lower level verification testing
- Used for finding Algorithmic Bugs after finding functional bugs

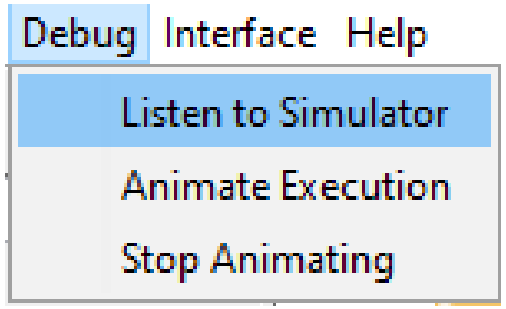
Profiling

- Generates the Instruction Mnemonics for the execution of the script
- Used to check the Algorithmic Flow

Listen to Simulator- Digital Debugger

- Provides a sequence of execution for the selected simulation.
- Integrated with Digital Debugging utility in the Digital simulator.
- This window displays the usage statistics, current block execution, and model summary information.
- **Debug -> Listen to Simulator**





Digital Debugger parameters

- Off – Disables the Debugging Mode
- Pause – Turns the Debugger to Stop at every block in the model Flow. Provides summary at the end of Simulation
- Run – Records the order in which each block is fired in the model. Provides summary at the end of simulation.
- Summary Only – Generates the List of all the blocks at the current level of simulation and the levels below

For each Block

- ✓ Records the number of time each block is fired in the model
- ✓ Average execution time for each firing
- ✓ Total time spent in each Block
- ✓ It also lists the Blocks that are not executed in the model

Pause and Resume

- Saves the simulation data, events and status in a file
- Handy to debug simulations that run for a large period of time.
- User can analyze system behavior at various points in the simulation.
- User can pause at a timestamp and analyze the system response and continue simulation step by step from that point onwards.
- The system can be analyzed for required functionality and also helps the designer to identify if the crucial tasks are being executed within the deadlines.

Autosave

- VisualSim provides the ability for the user to continuously save the currently open model, if they are modified.
- The interval between the saves is set in the VSconfig.properties. The format is: Auto_Save_Time=2 Where the number on the RHS is the time interval between saves.
- The intermediate xml files are saved in the <User Home>/.VisualSim directory.
- The format is as follows: Scheduler_SW_FCFS_Preempt_20151127_143903.xml Where

  File Name : Scheduler_SW_FCFS_Preempt

  Date : 20200214- Here is it 2020 February 14

  Time : 193603 is 7:36:03 PM

Checking Timing and Events

- Check expected order of execution
- Check Event Names match between WAIT, EVENT
- Check Event_Name field is removed after use and not reused inadvertently
- Check whether transactions are generated at the clock boundaries
- Perform listen to block of Script to see timing of transactions, types
  - ✓ Add messages to keep track of array values
  - ✓ Try to capture the essence of the block execution in one line messages

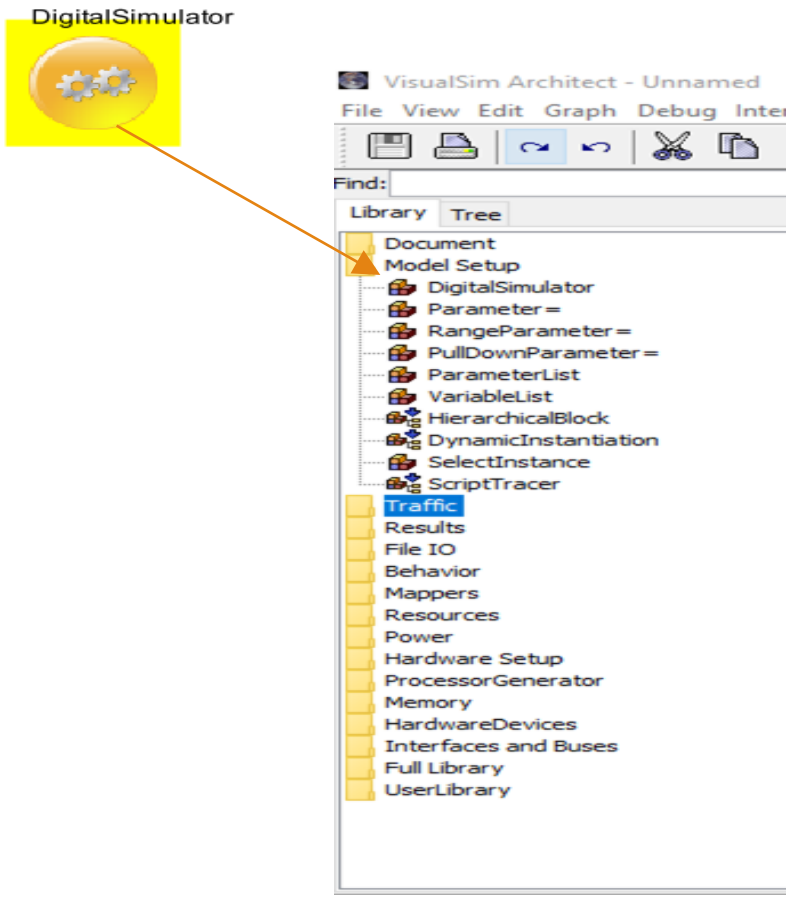
Other Debugging Functions

- Code Debugging
  - ✓ Attach a standard debugger to the process
  - ✓ Add any compiler options to the batch file
  - ✓ Copy logger.Properties from doc/SystemC directory (Used in CustomCPP, Verilog and SystemC)
    - Visibility into all communications and execution at code-level

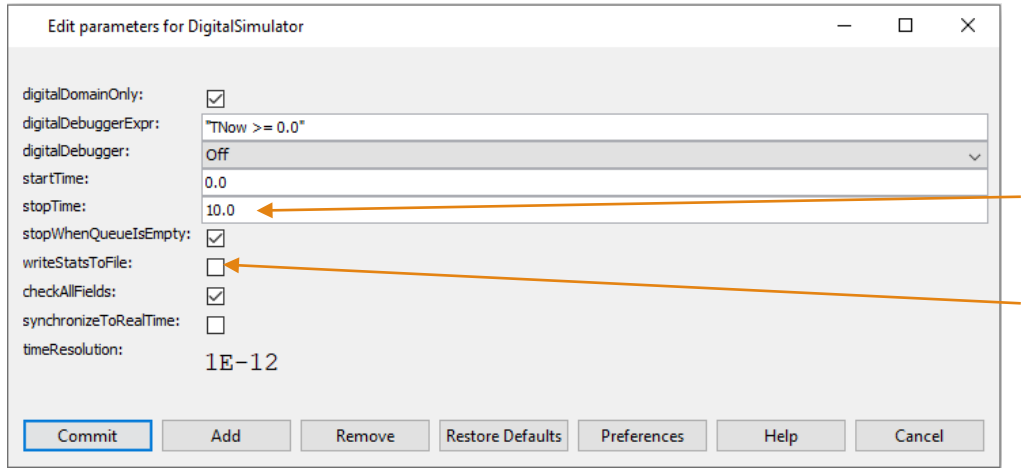
Configuring System Blocks

Digital Simulator

Where to get digital simulator?



Common Parameters to Edit

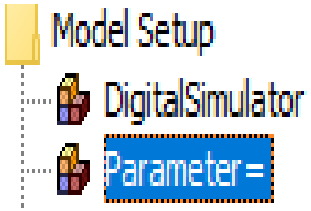


Simulation Duration

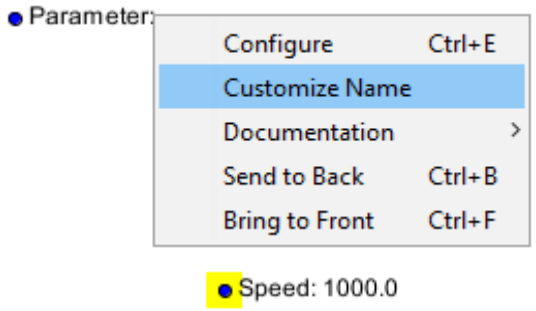
Enable Statistics Report

Parameter

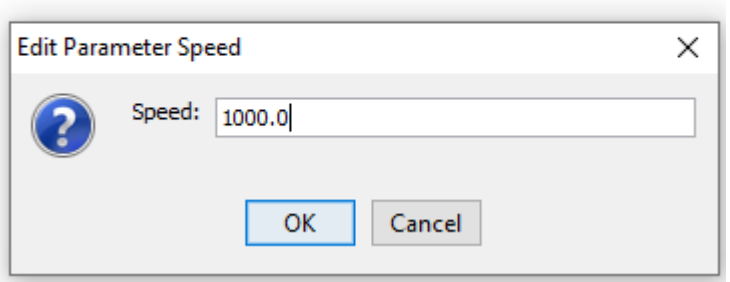
1. Drag-n-Drop the parameter from Library Folder **Model Setup >Parameter** ('parameter=') into an open Block Diagram Editor window.



