LISTEN.
THINK.
SOLVE.®

Arena®

VARIABLES GUIDE

PUBLICATION ARENAV-RM001I-EN-P–March 2009
Supersedes Publication ARENAV-RM001H-EN-P

Allen-Bradley • Rockwell Software

Rockwell Automation

Contact Rockwell

Customer Support Telephone — 1.440.646.3434
Online Support — http://www.rockwellautomation.com/support/

Version: 13.00.00 (CPR 9 SR 1)
Modified: June 9, 2009 10:34:46 AM

Contents

Variables

Introduction

This guide contains a comprehensive overview of the predefined variables that can be used or referenced in all Arena products. Some of the variables may not be available in every Arena product.

The variables described in this manual can be used in a variety of ways. They can be useful when building your model; for example, you might use a Decide module and follow one path of logic if the number of entities in the queue called "WaitQ" is greater than 10. This is done by putting the expression NQ(WaitQ)>10 in the Decide condition. Variables are also useful when animating your model; for example, you might want to animate the number of active transporters called "Trucks" on-screen. To do this, animate a variable and list MT(Trucks) as the expression to animate. Finally, variables are often necessary when collecting statistics. For example, if you want to perform output analysis on the average value of a tally called "Time in System," use the Statistic module and record the tally expression TAVG(Time in System).

Many of the predefined variables in Arena are user-assignable. This means that you can change them (e.g., with an Assign module) throughout the simulation run. The descriptions of the variables and the summary charts in this guide note whether or not a variable is user-assignable.

Attributes and entity-related variables

Arena provides a set of pre-defined, special-purpose attributes that store information for each entity.In addition to the pre-defined attributes, Arena also allows for user-defined, real- or string-value, general-purpose attributes. These are defined by you as you make attribute assignments in your model (e.g., with the Assign module).

The variables listed in this section give access to these attributes and other entity-related information. Many of the variables reference the active entity by default; or you may reference any entity that is currently valid in the simulation via a specific entity number. This *Entity Number* argument corresponds to the value of the variable *IDENT* for some entity that is active in the simulation.

Many of these variables also require an *Attribute Number* argument. This refers to the construct number of the attribute (as defined in the Attributes module from the Elements panel). The *Attribute Number* argument may be entered as a constant, an expression, or as NSYM(Attribute Name).

Note: All defined simulation constructs in Arena are internally numbered. If you do not explicitly number them yourself using the associated module from the Elements panel, Arena will automatically provide an internal numbering (the more common approach). These constructs can be referenced by either their name or their number. If you choose to reference any by its number,

it is then necessary to number the construct explicitly (so that you can rely on specific ordering). If you do not specifically number them, and therefore do not know the construct number, you can enter the construct name in a field where its number is required by using NSYM(Construct Name).

For example, NSYM(Attribute Name) returns the associated number of the attribute Attribute Name, NSYM(Station Name) returns the station number associated with that name, and NSYM(Intersection Name) returns the intersection number associated with that name.

General attributes variables

Attribute Name [(Index 1, Index 2)]—General-purpose entity attribute. You can define as many general-purpose attributes as are needed in your model. The attribute name itself may then be used to access or assign values of these attributes. If the attribute is defined as an indexed array (in an Attributes module from the Elements panel), the appropriate number of index values must be given. General-purpose attributes can store real or string values. An attribute's data type can be specified using the Attributes module from the Elements panel.

A (Attribute Number [, Entity Number])—General-purpose entity attribute. This is an alternate means of accessing attributes when you've defined attribute numbers (in the Attributes module from the Elements panel). Each individual attribute in an indexed group is given a unique attribute number.

Entity.Type [(Entity Number)]—Entity type attribute. This attribute refers to one of the types (or names) of entities defined in the Entities element. Entity type is used to set initial values for the entity picture and the cost attributes. It is also used for grouping entity statistics (e.g., each entity's statistics will be reported as part of all statistics for entities of the same type).

Entity.Picture [(Entity Number)]—Entity animation attribute. Models with animation use this attribute to store the value of an entity's graphical picture. This value is used to determine which animation picture is used to depict each entity. (Note: The use of the *Picture* keyword is still supported for old models.)

Entity.SerialNumber—Entity serial number. This attribute is set to a unique value each time an entity is created via the Create module. If that entity is ever duplicated, the duplicate will have the same value of Entity.SerialNumber as the original. This attribute is particularly useful to identify related entities when combining or synchronizing entities in a Batch module. (Please note that when the *Entity Number* argument is required for a variable listed in this guide, it is referring to the *IDENT* value of the entity, and not the *Entity.SerialNumber* value. See IDENT for more information.)

Entity.Jobstep [(Entity Number)]—Entity jobstep (sequence index) attribute. This attribute is used as the index into the entity's sequence when an entity is transferred by a

sequence. Each time a sequential transfer occurs, the jobstep attribute automatically is incremented by one unless the sequence changes it to a specific index value by the NEXT option. The *IS[(Entity Number)]* attribute is an alternate means of accessing the Entity.Jobstep value.

Entity.Sequence [(Entity Number)]—Entity sequence attribute. This attribute determines the sequence number to follow when an entity is transferred to the next station by a sequence. The model must explicitly assign a value to this attribute; Arena does not change or initialize it automatically. The *NS[(Entity Number)]* attribute is an alternate means of accessing the Entity.Sequence value.

Entity.Station [(Entity Number)]—Entity station location attribute. This attribute is used to store the entity's station location or destination. When an entity is transferred via any route, transport, or convey action, its station attribute automatically is assigned the destination station number. It also is used to determine the location of access to the material-handling device. The *M[(Entity Number)]* attribute is an alternate means of accessing the Entity.Station value.

Entity.CurrentStation—Entity current station location attribute. This attribute is used to store the entity's current station location. It will return the station number an entity is currently located in or 0 if the entity is not currently in a station. Unlike the *Entity.Station* (M) attribute, Entity.Current Station is *not* user-assignable. Entity.Current Station is automatically updated whenever an entity enters a station (e.g., executes a STATION block) or exits a station (e.g., transfers out of a station or disposes). If an entity is a member of a group, its Entity.CurrentStation is always the same station that the group's representative is currently in.

Entity.PlannedStation—Entity next planned station location attribute. This attribute is used to store the entity's next station visitation as defined in the entity's Entity.Sequence (Sequence module in Advanced Transfer panel). Unlike the Entity.Station (M) attribute, Entity.PlannedStation is not user-assignable.

If the entity has a valid Entity.Sequence assigned, then Entity.PlannedStation stores the number of the station associated with the next jobstep in the sequence. It returns a zero if the entity is at the end of its sequence or if Entity.Sequence is undefined. Entity.PlannedStation is automatically updated whenever the attributes Entity.Sequence (NS) or Entity.JobStep (IS) are changed, as well as whenever the entity enters a station.

Note: In a transfer block (i.e., TRANSPORT, CONVEY, ROUTE), if the destination is specified as "Sequential," then SIMAN first examines the entity's Entity.Sequence to retrieve the next station visitation. If Entity.Sequence is undefined, then the entity is transferred to Entity.PlannedStation if that attribute has a non-zero value.

Time attribute variables

Time attributes are created based on the settings of the statistics options on the Project Parameters page of Arena's **Run > Setup** dialog. If either process statistics or entity statistics are enabled, these attributes will be created automatically. If neither of these items is enabled, then these attributes will not be available for use. None of these attributes are user-assignable—they are all updated automatically to store information about entity times.

Entity.CreateTime—Entity creation time attribute. This attribute is set to the system time (TNOW) when an entity is first created. This attribute is most often used in calculating time in system or cycle time.

Entity.StartTime—Entity start time attribute. This stores the time that an entity started its current activity. If you look at an active entity, this will always be the same as TNOW. If you look at an entity that is currently in a delay or queue, this will be the time the entity started that delay or entered the queue. This is used internally for calculating time durations.

Entity.VATime—Entity value-added time attribute. This stores the total time accumulated in processes and delays designated as Value Added. When the entity statistics are recorded (typically on dispose), this attribute is used to determine the total value-added time accrued by this entity.

Entity.NVATime—Entity non-value-added time attribute. This stores the total time accumulated in processes and delays designated as Non-Value-Added. When the entity statistics are recorded (typically on dispose), this attribute is used to determine the total non-value-added time accrued by this entity.

Entity.WaitTime—Entity waiting time attribute. This stores the total time accumulated in queues (waiting areas) as well as processes and delays designated as Wait. When the entity statistics are recorded (typically on dispose), this attribute is used to determine the waiting time accrued by this entity.

Entity.TranTime—Entity transfer time attribute. This stores the total time accumulated in transfers (transporters and conveyors) as well as processes and delays designated as Transfer. When the entity statistics are recorded (typically on dispose), this attribute is used to determine the total transfer time accrued by this entity.

Entity.OtherTime—Entity other time attribute. This stores the total time accumulated in processes and delays designated as Other or with no category designated. When the entity statistics are recorded (typically on dispose), this attribute is used to determine the total other time accrued by this entity. This category can be used if you want to record time separate from one of the four standard categories above.

Cost attribute variables

Cost attributes are created based on the settings of the statistics options on the Project Parameters page of Arena's **Run > Setup** dialog. If costing statistics are enabled and either process statistics or entity statistics are enabled, these attributes will be created automatically. If these options are not enabled, then the cost attributes will not be available for use. Of these attributes, only Entity.HoldCostRate is user-assignable.

The cost attributes are all initialized to the values specified in the Entities module when an entity is first created. Thereafter, they are updated automatically to store information about entity costs.

Entity costs consist of three components: holding costs, resource ownership costs, and resource usage costs. Whenever there is a time delay, the busy cost rates for all resources owned (see ResCostRateGrp) are added to the Entity.HoldCostRate, and the sum is multiplied by the time duration. If resources are seized during the process, then any applicable resource usage costs will also be included. The resultant cost is then accrued to one of the following cost categories.

Entity.HoldCostRate—Entity holding cost rate. This attribute is an important part of all the cost calculations. If there is an inventory or holding cost associated with an entity, it should be specified here. Any time that the entity spends in the system waiting or in any type of process, it will incur costs based on its current holding cost rate. If multiple entities are combined, the representative automatically will be assigned the sum of the holding cost rates of its members. In addition, you may assign the holding cost rate to reflect changes in value at various stages of the process.

Entity.VACost—Entity value-added cost attribute. This stores the total costs accumulated in processes and delays designated as Value-Added. When the entity statistics are recorded (typically on dispose), this attribute is used to determine the total value-added costs accrued by this entity.

Entity.NVACost—Entity non-value-added cost attribute. This stores the total cost accumulated in processes and delays designated as Non-Value-Added. When the entity statistics are recorded (typically on dispose), this attribute is used to determine the total non-value-added cost accrued by this entity.

Entity.WaitCost—Entity waiting cost attribute. This stores the total cost accumulated in queues (waiting areas) as well as processes and delays designated as Wait. When the entity statistics are recorded (typically on dispose), this attribute is used to determine the waiting cost accrued by this entity.

Entity.TranCost—Entity transfer cost attribute. This stores the total cost accumulated in transfers (transporters and conveyors) as well as processes and delays designated as Transfer. When the entity statistics are recorded (typically on dispose), this attribute is used to determine the total transfer cost accrued by this entity.

Entity.OtherCost—Entity other cost attribute. This stores the total cost accumulated in processes and delays designated as Other or with no category designated. When the entity statistics are recorded (typically on dispose), this attribute is used to determine the total other cost accrued by this entity. This category can be used if you want to record cost separate from one of the four standard categories above.

Entity-type variables

Entity-type variables are related to the types or classes of entities defined by the Entity module (or Entities element). While each entity will have unique values for each of the attributes described above, the entity-type variables are common to all entities of the same type. (Note that entities with no specified entity type are assumed to be Entity Type 0 and will not have statistics collected for them.)

EntitiesIn (Entity Type)—Number of entities in. This variable stores the total number of entities of the specified type that have entered the system. Whenever an entity is created or its type is reassigned, EntitiesIn will be incremented.

EntitiesOut (Entity Type)—Number of entities out. This variable stores the total number of entities of the specified type that have left the system. Whenever an entity is disposed or its type is reassigned, EntitiesOut will be incremented.

EntitiesWIP (Entity Type)—Number of entities in process. This variable stores the total number of entities of the specified type that are currently in the system (Work In Process).

InitialPicture (Entity Type)—Initial picture. When an entity is initialized at a Create module, its Entity.Picture attribute will be assigned to this value based on its entity type.

InitialHoldCostRate (Entity Type)—Initial hold cost rate. When an entity is initialized, its Entity.HoldCostRate attribute will be assigned to this value based on its entity type.

InitialVACost (Entity Type)—Initial value-added cost. When an entity is initialized, its Entity.VACost attribute will be assigned to this value based on its entity type. This typically represents the cost or value associated with an incoming entity.

InitialNVACost (Entity Type)—Initial non-value-added cost. When an entity is initialized, its Entity.NVACost attribute will be assigned to this value based on its entity type. This typically represents the cost or value associated with an incoming entity.

InitialWaitCost (Entity Type)—Initial waiting cost. When an entity is initialized, its Entity.WaitCost attribute will be assigned to this value based on its entity type. This typically represents the cost or value associated with an incoming entity.

InitialTranCost (Entity Type)—Initial transfer cost. When an entity is initialized, its Entity.TranCost attribute will be assigned to this value based on its entity type. This typically represents the cost or value associated with an incoming entity.

InitialOtherCost (Entity Type)—Initial other cost. When an entity is initialized, its Entity.OtherCost attribute will be assigned to this value based on its entity type. This typically represents the cost or value associated with an incoming entity.

Group member variables

AG (Rank, Attribute Number)—Group member attribute. AG returns the value of general-purpose attribute *Attribute Number* of the entity at the specified *Rank* in the active entity's group. The function NSYM may be used to translate an attribute name into the desired *Attribute Number*.

ENTINGROUP (Rank [, Entity Number])—Grouped entity number. ENTINGROUP returns the entity number (i.e., IDENT value) of the entity at the specified *Rank* in the group of entity representative *Entity Number*.

GRPTYP [(Entity Number)]—Group type. When referencing the representative of a group formed at a Group module, GrpType returns 1 if it is a temporary group and 2 if it is a permanent group. If there is no group, then a 0 will be returned.

ISG (Rank)—Grouped entity jobstep attribute. This function returns the special-purpose jobstep (Entity.Jobstep or IS) attribute value of the entity at the specified *Rank* of the active entity's group.

MG (Rank)—Grouped entity station attribute. This function returns the special-purpose station (Entity.Station or M) attribute value of the entity at the specified *Rank* of the active entity's group.

NSG (Rank)—Grouped entity sequence attribute. This function returns the special-purpose sequence (Entity.Sequence or NS) attribute value of the entity at the specified *Rank* of the active entity's group.

NG [(Entity Number)]—Number of grouped entities. NG returns the number of entities in the group of representative *Entity Number*. If Entity Number is defaulted, NG returns the size of the active entity's group.

SAG (Attribute Number)—Sum of grouped entity attributes. SAG adds the values of the specified *Attribute Number* of all members of the active entity's group. The data type of the specified attribute must be numeric. The function NSYM may be used to translate an attribute name into the desired *Attribute Number*.

Other entity variables

ATTR (Attribute Number [, Index 1, Index 2])—Attribute value. ATTR returns the value of general-purpose attribute Attribute Number with associated indices Index 1 and Index 2. The number of indices specified must match the number defined for the attribute. This variable is used when the actual Attribute ID to be accessed is stored in another attribute, a variable, a static, etc.

Note: The value of ATTR(Attribute Name) is not the same as the value of simply the Attribute Name. Using the Attribute Name returns the actual attribute value. Function NSYM may be used to translate an attribute identifier into the desired Attribute Number.

EntityNumberIsValid (Entity Number)—Entity Number is Valid. Each entity is given a unique number when created to act as its record of existence. This variable function returns a 1 if the argument *Entity Number* is the number of an entity that currently exists in the simulation, and 0 otherwise (1 for True, 0 for False).

IDENT—Active entity number. Each entity is given a unique number when created to act as its record of existence. These numbers are reused as entities are disposed and new ones are created. The value returned by IDENT corresponds to the Entity Number argument specified in the entity-related variables in this section.

NUMENT—Number of active entities. As each entity is created, NUMENT is increased by one; each disposed entity decreases NUMENT by one. A time-persistent statistic on NUMENT often provides insight into when or whether a model reaches an approximate steady state. If NUMENT increases throughout the run, there may be problems with model logic (e.g., leaving entities in a queue) or the input rates of entities to the system may simply be larger than the total output rate possible.

Activity area variables

Variables are available to obtain the time and/or cost accrued in an activity area.

AreaVATime (Activity Area Name)—Total Value-Added Time. This variable returns the total value-added time accrued for the specified activity area. This represents the total value-added time that was accrued by entities in station logic directly associated with the activity area as well as time “rolled up” from its child activity area(s).

AreaNVATime (ActivityArea Name)—Total Non-Value-Added Time. This variable returns the total non-value-added time accrued for the specified activity area. This represents the total non-valued-added time that was accrued by entities in station logic directly associated with the activity area as well as time “rolled up” from its child activity area(s).