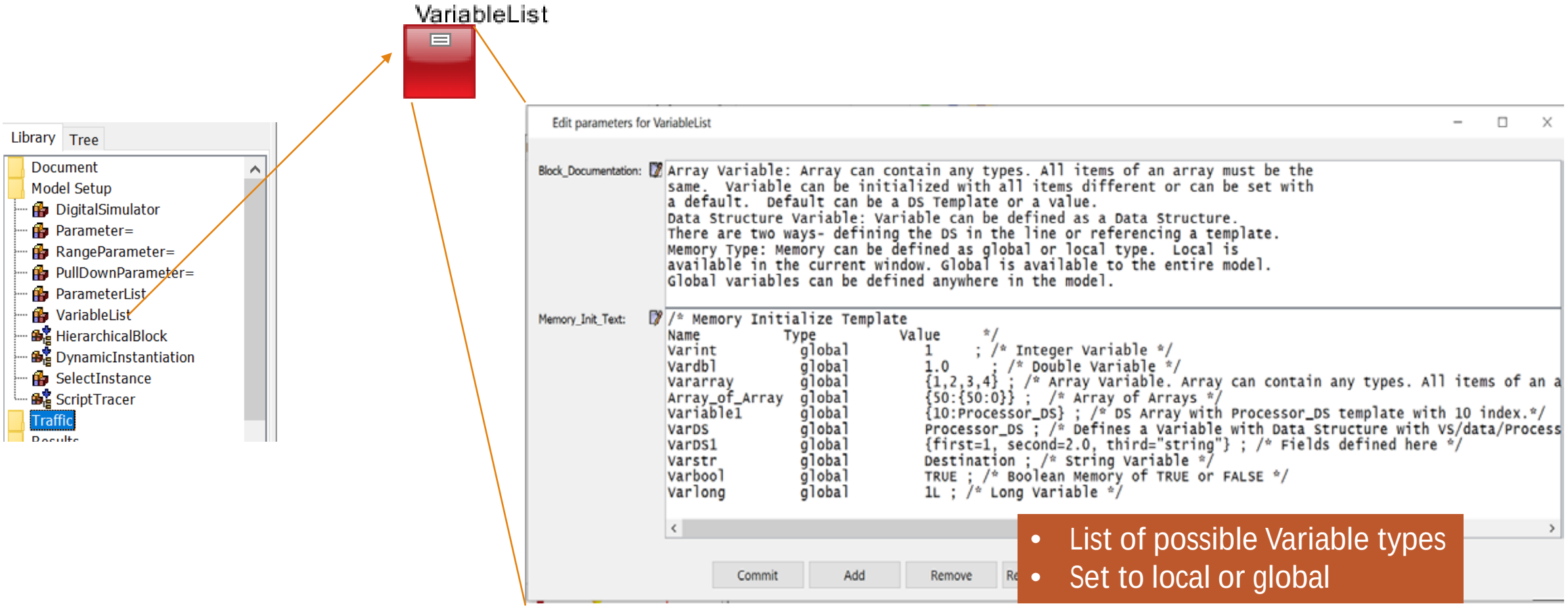
2. Right-click to select **Customize Name** of parameter & enter a name. Name must be unique, else BDE will generate exception.



3. Double click the new parameter name to set the value of the parameter.

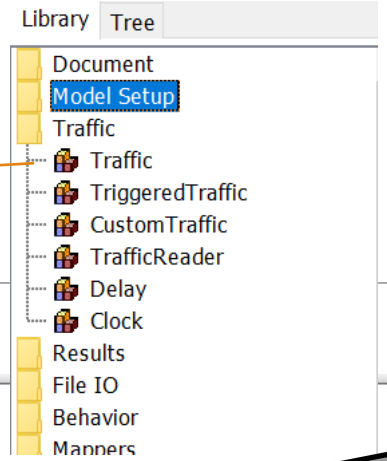


Variables

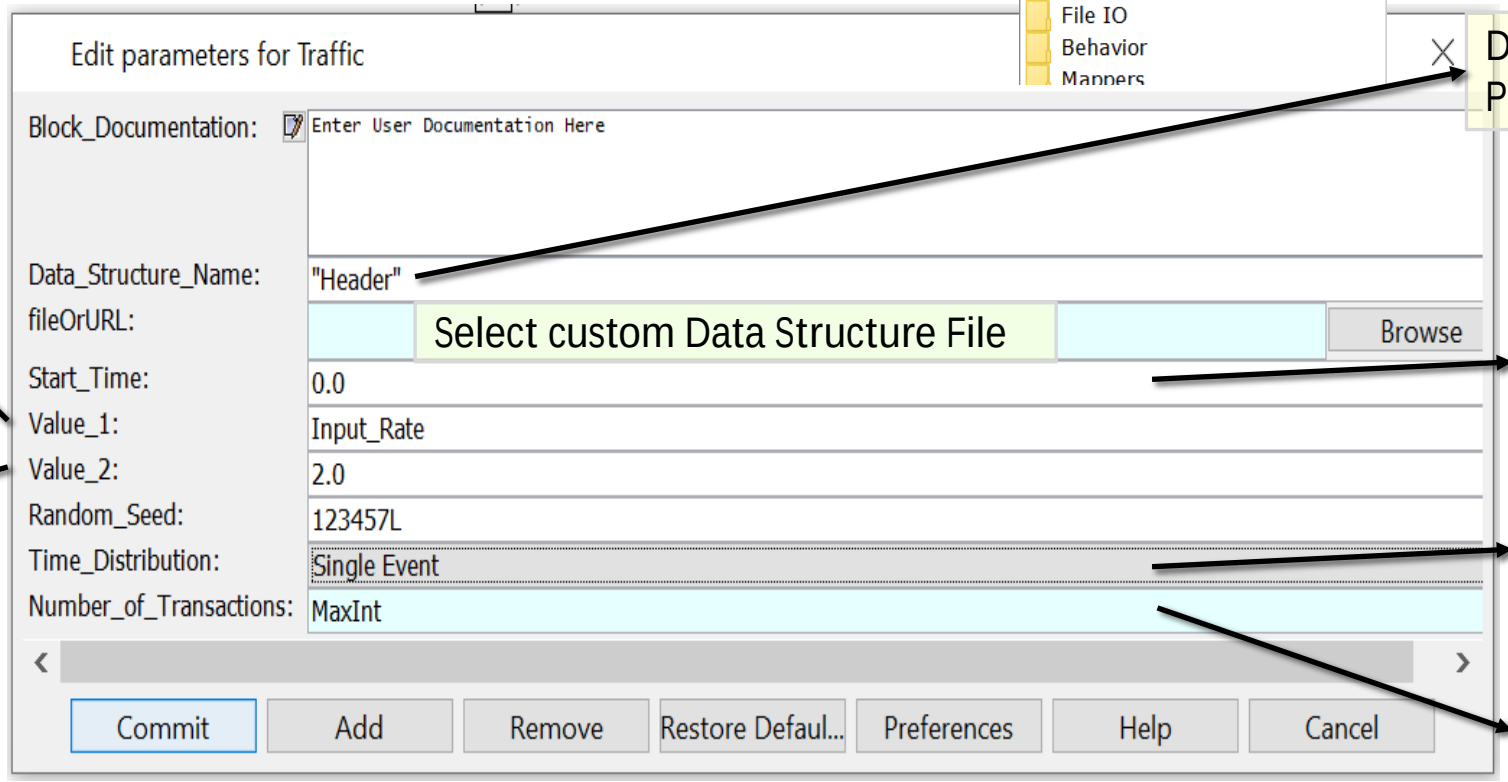


- List of possible Variable types
- Set to local or global

Traffic



- Double click to configure



Default:- Header, Pulse Processor_DS and Task_Class

Select custom Data Structure File

Mean/Min Value

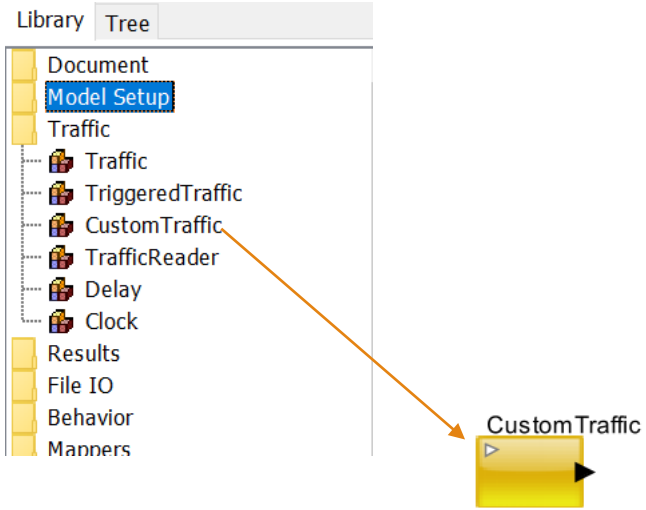
Max/Std Dev Value

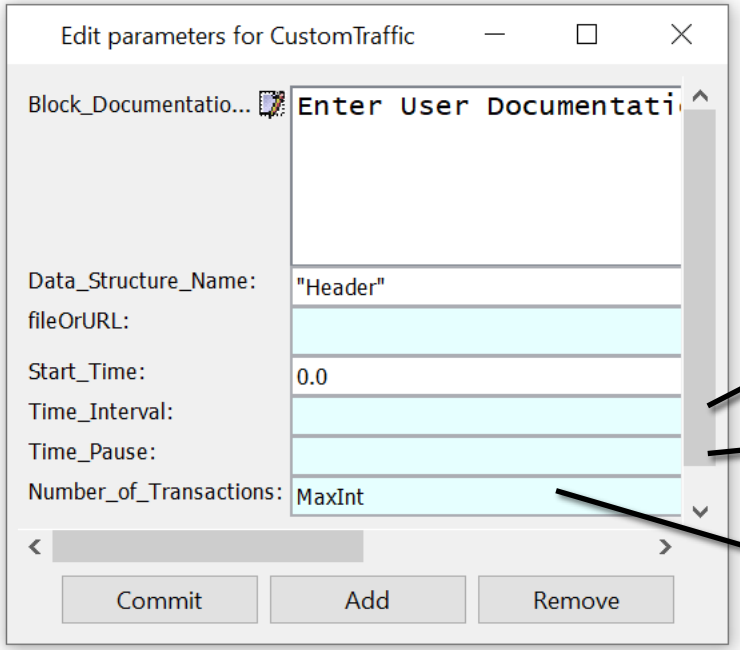
Delay to start output

Select the "Time_Distribution" according to design

Restrict the number of transactions

Custom Traffic





Time_Interval is period of traffic

Time_Pause is quiet period between transfer range

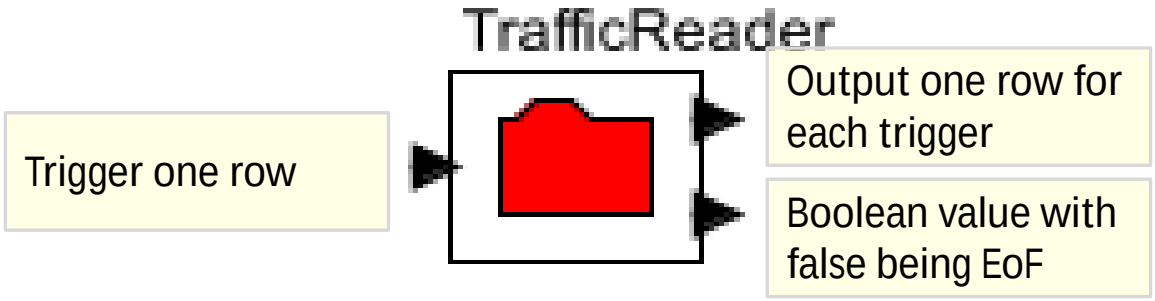
Number_of_Transaction during each Time_Interval