AreaTranTime (Activity Area Name)—Total Transfer Time. This variable returns the total transfer time that was accrued by entities in station logic directly associated with the activity area as well as time “rolled up” from its child activity area(s).

AreaOtherTime (Activity Area Name)—Total Other Time. This variable returns the total time categorized as “Other” accrued for the specified activity area. This represents the total other time that was accrued by entities in station logic directly associated with the activity area as well as time “rolled up” from its child activity area(s).

AreaWaitTime (Activity Area Name)—TotalWait Time. This variable returns the total wait time accrued for the specified activity area. This represents the total wait time that was accrued by entities in station logic directly associated with the activity area as well as time “rolled up” from its child activity area(s).

AreaTotalTime (Activity Area Name)—Total Time. This variable returns the total time (the sum of all five time categories) accrued for the specified activity area. This represents the total time that was accrued by entities in station logic directly associated with the activity area as well as time “rolled up” from its child activity area(s).

AreaVACost (Activity Area Name)—Total Value-Added Cost. This variable returns the total value-added cost accrued for the specified activity area. This represents the total value-added cost that was accrued by entities in station logic directly associated with the activity area as well as cost “rolled up” from its child activity area(s).

AreaNVACost (Activity Area Name)—Total Non-Value-Added Cost. This variable returns the total non-valued-added cost accrued for the specified activity area. This represents the total non-value-added cost that was accrued by entities in station logic directly associated with the activity area as well as cost “rolled up” from its child activity area(s).

AreaTranCost (Activity Area Name)—Total Transfer Cost. This variable returns the total transfer cost accrued for the specified activity area. This represents the total transfer cost that was accrued by entities in station logic directly associated with the activity area as well as cost “rolled up” from its child activity area(s).

AreaOtherCost (Activity Area Name)—Total Other Cost. This variable returns the total cost categorized as “Other” accrued for the specified activity area. This represents the total other cost that was accrued by entities in station logic directly associated with the activity area as well as cost “rolled up” from its child activity area(s).

AreaWaitCost (Activity Area Name)—Total Wait Cost. This variable returns the total wait cost accrued for the specified activity area. This represents the total wait cost that was accrued by entities in station logic directly associated with the activity area as well as cost “rolled up” from its child activity area(s).

AreaTotalCost (Activity Area Name)—Total Cost. This variable returns the total cost (the sum of all five cost categories) accrued for the specified activity area. This represents the total cost that was accrued by entities in station logic directly associated with the activity area as well as cost "rolled up" from its child activity area(s).

Event calendar variables

Arena stores pending and future events in two lists that are jointly referred to as the event calendar. The first list, called the current events list, stores entities that will return to the model at the current time. The future events list stores entities that are scheduled to return to the model at a future time. The calendar variables treat the two lists as one calendar. These variables return integer quantities and are not user-assignable.

The contents of the entire event calendar may be accessed by starting with the first entity and making use of the NEXTINCAL variable until reaching the end of the calendar. The values returned by these variables might be used to evaluate or change the attributes of entities on the calendar.

FIRSTINCAL—First entity on the calendar. FIRSTINCAL returns the entity number of the first entity on the calendar. If there is a current events list at the time that FIRSTINCAL is evaluated, it returns the first entity number on the current events list. Otherwise, it returns the number of the first entity on the future events list. If there are no entities on the calendar, FIRSTINCAL returns a value of 0.

NEXTINCAL (Entity Number)—Next entity on the calendar. NEXTINCAL returns the entity following the specified Entity Number on the calendar. If Entity Number is the last entity on the calendar, a value of 0 is returned. A value must be provided for Entity Number.

Continuous variables

The continuous modeling features in Arena are based on matched pairs of variables called levels and rates. The level variables (also called state or "S" variables) represent the value of a particular continuous-change process over time. The rate variables (also called derivative or "D" variables) represent the rate of change of the level variable. The Levels and Rates modules define these pairs of variables. All continuous variables are real-valued and user-assignable.

Level variables

Level Name (Index 1, Index 2)—Named level variable. The Level Name is defined in the Levels module. If the level variable is indexed, the appropriate number of index values

must be provided. The value of the level variable changes during the simulation based on the value of its corresponding rate variable.

S (Level Number)—Level variable. The S( ) array is an alternate means of accessing level variables defined in the Levels module. The Level Number is the construct number of the corresponding level variable.

Rate variables

Rate Name (Index 1, Index 2)—Named rate variable. The Rate Name is defined in the Rates module. If the rate variable is indexed, the appropriate number of index values must be provided.

D (Rate Number)—Rate variable. The D( ) array is an alternate means of accessing rate variables defined in the Rates module. The Rate Number is the construct number of the corresponding rate variables.

Conveyor variables

Conveyor variables provide information about the state of a conveyor and about the number and size of entities on the conveyor. The Conveyor ID is a conveyor name or number or an expression evaluating to a conveyor number. The only user-assignable conveyor variable is the velocity, VC. Time-persistent statistics often are collected on the conveying entity variables; frequency statistics on ICS provide an overall report of the conveyor state.

General variables

ICS (Conveyor ID)—Conveyor status indicator (0=Idle, 1=Moving, 2=Blocked, 3=Inactive). If the conveyor status is inactive (either initially or after an entity has executed a Stop module), ICS has a value of 3. If a non-accumulating conveyor is active and entities are conveying on it, but some entity has disengaged the conveyor (i.e., it is between Access and Convey modules or is undergoing an unloading delay), the status is blocked with ICS equal to 2. (This state does not apply to accumulating conveyors.) If the conveyor has entities on it and is not disengaged, the state is moving and ICS has a value of 1. (This state applies to all non-empty, active accumulating conveyors.) If no entities are on the conveyor and it is active, ICS equals 0. Note that for accumulating conveyors many entities may be on the conveyor with their progress suspended by a blocking entity; however, the conveyor status is still moving (ICS equals 1). To test for accumulation information, use variables NEA and/or CLA.

MLC (Conveyor ID)—Conveyor length. MLC provides the total length of a conveyor in the model's distance units. For circular conveyors, the length is the sum of the distances

listed in the conveyor's segment set. For straight conveyors, the length is the sum of the segment distances plus the length of the maximum cells per entity defined on the Conveyor module (i.e., maximum cells per entity multiplied by cell width). MLC is an integer variable.

VC (Conveyor ID)—Conveyor velocity. The initial value of VC is the velocity defined in the Conveyor module, with units of the model's distance units per time unit. The velocity may be assigned new values during a simulation run; these changes take place instantaneously. A value of 0.0 for VC stops movement of all entities on the conveyor without changing the conveyor state (ICS). VC is a real-valued, user-assignable variable.

Conveying entity variables

CLA (Conveyor ID)—Length of accumulated entities. This variable applies only to accumulating conveyors and provides the total length occupied by accumulated entities at all accumulation points on the conveyor. CLA is reported in the model's distance units; the length of each accumulated entity is the value specified in the Conveyor module, which may be different from the cell length occupied by entities when they are moving on the conveyor. CLA is a real-valued variable.

LC (Conveyor ID)—Number of occupied cells conveying. Each time an entity accesses cells of the conveyor, LC is incremented by the number of accessed cells. When an entity exits the conveyor, LC is decreased by the number of exited cells. LC is an integer variable. LC is also temporarily decreased when an entity accumulates.

LEC (Conveyor ID)—Length of conveying cells. LEC provides the length of cells occupied by all entities that are in a conveying state. It does not include accumulated entities. LEC is an integer variable.

NEA (Conveyor ID)—Number of accumulated entities. This variable applies only to accumulating conveyors and returns the total number of entities that are accumulated at all accumulation points on the conveyor. NEA is an integer variable.

NEC (Conveyor ID)—Number of conveying entities. NEC provides the number of entities that are in a conveying state. It does not include accumulated entities. NEC is an integer variable.

CNVDST (Conveyor Number, Entity Number)—Entity location on conveyor. CNVDST returns the distance from the beginning of the conveyor of the specified entity. This function would be used if you need to find the location of a particular entity while it is traveling on a conveyor.

Queue variables

Arena provides variables to access information about the entities in each queue. Also, there is a set of variables that directly access the attributes of entities contained in queues based on the entity rank.

The Queue ID is a queue name or number; it cannot be defaulted. Attribute ID is an attribute name or number and also must be specified. Entity Number is the IDENT value of the desired entity; a value must be provided for it as well. None of these variables are user-assignable.

General queue variables

ENTATRANK (Rank, Queue ID)—Entity number of queued entity. ENTATRANK returns the entity number (IDENT value) of the entity at the specified Rank in queue Queue ID.

FIRSTINQ (Queue ID)—First entity number in queue. FIRSTINQ returns the entity number (IDENT value) of the first entity in queue Queue ID.

LASTINQ (Queue ID)—Last entity number in queue. LASTINQ returns the entity number (IDENT value) of the last entity in queue Queue ID.

NQ (Queue ID)—Number in queue. NQ returns the number of entities in queue Queue ID.

Queued entity variables

AQUE (Queue ID, Rank, Attribute Number)—Attribute of queued entity. AQUE returns the value of a general-purpose attribute of the entity at the specified Rank in queue Queue ID. Attribute Number is the number of the desired general-purpose attribute. The function NSYM may be used to translate an attribute name into the desired Attribute Number.

ISQUE (Queue ID, Rank)—Sequence index attribute of queued entity. ISQUE returns the value of the jobstep (Entity.Jobstep or IS) attribute of the entity at the specified Rank in queue Queue ID.

MQUE (Queue ID, Rank)—Station attribute of queued entity. MQUE returns the value of the station (Entity.Station or M) attribute of the entity at the specified Rank in queue Queue ID.

NSQUE (Queue ID, Rank)—Sequence number attribute of queued entity. NSQUE returns the value of the sequence (Entity.Sequence or NS) attribute of the entity at the specified Rank in queue Queue ID.

PREDECESSOR (Entity Number)—Queued entity predecessor. PREDECESSOR returns the entity number (IDENT value) of the entity that directly precedes the specified entity

(Entity Number). The entity provided to PREDECESSOR should be in a queue; otherwise, a 0 is returned. If Entity Number is the first entity in its queue, a 0 also is returned. The predecessor entity is based on the order of entities in the queue (defined by the queue ranking rule).

SAQUE (Queue ID, Attribute Number)—Sum of the attributes of queued entities. SAQUE returns the sum of the specified Attribute Number values of all entities in queue The data type of the specified attribute must be numeric. The function NSYM may be used to translate an attribute name into the desired Attribute Number.

SUCCESSOR (Entity Number)—Queued entity successor. SUCCESSOR is the counterpart to the PREDECESSOR variable. If the specified entity (Entity Number) is the last entity in the queue, a value of 0 is returned.

Resource variables

Arena provides a number of general modeling constructs for controlling the flow of entities through the model. The ID argument for the variables related to the resource, blockage, and storage constructs must be provided as an integer construct number or a construct name.

General resource variables

IRF (Resource ID)—Resource failure. IRF indicates if the specified Resource ID is failed. IRF returns a number greater than 0 if Resource ID is currently failed, or 0 if Resource ID is not currently failed. IRF is an integer quantity.

MR (Resource ID)—Resource capacity. MR returns the number of capacity units currently defined for the specified Resource ID. The Alter module or Schedules option may be used to change the value of MR for a resource. MR is an integer quantity.

NR (Resource ID)—Number of busy resource units. Each time an entity seizes or preempts capacity units of a resource, the NR variable changes accordingly. NR is not user-assignable; it is an integer value.

RESSEIZES (Resource ID)—Number of seizes. This returns the total number of units of the specified resource that have been seized.

RESUTIL (Resource ID)—Resource utilization. ResUtil returns the instantaneous utilization of a resource as a real number between 0 and 1, inclusive. If no units are busy (NR equals 0), this returns 0. If the number busy is greater than or equal to the current capacity (NR >= MR), this returns 1.0. Otherwise, ResUtil will return the ratio of NR/MR. This is most useful in a DSTAT to obtain a time-weighted average utilization.

RTYP (Resource ID)—Resource location. This variable returns a 1 if the resource specified is stationary, and a 2 if the resource specified is positional.

LR (Resource ID)—Resource location. This variable returns the current location of the resource (for positional resources, this is a station number). If the specified resource is not permitted to change locations, LR returns to 0.

STATE (Resource ID)—Resource state. The STATE keyword returns the current state of the specified Resource ID as defined in the Statesets option for the resource. The STATE variable returns an integer number corresponding to the position within the specified Resource ID's associated stateset. It also may be used to assign a new state to the resource.

STATEVALUE (ResourceExpr, StateString)—Resource stateset. STATEVALUE will search the stateset of the resource ResourceExpr for a state with the name StateString, where ResourceExpr can be any Arena expression and StateString can be any character string. If a matching state name is found, the integer value for that state is returned. If StateString is not found, then an error is generated. If resource ResourceExpr does not have any stateset associated with it, the StateString is compared against the autostates BUSY, IDLE, INACTIVE, and FAILED. If a match with one of the autostates is found, then the appropriate autostate integer associated with it is returned (see below for these values).

The four resource state constants below are available to check the automatic state of a resource. A state constant is only available if the autostate has not been associated with a user-defined state. Unlike other variables, the state constants are not user-assignable. Typically, they are used to monitor the state of a resource in a conditional expression when using autostates. For example, STATE(Resource ID) == IDLE_RES returns true if Resource ID is currently in the idle autostate.

IDLE_RES—Resource state constant. The IDLE_RESource is used in an expression to check whether a resource is currently in the idle state. A resource is in the idle state when all units are idle and the resource is not failed or inactive. (The numerical equivalent of this variable is -1.)

BUSY_RES—Resource state constant. The BUSY_RESource is used in an expression to check whether a resource is currently in the busy state. A resource is in the busy state when it has one or more busy units. (The numerical equivalent of this variable is -2.)

INACTIVE_RES—Resource state constant. The INACTIVE_RESource is used in an expression to check whether a resource is currently in the inactive state. A resource is in the inactive state when it has zero capacity and is not failed. (The numerical equivalent of this variable is -3.)

FAILED_RES—Resource state constant. The FAILED_RESource is used in an expression to check whether a resource is currently in the failed state. A resource is in the failed state when a failure is currently acting on the resource. (The numerical equivalent of this variable is -4.)

Resource cost variables

One major component of costs in many models is the cost associated with resources. These costs are in three categories: costs applied during the time a resource is busy, costs applied during the time a resource is idle (scheduled, but not busy), and costs applied each time a resource is used (regardless of how long it is used).

If both the costing statistics and the resource statistics options are enabled on the **Run > Setup > Project Parameters** property page, then resource costs will be calculated and reported automatically. The following variables are supplied if you instead want to calculate resource costs manually. Note that you should not do both manual and automatic calculations concurrently or both may report incorrect results.

ResBusyCost (Resource ID)—Resource busy cost rate. This returns the busy cost rate specified in the resource module. This rate is the cost per time unit to be applied during the entire time a resource is busy. Note that although this rate is entered as a cost per hour, this variable always returns the rate as a *cost per base time unit* (e.g., TNOW units). So regardless of the current settings for base time units, this rate can be used without conversion.

ResIdleCost (Resource ID)—Resource idle cost rate. This returns the idle cost rate specified in the resource module. This rate is the cost per time unit to be applied during the entire time a resource is scheduled but not in use. Note that although this rate is entered as a cost per hour, this variable always returns the rate as a *cost per base time unit* (e.g., TNOW units). So regardless of the current settings for base time units, this rate can be used without conversion.

ResUseCost (Resource ID)—Resource usage cost. This returns the usage cost specified in the resource module. This is not a rate, but rather the cost associated with each use (or Seize) of a resource.

Replication variables

Arena uses two variables to record the current and maximum number of replications.

MREP—Maximum replications. This integer variable is the Number of Replications value on the **Run > Setup > Replication Parameters** property sheet. It is user-assignable.

NREP—Replication number. NREP returns the current replication number, an integer value. It may not be changed by the user.

Note: Use the Expression Builder to look up replication variables easily when building or editing an expression.

Date and time variables

Calendar dates and times variables

CalYear(TimeExpr)—Returns the year with the century (e.g., 1981, 2010) of the calendar date corresponding to the simulation time *TimeExpr*.

CalMonth(TimeExpr)—Returns an integer from 1 through 12 that represents the month of the calendar date corresponding to the simulation time *TimeExpr*. January is month 1 and December is month 12.

CalWeek(TimeExpr)—Returns an integer from 1 to 53 that represents the week of the year of the calendar date corresponding to the simulation time TimeExpr. The first week (i.e., week 1) is the week that contains January 1st. The first day of each week is Sunday. Note that dates late in a year could actually belong to week 1 of the following year.

CalDayOfYear(TimeExpr)—Returns an integer from 1-366 that represents the day of the year of the calendar date corresponding to the simulation time *TimeExpr*.

CalDayOfMonth(TimeExpr)—Returns an integer from 1-31 that represents the day of the month of the calendar date corresponding to the simulation time *TimeExpr*.

CalDayOfWeek(TimeExpr)—Returns an integer from 1-7 that represents the day of the week of the calendar date corresponding to the simulation time *TimeExpr*. The first day of each week (i.e., day 1) is Sunday.