Trace File Input

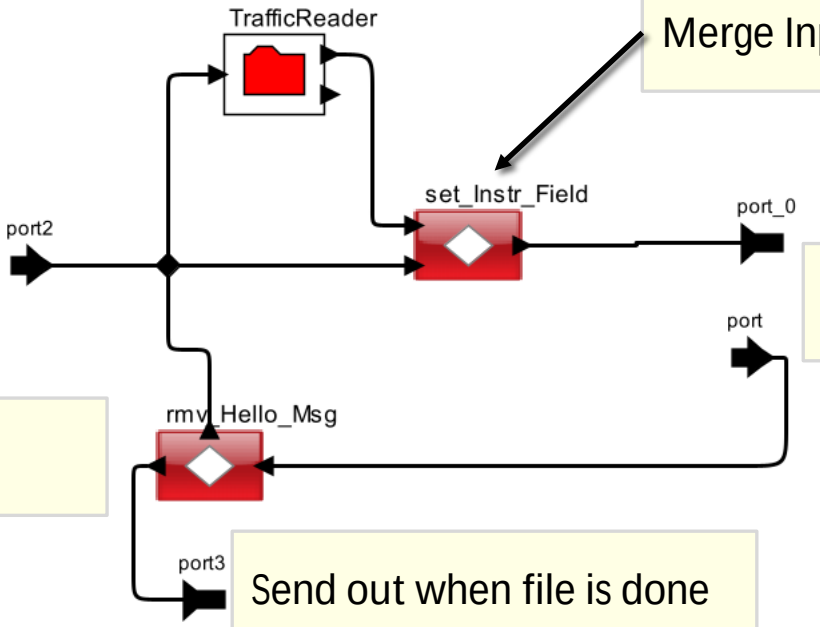
File Format- csv file only

- First line are the column names
- Data type per column
- Values of any data type

```
I_Cache_Address,A_Instruction,D_Cache_Address
array,array,array
{"0x1044c","0x10450","0x10454"},{"mov","mov","ldr"},{"0x00","0x00","0xbefffe50"}
```



Using Traffic Reader Block



Merge Input and Trace row

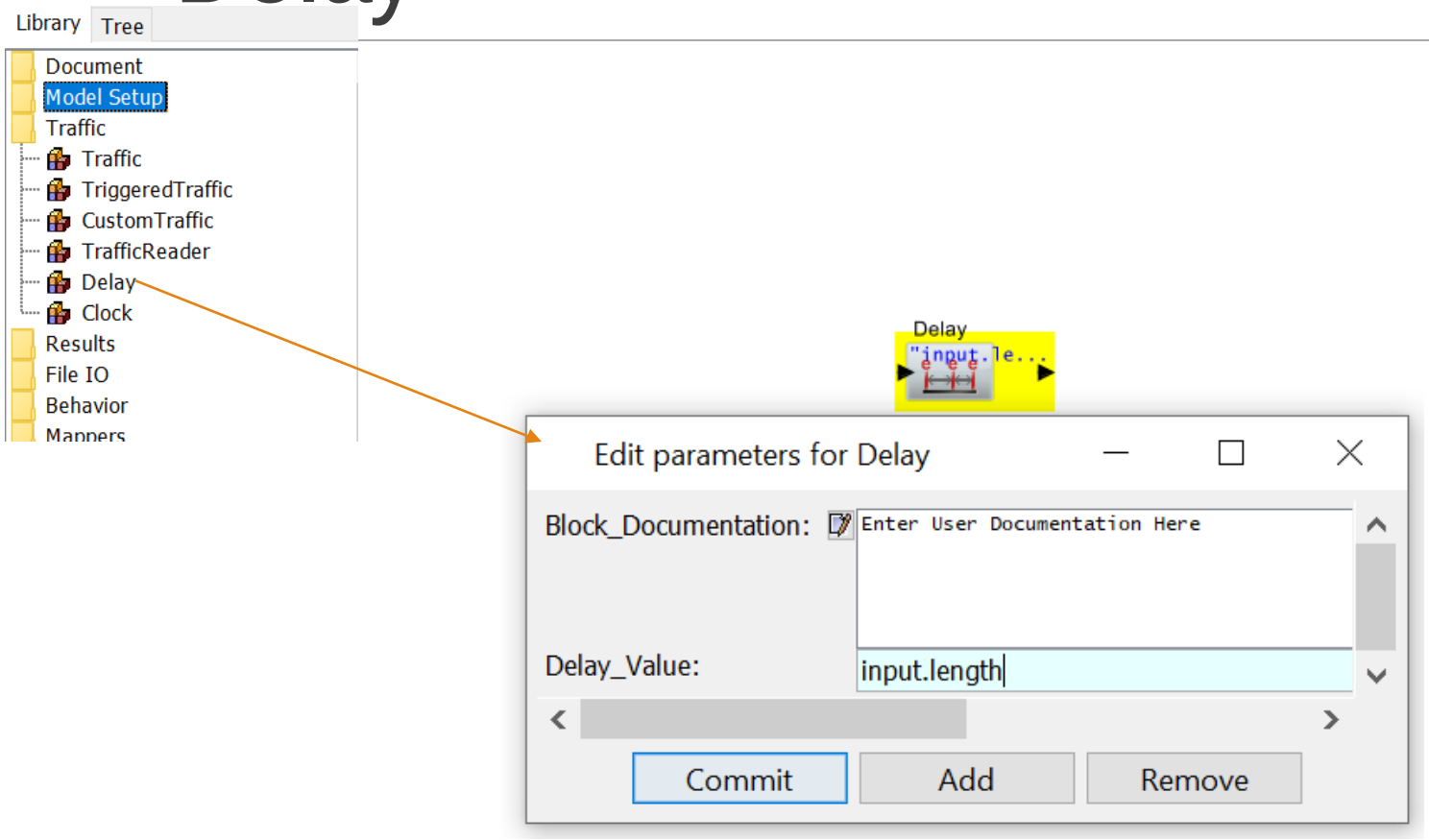
Input trigger, Use Processor_DS for Hardware

To and return from processor, bus or memory

Loop until file is empty

Send out when file is done

Delay

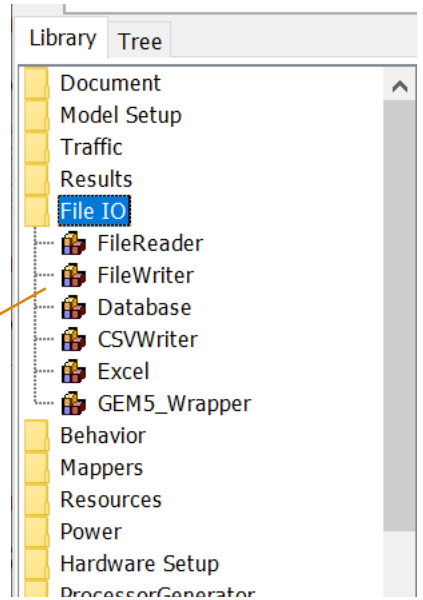


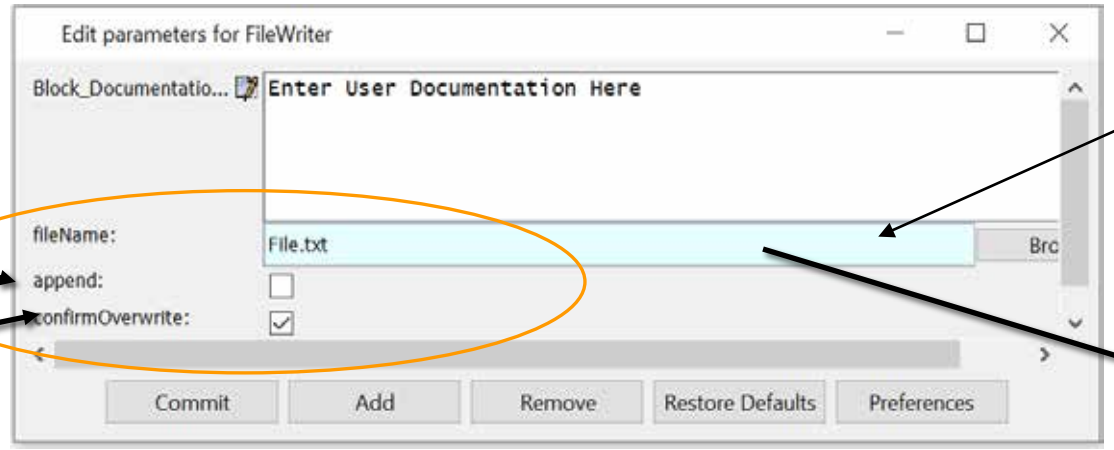
Input.FieldName
Variable
Parameter
RegEx expression
Double value

File I/O

Any data value

FileWriter





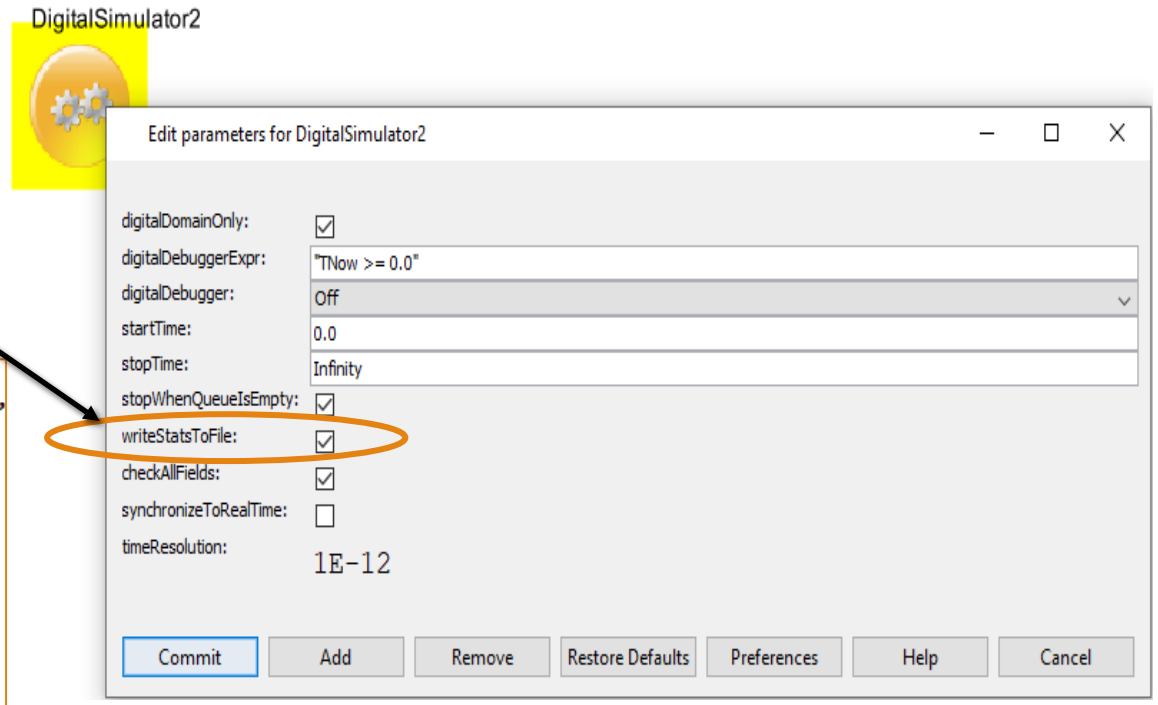
Add to existing file

Overwrite current content

fileName to save in model folder
Using Selector button for full path

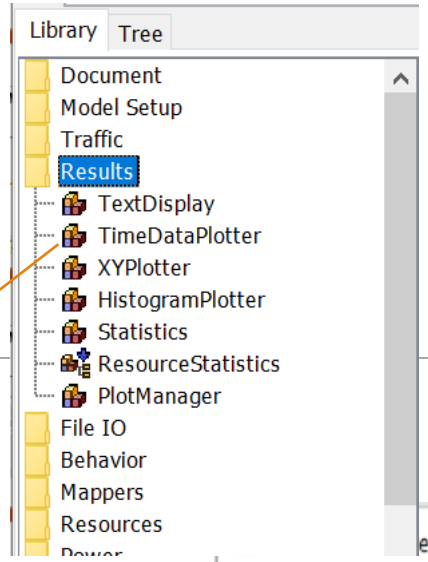
Write Stats To File

Select in Digital Simulator




```
Queue_Statistics                6.00000000000 sec
{BLOCK                        = "SR_SrExtend_example.SystemResource_Extend",
DELTA                         = 0.0,
DS_NAME                       = "Queue_Common_Stats",
ID                            = 1,
INDEX                         = 0,
Number_Entered                = 7,
Number_Exited                 = 1,
Number_Rejected               = 0,
Occupancy_Max                 = 6.0,
Occupancy_Mean                = 3.7777777777778,
Occupancy_Min                 = 1.0,
Occupancy_StDev               = 1.4740554623802,
Queue_Number                  = 1,
TIME                          = 6.0,
Total_Delay_Max               = 4.0,
Total_Delay_Mean              = 4.0,
Total_Delay_Min               = 4.0,
Total_Delay_StDev             = 0.0,
Utilization_Mean              = 0.0}
```

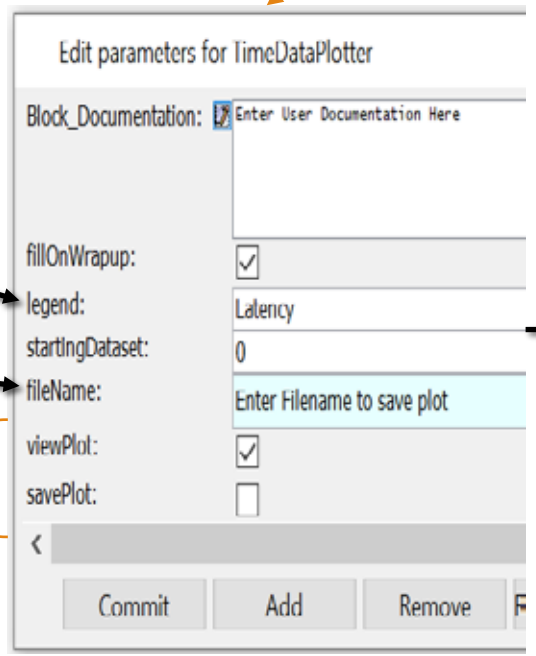
TimeData Plotter

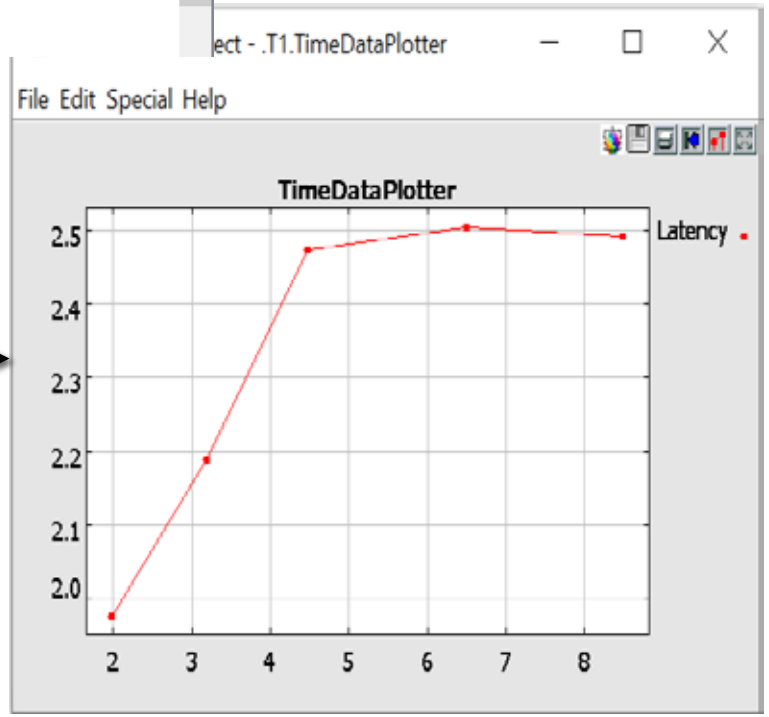


Name for Datasets in order of connection

FileName required for savePlot. Extension .txt

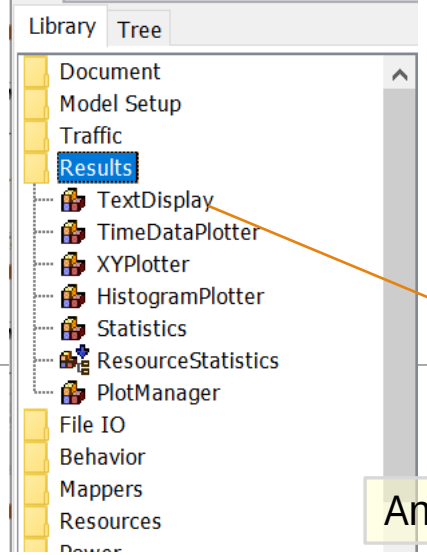
Select View or Save file
For large files, use Save