CalHour(TimeExpr)—Returns the integer hour portion in 24-hr format (0-23) of the calendar time corresponding to the simulation time *TimeExpr*.

CalMinute(TimeExpr)—Returns the integer minute portion (0-59) of the calendar time corresponding to the simulation time *TimeExpr*.

CalSecond(TimeExpr)—Returns the integer second portion (0-59) of the calendar time corresponding to the simulation time *TimeExpr*.

CalDateToBaseTime—(*YearExpr, MonthExpr, DayExpr*) Returns the value of simulated time (in the base time units) corresponding to midnight of the calendar date specified by the *YearExpr*, *MonthExpr*, and *DayExpr* arguments.

Expressions evaluated in the *YearExpr*, *MonthExpr*, and *DayExpr* arguments are rounded to the nearest integer. The accepted range for the *YearExpr* argument is 100 to 9999.

The accepted ranges for the *MonthExpr* and *DayExpr* arguments are 1–31 for days and 1–12 for months. The expressions entered must evaluate to a valid date.

Current and final simulation time variables

TNOW—Current simulation time. TNOW records the simulation clock time as the model progresses. After all activities at a particular simulation time have been processed, TNOW is updated to the time of the next activity (e.g., entity event). TNOW is a real-valued quantity; it is not user-assignable.

TFIN—Final simulation time. TFIN is the ending time scheduled for the replication; it is a real-valued quantity. If the ending time is defaulted, TFIN returns a large value (1.0E+20). TFIN may be assigned a value (greater than the current simulation time, TNOW) during a replication. Any subsequent replications use the value of TFIN from the **Run > Setup > Replication Parameters**.

Converting durations to the base time units variables

Arena also includes four functions that may be used to convert a time value expressed in seconds, minutes, hours, or days into a time value expressed in the simulation's base time units that is specified in **Run > Setup > Replication Parameters**.

SecondsToBaseTime (Expression)—Converts the expression evaluated to a time duration in seconds into a time duration in the base time units.

MinutesToBaseTime (Expression)—Converts the expression evaluated to a time duration in minutes into a time duration in the base time units.

HoursToBaseTime (Expression)—Converts the expression evaluated to a time duration in hours into a time duration in the base time units.

DaysToBaseTime (Expression)—Converts the expression evaluated to a time duration in days into a time duration in the base time units.

System response variables

The System Response variables provide an overview of system performance.

The entity-related variables below are only enabled when the entity statistics option is selected on the Project Parameters page of Arena's **Run > Setup** dialog. The entity cost-related variables also require that the costing option be selected. The entity cost variables are updated each time costs are accrued to an entity.

The resource cost-related variables below are only enabled when both the resource statistics and costing options are selected from **Run > Setup > Replication Parameters**.

Throughput variable

Total.Throughput—Total entity throughput. This is the total number of entities that have been recorded. This can be a simple way of determining the total number of entities that have finished processing.

Cost variables

Total.VACost—Total value-added cost. This is the total value-added cost for all entities.

Total.NVACost—Total non-value-added cost. This is the total non-value-added cost for all entities.

Total.WaitCost—Total waiting cost. This is the total wait cost for all entities.

Total.TranCost—Total transfer cost. This is the total transfer cost for all entities.

Total.OtherCost—Total other cost. This is the total other cost for all entities.

Total.EntityCost—Total entity cost. This is the sum of the costs in the five allocation categories above for all entities. This can be a simple way of determining the total cost of all entities. Note that this number will typically include those resource costs that can be attributed to a specific entity.

Total.ResUseCost—Total resource usage cost. This is the total of the usage costs (e.g., costs per seize) for all resources. It is the product of ResSeizes( ) * ResUseCost( ), summed for all resources. Note that these costs are typically also included in entity costs because they can be assigned to a particular entity.

Total.ResBusyCost—Total resource busy cost. This is the total of the busy costs (e.g., costs while seized) for all resources. It is the product of ResBusyCost( ) * Average Number Busy * TNOW, summed for all resources. Note that these costs are typically included in entity costs as well because they can be assigned to a particular entity.

Total.ResIdleCost—Total resource idle cost. This is the total of the idle costs (e.g., costs while scheduled, but not busy) for all resources. It is the product of ResIdleCost( ) * Average Number Idle * TNOW, summed for all resources. Note that these costs are *never* included in entity costs because they cannot be assigned to a particular entity.

Total.ResourceCost—Total resource cost. This is the sum of the costs in the three categories above for all resources. This can be a simple way of determining the total cost of all resources used in the system.

Total.SystemCost—Total system costs. This is the sum of all costs. This can be a simple way of evaluating the total cost of one system relative to another. It is the sum of Total.EntityCost and all costs that have been incurred, but not yet allocated to an entity.

Statistics collection variables

A set of variables is provided to access information about each type of Arena statistic. The types of variables provided depend upon the statistic type. Most of these variables take the statistic identifier as an argument; frequency variables take the frequency number and, in some cases, category number. All of these arguments are required.

When simulation statistics are cleared between replications or by the Warmup Period, the statistics variables are reinitialized to their default values unless otherwise noted. Unless otherwise stated, these variables are not user-assignable.

Counter statistics variables

MC (Counter ID)—Count limit. MC is the limit defined in the Counters (or Statistic) module; if the limit is defaulted, MC is given a value of 0. MC is user-assignable; its value (if changed) is retained between replications. It is not affected by statistics reinitialization.

NC (Counter ID)—Count value. Each time a count occurs, NC is changed by the specified value. The Initialize Statistics option on the Replication Parameters page of the **Run > Setup** dialog determines whether counters are initialized between replications.

Time-persistent statistics (Cstat) variables

CAVG (Cstat ID)—Average value. CAVG records the average of the cstat expression throughout the replication.

CMAX (Cstat ID)—Maximum value. CMAX records the maximum value taken by the cstat expression during the replication.

CMIN (Cstat ID)—Minimum value. CMIN records the minimum value taken by the cstat expression during the replication.

CSTD (Cstat ID)—Standard deviation. CSTD calculates the standard deviation of the recorded values of the cstat expression.

CTPD (Cstat ID)—Time period. CTPD returns the time period over which the statistics have been collected. If a Warmup Period is provided on the Replication Parameters page of the **Run > Setup** dialog, CTPD increases from 0.0 to the warmup time. After the warmup time, it is restarted from 0.0.

CHALF (Cstat ID)—Half-width. CHALF returns the 95% confidence interval around the mean value of the specified cstat. If there is insufficient data, the data is correlated or an error is detected; this will return a very large number.

CVALUE (Cstat ID)—Last recorded value. CVALUE returns the last recorded value for the specified cstat. When animating a cstat histogram, it is CVALUE, not the CAVG, that is typically displayed.

CBATCH (Cstat ID, Batch Number)—Average value in batch. This variable returns the current average value in the batch number specified, for the cstat specified. This is used in conjunction with the CHALF variable, which calculates the confidence interval for a cstat.

CNUMBAT (Cstat ID)—Number of batches. CNUMBAT returns the current number of full batches being used in the calculation of the variable CHALF. For the cstat specified, there is always a minimum of 20 and a maximum of 40 batches. The value of this variable changes as more data is collected in a replication.

CBATSIZ (Cstat ID)—Batch size. CBATSIZ is a variable that is used in conjunction with the CHALF variable, which calculates the confidence interval for a cstat. CBATSIZ returns the current sample size of each batch used in these calculations. The value of this variable changes as more data is collected in a replication.

Time-persistent statistics (Dstat) variables

DAVG (Dstat ID)—Average value. DAVG records the average of the dstat (time-persistent) expression throughout the replication.

DMAX (Dstat ID)—Maximum value. DMAX records the maximum value taken by the dstat (time-persistent) expression during the replication.

DMIN (Dstat ID)—Minimum value. DMIN records the minimum value taken by the dstat (time-persistent) expression during the replication.

DSTD (Dstat ID)—Standard deviation. DSTD calculates the standard deviation of the recorded values of the dstat (time-persistent) expression.

DTPD (Dstat ID)—Time period. DTPD returns the time period over which the statistics have been collected. If a warmup period is provided on the Replication Parameters page of the **Run > Setup** dialog, DTPD increases from 0.0 to the warmup time. After the warmup time, it is restarted from 0.0.

DHALF (Dstat ID)—Half-width. DHALF returns the 95% confidence interval around the mean value of the specified dstat. If there is insufficient data, the data is correlated or an error is detected; this will return a very large number.

DVALUE (Dstat ID)—Last recorded value. DVALUE returns the last recorded value for the specified dstat. When animating a dstat histogram, it is DVALUE, not the DAVG, that is typically displayed.