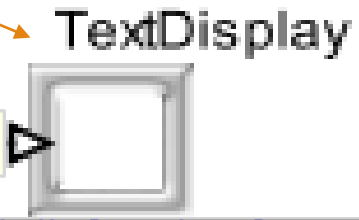




Return

Text Display



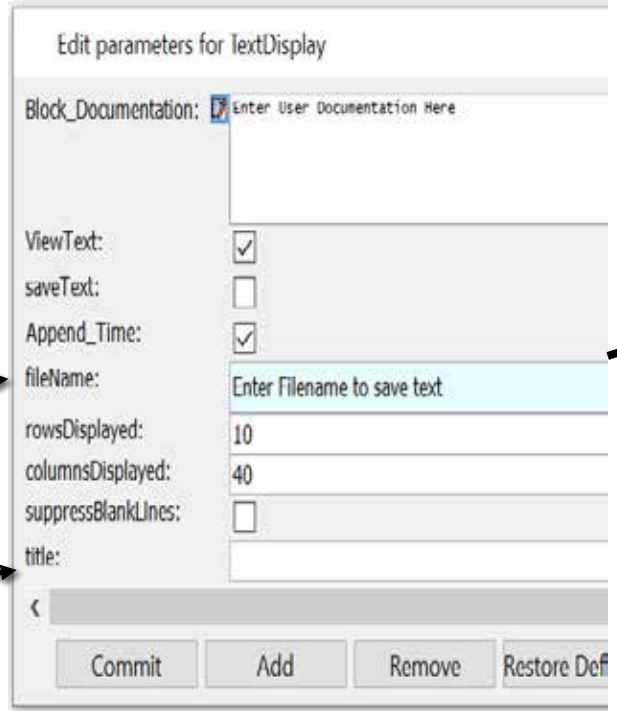


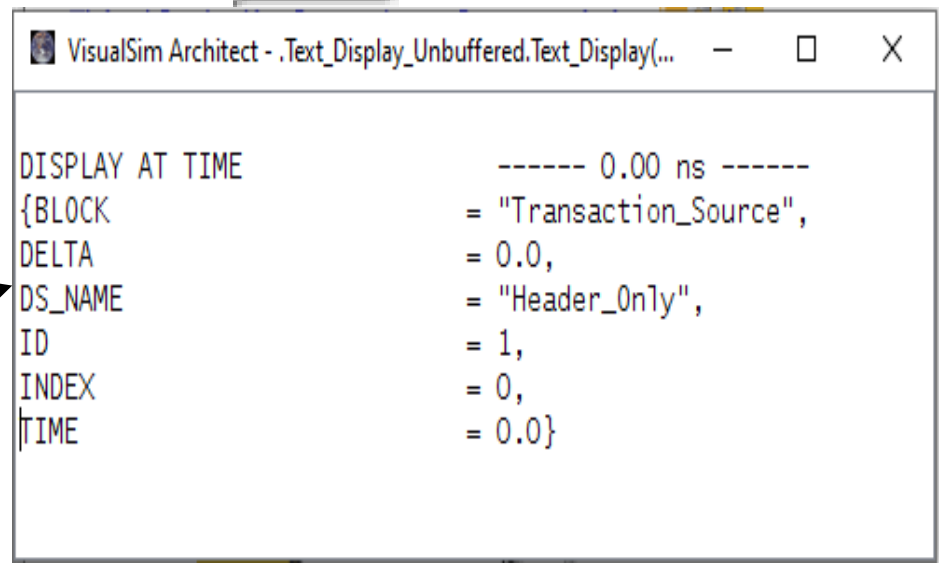
Any data type

Select View or Save file
For large files, use Save

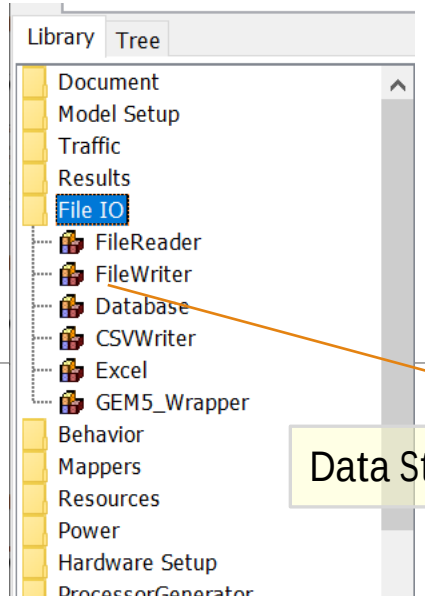
FileName required for savePlot. Extension .txt

Display in Top bar





Database



Data Structure

Matched Row /Array of rows

No match return Data Structure

LinkingName access this Database from other database and RegEx

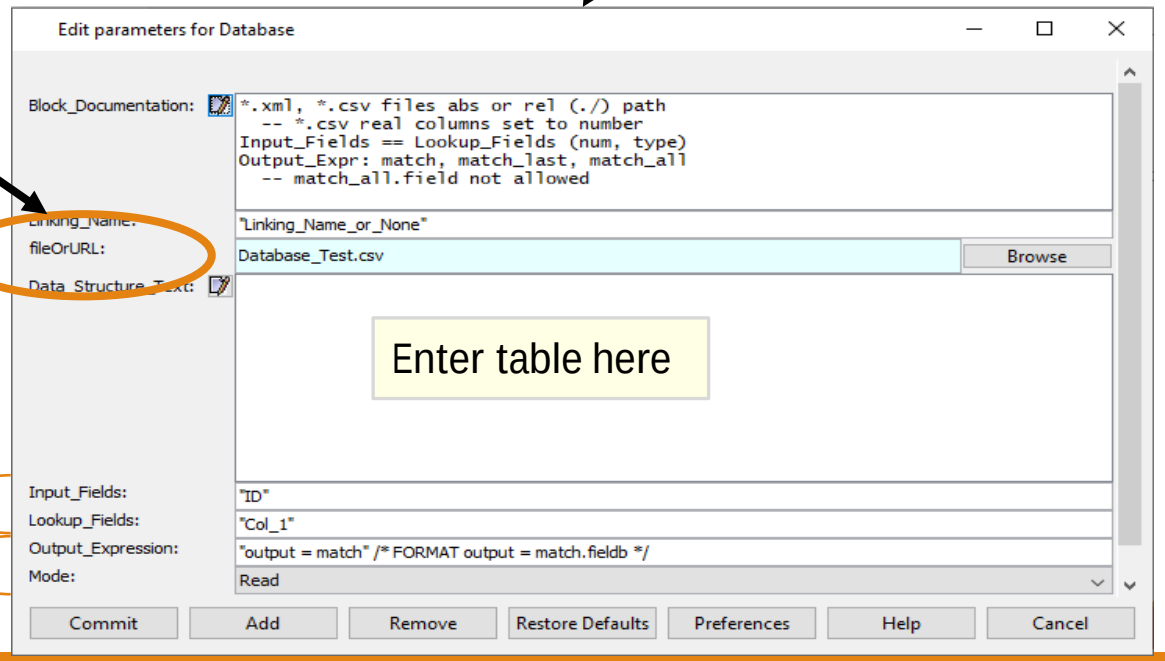
File name+path to csv file
Use for big file

Input DS fields to compare

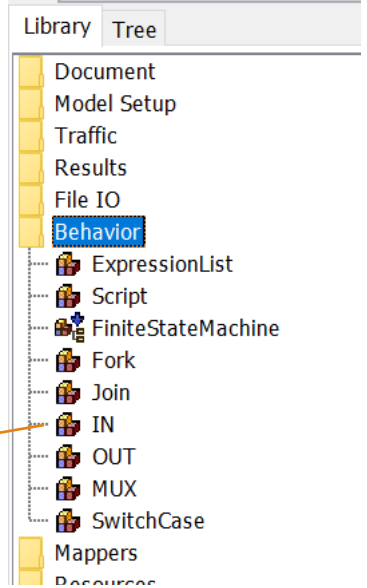
Database columns to compare

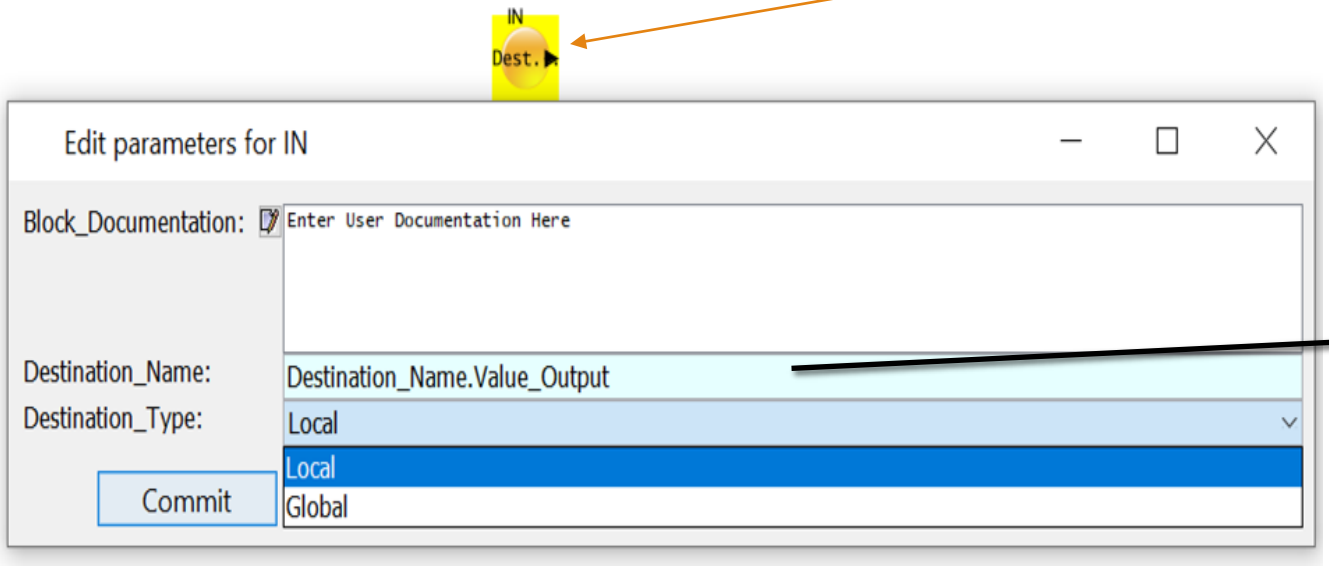
Type of output

Read/Write/Erase



In

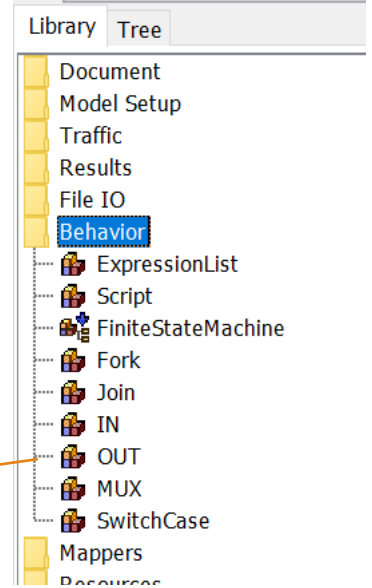


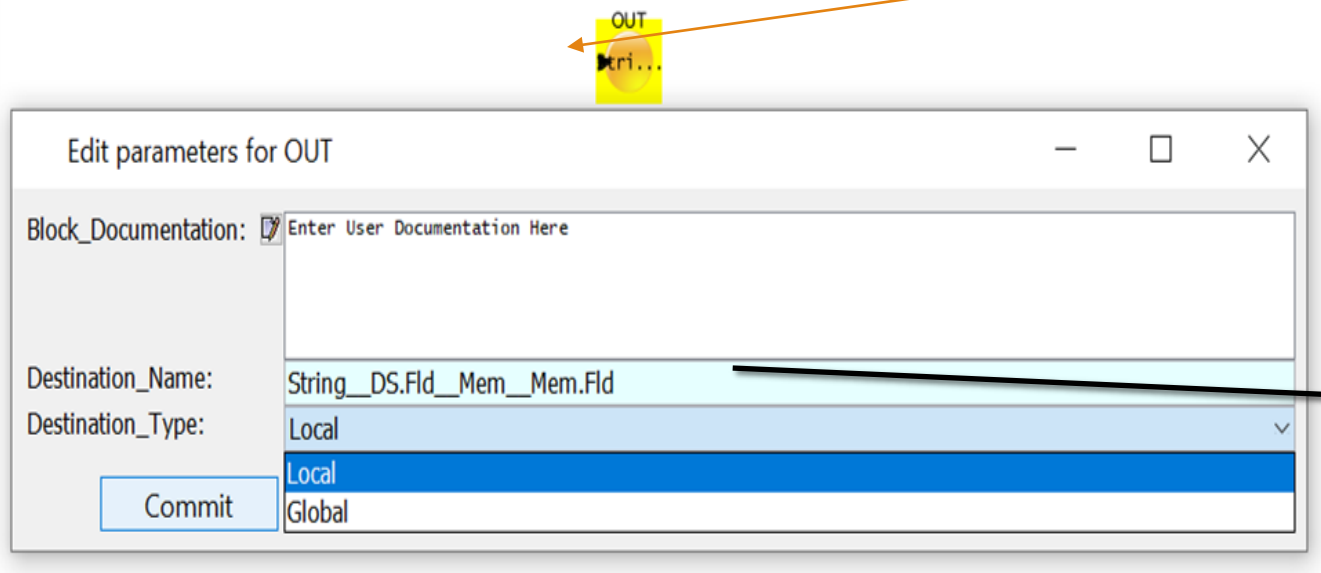


Value is DS, DS field, variable or parameter
Destination can be parameter or string
If no '.value_output', send incoming value

Return

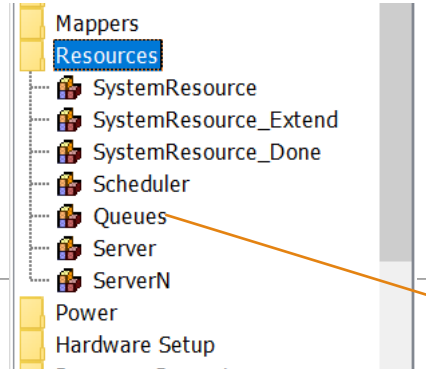
Out

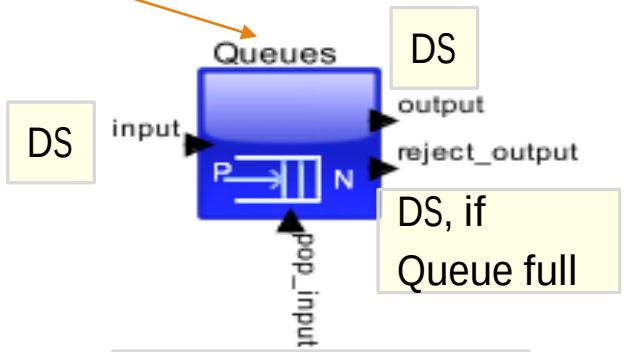




Concatenated string of string DS, DS field, variable and parameter.

Queue





Edit parameters for Queues

Block_Documentation:	Enter User Documentation Here
Block_Name:	"Queue"
Queue_Number_Field:	input.queue
Priority_Field:	input.priority
Max_Queue_Length:	30
Number_of_Queues:	1
Initial_Queue_State:	First_Token_Flow_Through
Queue_Reject_Mechanism:	Incoming_Token_Rejected
Queue_Type:	FIFO

Commit | Add | Remove | Res

Field, parameter, Variable or RegEx

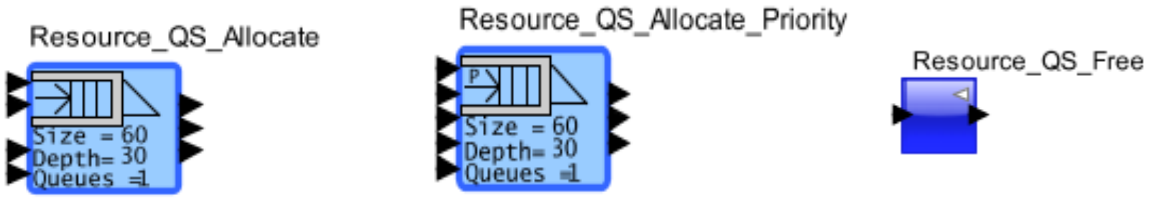
Priority to reorder queue. Field, parameter, Variable or RegEx

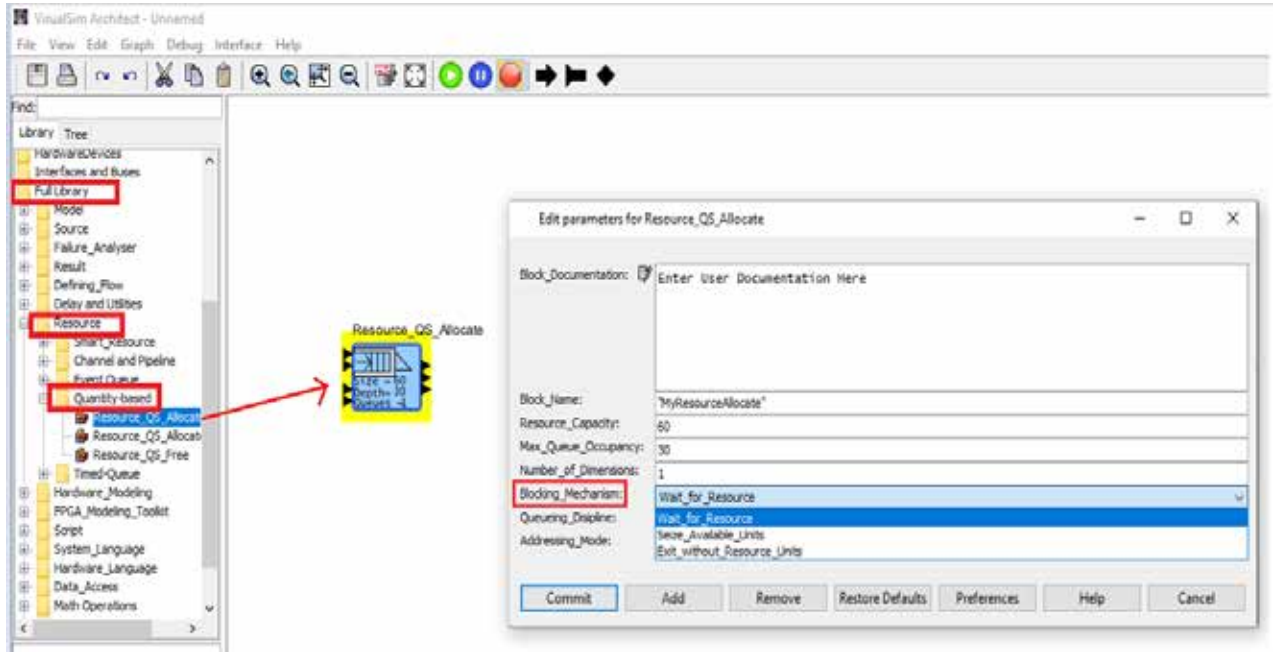
Queue organization

Queue organization

When Queue is Empty: Send now if prior DS received pop after (or) First token needs Pop

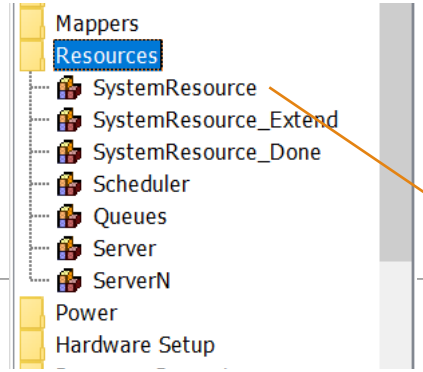
Quantity Based Resources





- *Queue_Number_Field* selects the queue
- Queue is ordered based on the *Priority field*
- Queues the data in FIFO or LIFO order according to the *Queue_Type* selected
- If Indexed, resource amt must be sequential
- Blocking mechanism determines to fulfill request

System Resource



Edit parameters for SystemResource

Mapper uses this name

Block_Documentation:	Enter User Documentation Here
Resource_Name:	"CPU"
Next_Resource:	"Fld_Name_or_String_or_None"
Task_Context_Switch_Time:	0.0
Round_Robin_Time_Slice:	1.0E-3
Clock_Rate_Mhz:	500.0
Max_Scheduler_Length:	30
Time_Type:	Relative Time
Scheduler_Type:	Scheduler_FCFS
Add_Scheduler_Times_to_DS:	Scheduler_FCFS

Commit Add

- Scheduler_FCFS
- FCFS + Preempt
- Scheduler_RR
- Scheduler_User_1
- Scheduler_User_2