DBATCH (Dstat ID, Batch Number)—Average value in batch. This variable returns the current average value in the batch number specified, for the dstat specified. This is used in conjunction with the DHALF variable, which calculates the confidence interval for a cstat.

DNUMBAT (Dstat ID)—Number of batches. DNUMBAT returns the current number of full batches being used in the calculation of the variable DHALF. For the dstat specified, there is always a minimum of 20 and a maximum of 40 batches. The value of this variable changes as more data is collected in a replication.

DBATSIZ (Dstat ID)—Batch size. DBATSIZ is a variable that is used in conjunction with the DHALF variable, which calculates the confidence interval for a dstat. DBATSIZ returns the current sample size of each batch used in these calculations. The value of this variable changes as more data is collected in a replication.

Frequencies statistics variables

FAVG (Frequency ID, Category)—Average time in category. FAVG is the average time that the frequency expression has had a value in the specified category range. FAVG equals FRQTIM divided by FCOUNT.

FCATS (Frequency ID)—Number of categories. FCATS returns the number of categories of a frequency, including the out-of-range category. FCATS is an integer value.

FCOUNT (Frequency ID, Category)—Frequency category count. FCOUNT is the number of occurrences of observations for the Frequency Number of values in the Category number; it is an integer value. Only occurrences of time > 0 are counted.

FHILIM (Frequency ID, Category)—Frequency category high limit. FHILIM is the upper limit of a category range or simply the value if no range is defined for the particular category number of Frequency Number. FHILIM is user-assignable.

FLOLIM (Frequency ID, Category)—Frequency category low limit. FLOLIM defines the lower limit of a frequency category range. Values equal to FLOLIM are not included in the category; all values larger than FLOLIM and less than or equal to FHILIM for the category are recorded. FLOLIM is user-assignable.

FSTAND (Frequency ID, Category)—Standard category percent. FSTAND calculates the percent of time in the specified category compared to the time in all categories.

FRQTIM (Frequency ID, Category)—Time in category. FRQTIM stores the total time of the frequency expression value in the defined range of the category number.

FRESTR (Frequency ID, Category)—Restricted category percent. FRESTR calculates the percent of time in the specified category compared to the time in all restricted categories.

FTOT (Frequency ID)—Total frequency time. FTOT records the total amount of time that frequency statistics have been collected for the specified frequency number.

FTOTR (Frequency ID)—Restricted frequency time. FTOTR records the amount of time that the specified frequency number has contained values in non-excluded categories (i.e., categories that have a value in the restricted percent column).

FVALUE (Frequency ID)—Last recorded value. FVALUE returns the last recorded value for the specified frequency. When animating a frequency histogram, it is FVALUE, not the FAVG, that is typically displayed.

Tally statistics variables

TAVG (Tally ID)—Average value. TAVG records the average of the tally variable Tally ID throughout the replication.

TMAX (Tally ID)—Maximum value. TMAX returns the largest observed value of the tally variable.

TMIN (Tally ID)—Minimum value. TMIN returns the smallest observed value of the tally variable.

TNUM (Tally ID)—Number of observations. TNUM returns the number of observed values of the tally variable; it is an integer quantity.

TSTD (Tally ID)—Standard deviation. TSTD calculates the standard deviation of the recorded values of the tally variable.

THALF (Tally ID)—Half-width. THALF returns the 95% confidence interval around the mean value of the specified tally. If there is insufficient data, the data is correlated or an error is detected; this will return a very large number.

TVALUE (Tally ID)—Last recorded value. TVALUE returns the last recorded value for the specified tally. When animating a tally histogram, it is TVALUE, not the TAVG, that is typically displayed.

TBATCH (Tally ID, Batch Number)—Average value in batch. This variable returns the current average value in the batch number specified for the tally specified. This is used in conjunction with the THALF variable, which calculates the confidence interval for a tally.

TNUMBAT (Tally ID)—Number of batches. TNUMBAT returns the current number of full batches being used in the calculation of the variable THALF. For the tally specified, there is always a minimum of 20 and a maximum of 40 batches. The value of this variable changes as more data is collected in a replication.

TBATSIZ (Tally ID)—Batch size. TBATSIZ is a variable that is used in conjunction with the THALF variable, which calculates the confidence interval for a tally. TBATSIZ returns the current sample size of each batch used in these calculations. The value of this variable changes as more data is collected in a replication.

Output statistics variable

OVALUE (Output ID)—Last recorded value. OVALUE returns the last recorded value for the specified output.

Post-run statistics variables

ORUNAVG (Output ID)—Average value. This function returns the average value recorded for a particular output statistic across all replications run so far. This considers only the final values of completed replications.

ORUNMAX (Output ID)—Maximum value. This function returns the maximum value recorded for a particular output statistic across all replications run so far. This considers only the final values of completed replications.

ORUNMIN (Output ID)—Minimum value. This function returns the minimum value recorded for a particular output statistic across all replications run so far. This considers only the final values of completed replications.

ORUNHALF (Output ID)—Half-width. This function returns the value of the half-width of the 95% confidence interval around the mean for a particular output statistic across all replications run so far. This considers only the final values of completed replications.

Transporter variables

Transporter-related variables fall into four broad categories. First, general-status variables apply to both free-path and guided transporters, describing the idle or busy status and velocity characteristics of the transporters. Two additional sets of variables for free-path and guided transporters provide information related to the position and other characteristics of the transporters and their units. Finally, a set of variables related to the guided transporter network are available to access information about the network composition.

Most variables have one or more arguments that take on a transporter, link, intersection, station, network, or distance ID. In each of these cases, a number, name, or expression evaluating to a construct number may be provided. The Unit Number of many variables defines the specific transporter unit from a set of vehicles; it may be specified as an expression that evaluates to an integer quantity. Unless otherwise described, all arguments must be specified for transporter variables.

Velocity-related variables (including acceleration and deceleration) are real-valued quantities and are user-assignable. Most other variables are integer quantities; they are not user-assignable unless otherwise noted.

General-status transporter variables

IT (Transporter ID, Unit Number)—Transporter unit status. IT takes a value of 0 when the transporter is idle and active, a value of 1 when the transporter unit is busy, and a value of 2 when the transporter is inactive. The model may assign a value of 0 to an inactive transporter or a value of 2 to an idle transporter. However, the transporter may not be taken directly into a busy state by assignment.

MT (Transporter ID)—Number of active units. MT is the total number of active individual units in transporter set Transporter ID.

NT (Transporter ID)—Number of busy units. NT records the number of busy transporter units of a transporter set.

VT (Transporter ID)—Transporter set velocity. VT is the default velocity for transporter units of set Transporter ID. Initially, all transporters have this velocity for movement (defined in the Transporter module). A temporary change to VT may be affected by defining a movement velocity on Move, Request, or Transport modules; these values do not change VT. An individual unit's velocity may be reassigned (see VTU) to override the default transporter set velocity.

VTU (Transporter ID, Unit Number)—Transporter unit velocity. By default, all transporters use velocity VT. An individual transporter unit may use a different velocity if the value of VTU for the unit is changed.

Free-path transporter variables

ID (Transporter ID, Station ID)—Transporter distance. ID returns the distance from the station stored in the active entity's station (M) attribute to destination Station ID in the distance set followed by the specified Transporter ID.

IDIST (Distance Set ID, Beginning Station ID, Ending Station ID)—Distance set value. IDIST returns the distance value from station Beginning Station ID to station Ending Station ID in the specified Distance Set ID.

LT (Transporter ID, Unit Number)—Transporter location. LT returns the current station location or destination (if the transporter is moving) of the specified transporter unit.

Guided transporter variables

ACC (Transporter ID)—Acceleration. ACC returns the acceleration factor for transporters in set Transporter ID.

DEC (Transporter ID)—Deceleration. DEC returns the deceleration factor for transporters in set Transporter ID.

ISZT (Transporter ID, Unit Number)—Size type. ISZT returns a 1 if the specified transporter size is based on zones, or 2 if it is based on length. The value of ISZT is constant for a particular transporter throughout a run; however, the value of NSZT (transporter size value) may change if a transporter executes Capture or Relinquish modules.

LDL (Transporter ID, Unit Number)—Destination link. If the transporter is moving toward a destination that was defined as a position on a link (on a Move, Request, or Transport module), LDL returns the link number to which the unit is being moved. (Also see LDZ.) Otherwise, LDL returns 0 (e.g., transporter is moving to a station/intersection destination or is stationary).

LDX (Transporter ID, Unit Number)—Destination intersection. If the transporter is moving toward a station or intersection destination, LDX returns the intersection number to which the unit is moving. LDX returns 0, otherwise.

LDZ (Transporter ID, Unit Number)—Destination zone number. If the transporter is moving toward a link destination (see LDL), LDZ returns the destination zone number. LDZ returns 0, otherwise.

LT (Transporter ID, Unit Number)—Intersection location. LT returns the current intersection number if the transporter is stationary, or moving in an intersection, or the ending intersection of the link if the transporter is in transit through a link (based on current travel direction).

LTL (Transporter ID, Unit Number)—Link location. If the transporter is moving through a link or is stationary in a link, LTL returns the link number. If the transporter is in an intersection, LTL returns 0.

LTZ (Transporter ID, Unit Number)—Zone location. LTZ returns the zone number in link LTL, or 0 if the transporter is in an intersection.

NSZT (Transporter ID, Unit Number)—Transporter size value. NSZT returns the current number of zones or length units occupied by the specified transporter. The actual NSZT value may be interpreted as zones or length based upon the size type value, ISZT. When a transporter captures new zones/length, NSZT increases to the new transporter size; when it relinquishes zones/length, NSZT decreases. If NSZT is 0 (i.e., all zones have been relinquished), the transporter moves through the system without blocking or being blocked by other transporters.

TAZ (Transporter ID, Unit Number)—Zone arrival time. TAZ returns the time at which the transporter arrived at the end of the last zone through which it moved. If the transporter is moving in a zone (or intersection), TAZ returns the time that it arrived at the

previously occupied zone/intersection. If the transporter is stationary, TAZ records the time that it stopped moving.

TVF (Transporter ID)—Turning velocity factor. TVF returns the turning velocity factor for the transporter set specified by Transporter ID. This quantity is multiplied by a transporter unit's current velocity when the transporter changes travel direction, if directions are specified in the guided network.

TWZ (Transporter ID, Unit Number)—Time waiting in zone. TWZ accumulates the total amount of time that the specified transporter unit has spent waiting for access to zones because of blockage by another transporter unit. TWZ does not record time spent stationary due to delays in the model (e.g., idle transporter waiting for a request).

Guided network variables

INXNUM (Station ID)—Intersection number. INXNUM returns the intersection number that is associated with station, Station ID, or 0 if none was specified. This value may be changed by making an assignment to INXNUM, in which case any subsequent transporter movements to the station will send the transporter to the newly specified intersection.

IDSNET (Network ID, Beginning Intersection ID, Ending Intersection ID)—Network distance. IDSNET returns the travel distance in the specified Network ID from Beginning Intersection ID to Ending Intersection ID. The length of the beginning intersection is not included in this quantity; the full length of the ending intersection is included.

LENZ (Link ID)—Zone length. LENZ returns the length of each zone of the specified Link ID.

LNKNUM (Beginning Intersection ID, Ending Intersection ID)—Connecting link. LNKNUM returns the link number that connects Beginning Intersection ID with Ending Intersection ID.

LTYP (Link ID)—Link type. LTYP returns a value of 1 if Link ID is unidirectional, 2 if it is bi-directional, and 3 if it is a spur.

LX (Intersection ID)—Intersection length. LX returns the length of the specified Intersection ID.

MZ (Link ID)—Number of zones. MZ returns the number of zones defined for the specified Link ID. (Also see LENZ.)

NDX (Link ID)—Destination intersection. NDX returns the destination intersection of travel of transporters on the specified Link ID; if the link is unoccupied, NDX returns a 0. If there are vehicles on the link, then the ending intersection number is returned if travel is in the forward direction (always the case for uni-directional links). The beginning intersection is returned if travel is backward on a bi-directional link or a spur.

NEXTX (Network ID, Beginning Intersection ID, Destination Intersection ID)—Next travel intersection. NEXTX returns the next intersection of travel in the specified Network ID from the Beginning Intersection ID to the Destination Intersection ID. If NEXTX returns a value equal to the Destination Intersection, then the two intersections are directly connected by a link.

NL (Link ID)—Number of occupied zones in link. NL returns the number of zones occupied by transporters in the specified Link ID. The value of NL does not include reserved zones (i.e., via Capture module).

NX (Intersection ID)—Intersection status. NX returns a value of 0 if the intersection is unoccupied, 1 if it is occupied, or -1 if it is reserved (i.e., via Capture module).

NXB (Link ID)—Beginning intersection. NXB returns the beginning intersection number of the specified Link ID.

NXE (Link ID)—Ending intersection. NXE returns the ending intersection number of the specified Link ID.

NZ (Link ID, Zone Number)—Zone status. NZ returns the status of the specified Zone Number in Link ID. A value of 0 is returned if the zone is unoccupied, 1 if it is occupied, or -1 if it is reserved (i.e., via a Capture module).

VL (Link ID)—Link velocity factor. VL returns the velocity factor applied to travel through the specified Link ID, as defined in the NetworkLink module.

VX (Intersection ID)—Intersection velocity factor. VX returns the velocity factor applied to travel through the specified Intersection ID.

Miscellaneous Variables

Blockage status variable

NB (Blockage ID)—Current blockage quantity. NB returns the number of blockages set (queue and block blockages) for the specified Blockage ID. The value of NB may be changed by the Block and Unblock modules and by entities that reference blockages when entering and leaving queues. NB is not user-assignable.

Expressions variables

Three mechanisms are provided to access the value of a defined expression. All return real- or string-valued quantities; they are not user-assignable. Where an *Expression Number* argument is required, it is the construct number of the expression (as listed in the Expressions module from the Elements panel). The number may be entered as a constant, an expression, or as NSYM(Expression Name).

ED (Expression Number)—Expression value. ED returns the current value of the specified expression number.

EXPR (Expression Number [, Index 1, Index 2])—Expression value. EXPR returns the value of the specified expression, where *Expression Number* is the instance number in the Expressions module. *Index 1* and *Index 2* must be specified if the expression is indexed. EXPR is used when the actual expression number to be used is stored in an attribute, variable, or static.

Note: The value of EXPR(Expression ID) is not the same as the value of Expression ID. The Expression ID returns the actual expression value; function NSYM may be used to translate an expression identifier into the desired Expression Number.

Expression Name [(Index 1, Index 2)]—Expression value. To return the value of an expression, the simple form using the expression name itself (with indices if the expression is indexed) may be used.

Functions variables

NSYM (Symbol Name)—Symbol number. All defined simulation elements have a unique number. For those constructs that have names, the function NSYM may be used to return the number corresponding to the construct name. The Symbol Name must be specified as a simple name or indexed name; the index, if used, may be an expression. NSYM returns an integer value; it is not user-assignable.

NUMBLK [(Block Label)]—Block number. NUMBLK returns the number of the block corresponding to Block Label. It is an integer value; it is not user-assignable. If Block Label is omitted, NUMBLK returns the number of the current block.

TF (Table ID, X Value)—Table function value. TF evaluates the value stored in Table ID for a real-valued quantity X value. TF returns a real-valued quantity; it is not user-assignable.

UF (User Function Number)—User-coded function. UF executes the C, FORTRAN, or VBA code of function UF, passing User Function Number as the function number parameter. The return value of UF is provided by the user-coded routine; it is a real-valued quantity.

General-purpose global variables

Three mechanisms are provided to access the value of a variable. All return real- or string-valued quantities; variables are user-assignable. Where a *Variable Number* is required, it is the construct number of the variable (as listed in the Variables module from the Elements panel). The number may be entered as a constant, an expression, or as NSYM(Variable Name).

V (Variable Number)—Variable value. The V( ) array returns the current value of the specified Variable Number, which is the instance number of the corresponding variable in the Variables module.

VAR (Variable Number [, Index 1, Index 2])—Variable value. VAR returns the value of the general-purpose variable *Variable Number* with associated indices *Index 1* and *Index 2*; the number of indices specified must match the number defined for the variable. This variable is used when the actual Variable ID to be accessed or changed is stored in an attribute, another variable, a static, etc.

Note: The value of VAR(Variable Name) is not the same as the value of simply the Variable Name. Using the Variable Name returns the actual variable value; function NSYM may be used to translate a variable identifier into the desired Variable Number.

Variable Name [(Index 1, Index 2)]—Variable value. To assign or return the value of a variable, the simple form using the variable name itself (with indices if the variable is indexed) may be used.

Parameters variables

CO (Parameter Set ID)—Constant value. CO returns the value of the first parameter value in the specified Parameter Set ID. CO returns a real-valued quantity; it is not user-assignable.

NMPAR (Parameter Set ID)—Number of parameter values. NMPAR returns the number of values defined in the specified Parameter Set ID. It is an integer value; it is not user-assignable.

P (Parameter Set ID, Parameter Number)—Parameter value. The function P returns the value of the specified Parameter Number in element Parameter Set ID. New values may be assigned within a parameter set by using the P function; it is a real-valued quantity.