Time between tasks including preempt

Time the scheduler will devote to each task for Round Robin

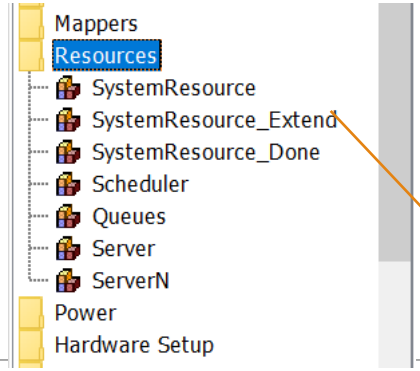
Used with Time_Type- clock

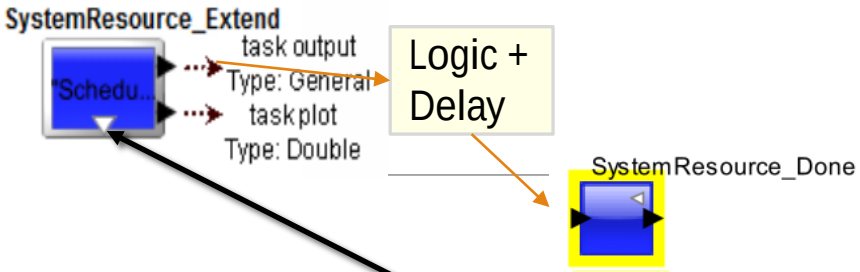
Relative- Mapper time is delay
Clock- Mapper time is cycles/Clock_Rate_Mhz

Set Scheduler from list
Program custom schedulers in Java

Arrives virtual From Mapper

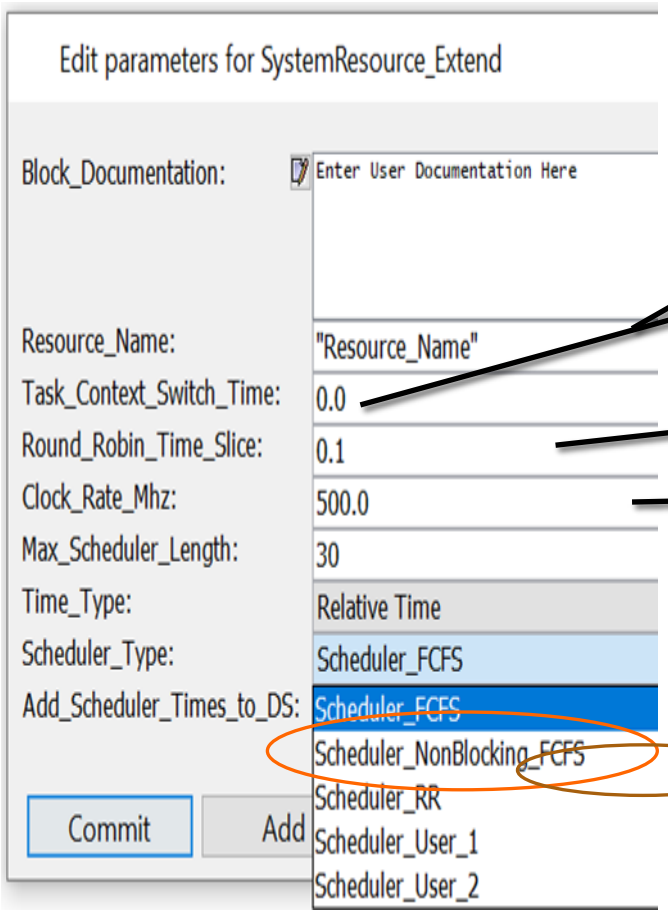
System Resource Extend





Logic + Delay

Arrives virtual From Mapper



Mapper uses this name

Time between tasks

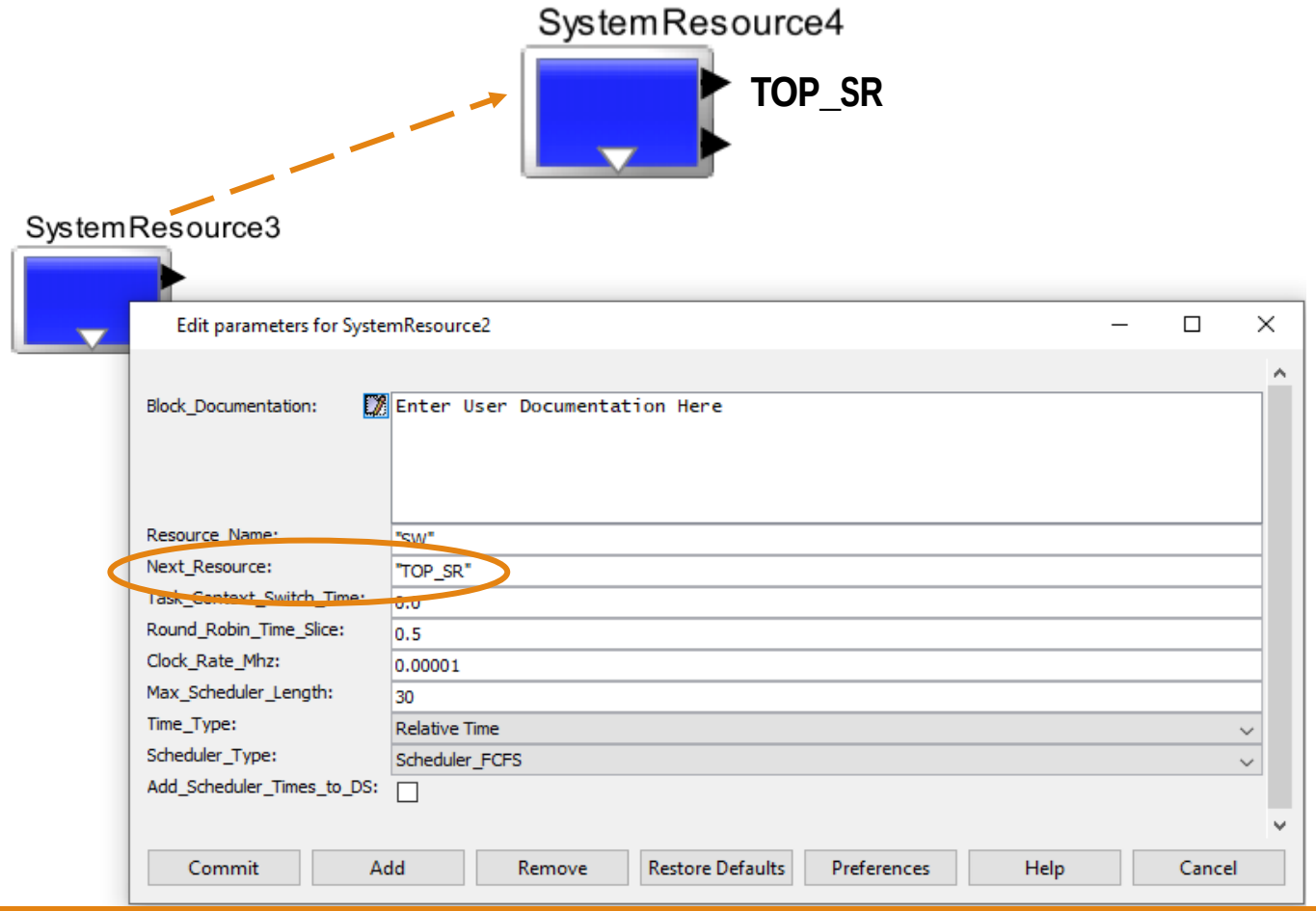
Time the scheduler will devote to each task for Round Robin

Used with Time_Type- clock

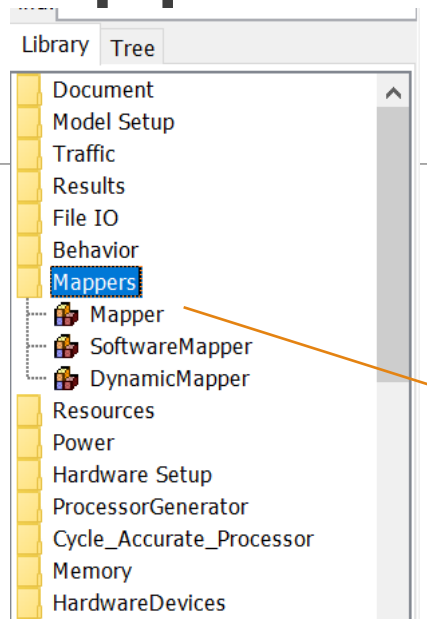
Relative- Mapper time is delay
Clock- Mapper time is cycles/Clock_Rate_Mhz

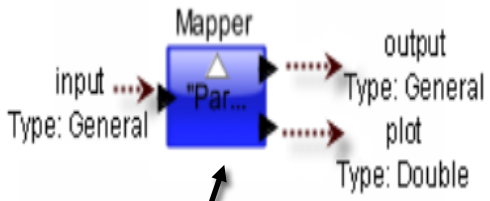
Non-blocking means that next DS is delayed and sent out immediately without waiting for the previous to reach Done block

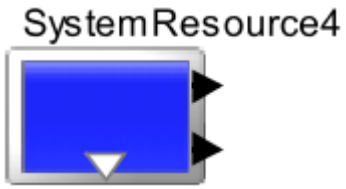
Hierarchical System_Resource



Mapper



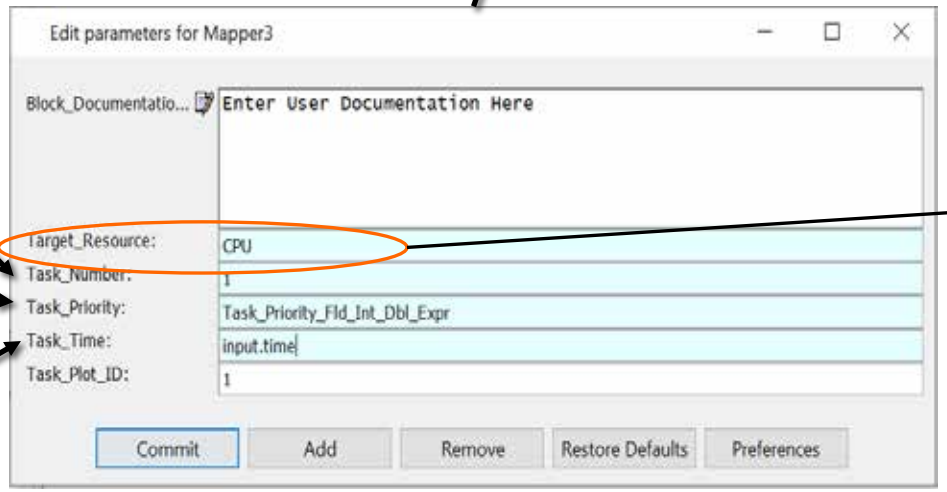


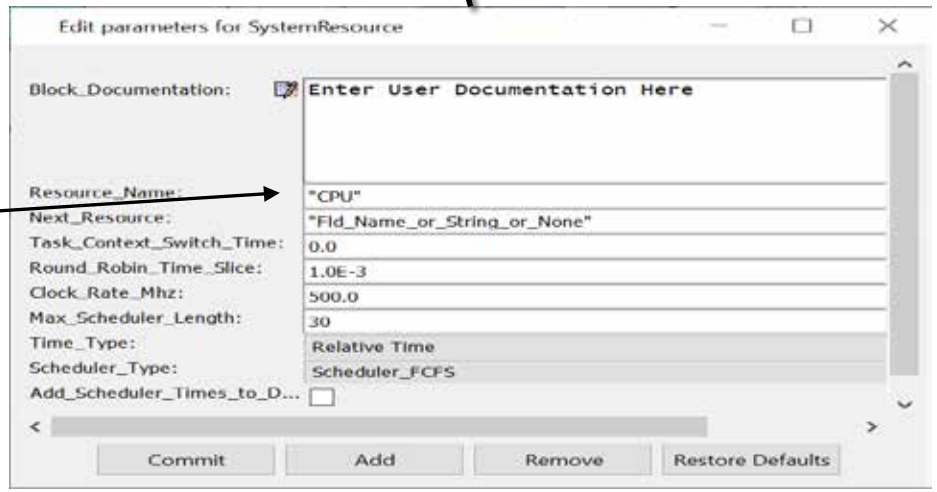


Unique ID for Plotting

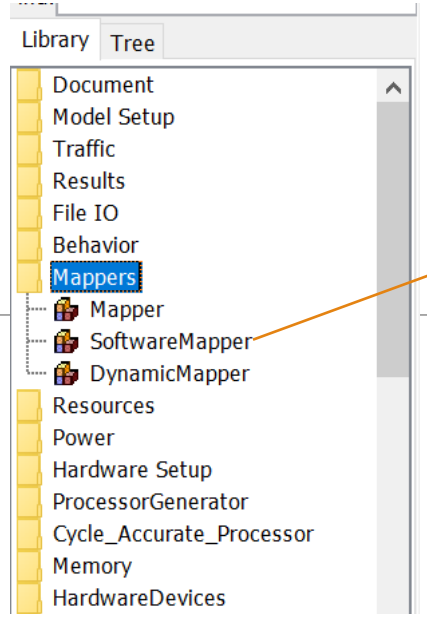
Reorder queue

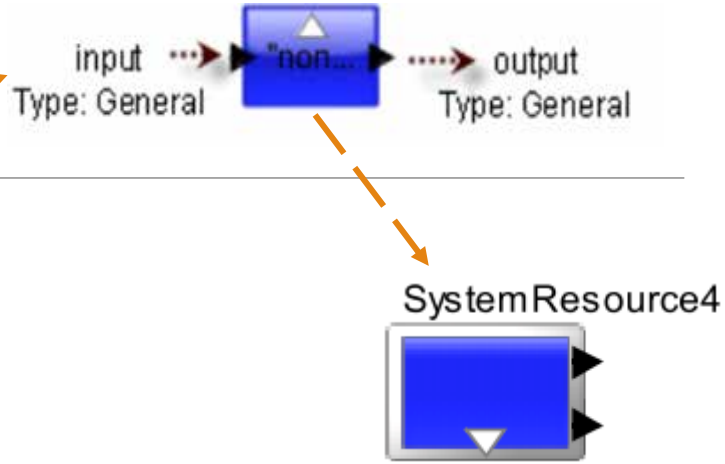
Delay in resource





Software Mapper



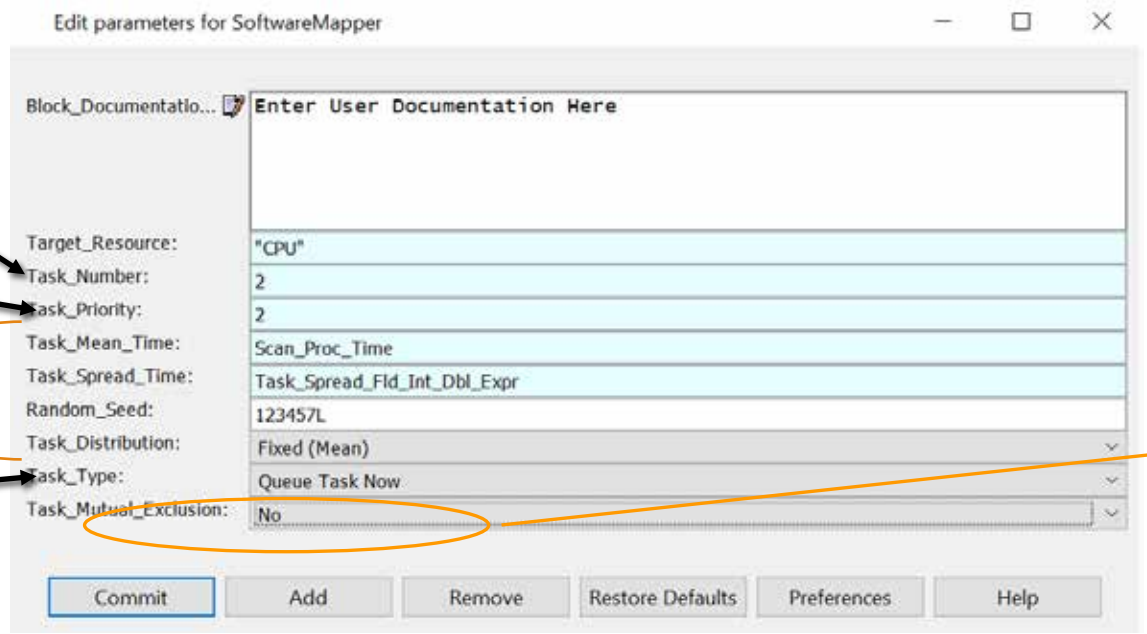


Unique ID for Plotting

Reorder queue

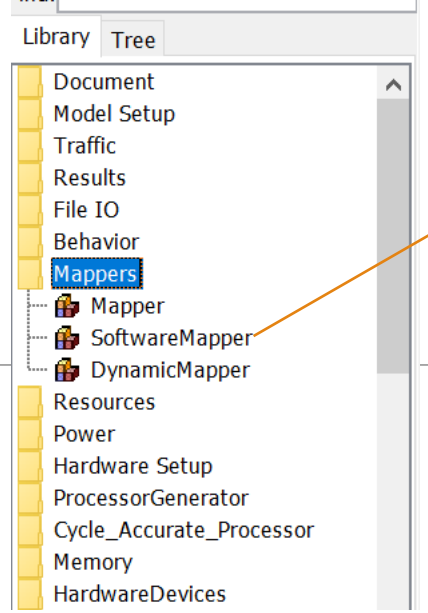
Distribution for Delay

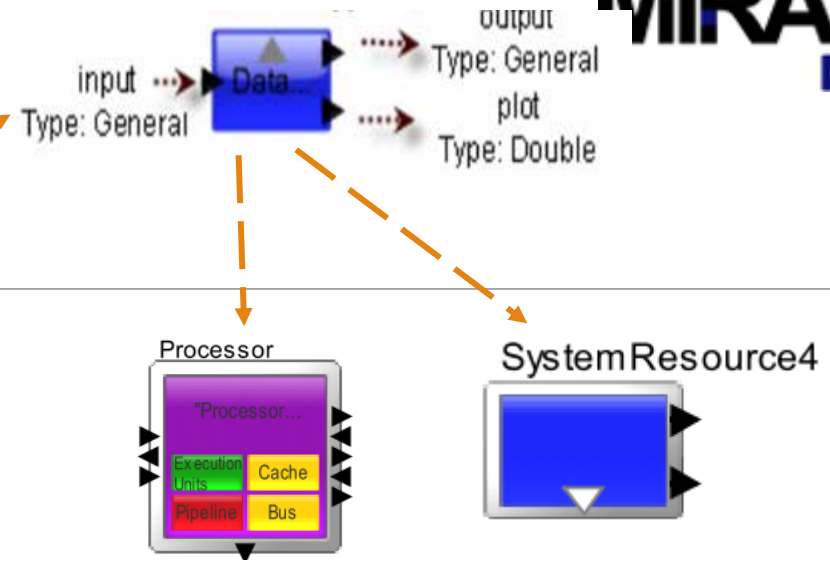
Queue in Mapper/SR



Lock out all other tasks from preempting this Task at the SystemResource

Dynamic Mapper





Unique Name

Match Task_Name to get below field values

Array for Processor

For Plotting

Reorder queue

Delay in SystemResource

Edit parameters for DynamicMapper

Block_Documentatio...	Enter User Documentation Here
Block_Name:	Block_Name
Database_Lookup:	Database_Name_or_Fld_or_Expr_or_None
Task_Name:	Task_Name_Fld_Expr
Target_Resource:	Task_Destination_Fld_Expr
Task_Instruction:	Task_Instruction_Fld_Expr_None
Task_Plot_ID:	1
Task_Number:	Task_Number_Fld_Int_Dbl_Expr
Task_Priority:	Task_Priority_Fld_Int_Dbl_Expr
Task_Time:	Task_Time_Fld_Int_Dbl_Expr

Database LinkingName

Processor or SystemResource

VISUALSIM TRAINING