Schedule variables

NSExpo (Schedule ID)—Non-stationary exponential distribution. This function is typically used in the Create module for specifying interarrival times that vary according to a schedule. The schedule must be specified as type *Arrivals*. This distribution is used in situations where arrivals follow a Poisson process; however, the arrival rate varies over time. For example, the arrival rate at a fast-food restaurant will be larger during the lunch time rush hour than during mid-morning. In this case, the arrival rate automatically changes to follow the values specified in its schedule.

SchedValue (Schedule ID)—Schedule value. This function returns the current value of a schedule. The schedule must be of type Other. This is useful for returning a value that varies according to a schedule. For example, a learning curve (skill level) could be

modeled using the expression NominalProcessTime * SchedValue(SkillLevel) for a process time where SkillLevel is the name of a schedule of type Other.

J index variable

J—Search index variable. J is an integer, user-assignable variable. It is used in the Search and FindJ modules to return the selected index value, based on a search criterion and range. J may be used within a model for other purposes as well; however, execution of a Search or FindJ module will give J a new value.

Set variables

The Sets module allows definition of an indexed group of constructs of the same type (e.g., resources, queues, stations). The set index refers to the order of a particular construct within a set. Set variables return integer quantities. They are not user-assignable.

MEMBER (Set ID, Index)—Set member. The MEMBER function returns the construct number of a particular set member. Set ID defines the set to be examined; Index is the index of the set. The construct number (e.g., resource number) returned by MEMBER may be used in a block or assignment.

MEMIDX (Set ID, Member ID)—Member index in set. MEMIDX returns the set index value of a particular construct (Member ID) in the specified Set ID. Member ID is an expression evaluating to a construct number.

Note: If Set ID contains attributes, variables, or expressions constructs, the NSYM function may be necessary to define Member ID correctly since these construct names are evaluated to their values (rather than simply translating into the construct number).

NUMMEM (Set ID)—Number of members. NUMMEM returns the number of constructs in the specified Set ID.

Station variables

INXNUM (Station ID)—Intersection number. INXNUM returns the intersection number that is associated with station, Station ID, or 0 if none was specified. This value may be changed by making an assignment to INXNUM, in which case any subsequent transporter movements to the station will send the transporter to the newly specified intersection.

MSQ (Sequence ID, Sequence Index)—Sequence station. MSQ returns the station to be visited at the specified Sequence Index (i.e., IS attribute value) of Sequence ID. MSQ is an integer quantity; it is not user-assignable.

NE (Station ID)—Number of entities transferring. NE returns the number of entities currently in transit to the specified destination Station ID. Each time an entity conveys, routes, or transports to a station, variable NE is incremented; when an entity arrives at the

end of such a transfer, NE is decremented. Note that material-handling modules such as Move and Request do not change NE; only the transferring entities affect its value. NE is an integer quantity; it is not user-assignable.

The following variables return the time or cost accrued in the activity area associated with Station Name, or 0 if an activity area is not associated with the station:

StnVATime(Station Name)— Total Value-Added Time. This variable returns the total value added time accrued for the activity area associated with the specified station.

StnNVATime(Station Name)—Total Non-Value Added Time. This variable returns the total non-value added time accrued for the activity area associated with the specified station.

StnTranTime(Station Name)—Total Transfer Time. This variable returns the total transfer time accrued for the activity area associated with the specified station.

StnOtherTime(Station Name)—Total Other Time. This variable returns the total time categorized as “Other” accrued for the activity area associated with the specified station.

StnWaitTime(Station Name)—Total Wait Time. This variable returns the total wait time accrued for the activity area associated with the specified station.

StnTotalTime(Station Name)—Total Time. This variable returns the total time (the sum of all 5 time categories) accrued for the activity area associated with the specified station.

StnVACost(Station Name)—Total Value Added Cost. This variable returns the total value-added cost accrued for the activity area associated with the specified station.

StnNVACost(Station Name)—Total Non-Value Added Cost. This variable returns the total non-value added cost accrued for the activity area associated with the specified station.

StnTranCost(Station Name)—Total Transfer Cost. This variable returns the total transfer cost accrued for the activity area associated with the specified station.

StnOtherCost(Station Name)—Total Other Cost. This variable returns the total cost categorized as “Other” accrued for the activity area associated with the specified station.

StnWaitCost(Station Name)—Total Wait Cost. This variable returns the total wait cost accrued for the activity area associated with the specified station.

StnTotalCost(Station Name)—Total Cost. This variable returns the total cost (the sum of all 5 cost categories) accrued for the activity area associated with the specified station.

Storage variable

NSTO (Storage ID)—Number of entities in storage. NSTO records the number of entities that are stored in the specified Storage ID. It is changed by the Store and Unstore modules and by Delay, Request, and Move modules that specify storages. NSTO is not user-assignable.

Stack variables

The following variables are used in conjunction with the Stack module. When a Stack Save operation is performed, the full set of internal time attributes and cost attributes (see “Attributes and entity-related variables ” on page 40) is saved. The variables below are used to access those saved values.

Diff.StartTime—Difference in saved start time. This returns the current start time (Entity.StartTime) minus the saved start time. This is commonly used for time in process.

Diff.VATime—Difference in value-added time. This returns the amount of value-added time that has accrued since the last Stack Save operation. Its value is the entity’s current value-added time minus the saved value-added time.

Diff.VACost—Difference in value-added cost. This returns the amount of value-added cost that has accrued since the last Stack Save operation. Its value is the entity’s current value-added cost minus the saved value-added cost.

Diff.NVATime—Difference in non-value-added time. This returns the amount of non-value-added time that has accrued since the last Stack Save operation. Its value is the entity’s current non-value-added time minus the saved non-value-added time.

Diff.NVACost—Difference in non-value-added cost. This returns the amount of non-value-added cost that has accrued since the last Stack Save operation. Its value is the entity’s current non-value-added cost minus the saved non-value-added cost.

Diff.WaitTime—Difference in waiting time. This returns the amount of waiting time that has accrued since the last Stack Save operation. Its value is the entity’s current waiting time minus the saved waiting time.

Diff.WaitCost—Difference in waiting cost. This returns the amount of waiting cost that has accrued since the last Stack Save operation. Its value is the entity’s current waiting cost minus the saved waiting cost.

Diff.TranTime—Difference in transfer time. This returns the amount of transfer time that has accrued since the last Stack Save operation. Its value is the entity’s current transfer time minus the saved transfer time.

Diff.TranCost—Difference in transfer cost. This returns the amount of transfer cost that has accrued since the last Stack Save operation. Its value is the entity's current transfer cost minus the saved transfer cost.

Diff.OtherTime—Difference in other time. This returns the amount of other time that has accrued since the last Stack Save operation. Its value is the entity's current other time minus the saved other time.

Diff.OtherCost—Difference in other cost. This returns the amount of other cost that has accrued since the last Stack Save operation. Its value is the entity's current other cost minus the saved other cost.

Flow variables

Arena provides variables to access information about tanks, regulators, and sensors in the model via the Flow Process panel.

The arguments Tank ID, Regulator ID, and Sensor ID refer to the tank, regulator, and sensor name respectively. They cannot be defaulted.

Tank variables

TankCapacity (Tank ID)—Tank Capacity. Returns the capacity of the tank. This variable is assignable.

TankLevel (Tank ID)—Tank Level. Returns the current level of material in the tank. This variable is assignable.

TankNetRate (Tank ID)—Tank Net Rate. Returns the net flow rate in the tank. The rate is positive if the tank's level is increasing, and negative if the Tank's level is decreasing.

TankQtyAdded (Tank ID)—Quantity Added To Tank. Returns the total quantity of material added to the tank.

TankQtyRemoved (Tank ID)—Quantity Removed From Tank. Returns the total quantity of material removed from the tank.

Regulator variables

RegulatorMaxRate (Regulator ID)—Regulator Maximum Rate. Returns the maximum rate of flow allowed through the regulator. This variable is assignable.

RegulatorState (Regulator ID)—Regulator State. Returns the state of the regulator (1=Adding, -1=Removing, 0=Not In Use).

RegulatorRate (Regulator ID)—Regulator Rate. Returns the current rate of flow through the regulator.

RegulatorQtyAdded (Regulator ID)—Quantity Added By Regulator. Returns the total quantity of material added using the regulator.

RegulatorQtyRemoved (Regulator ID)—Quantity Removed By Regulator. Returns the total quantity of material removed using the regulator.

RegulatorTank (Regulator ID)—Tank Number Of Regulator. Returns the number of the tank with which the regulator is associated.

FlowRate (Source Regulator ID, Destination Regulator ID)—Flow Rate Between Regulators. Returns a 0.0 if there is no flow between the specified regulators. Returns a positive value if there is flow from the source regulator to the destination regulator, or a negative value if there is flow from the destination to the source.

Sensor variables

SensorLocation (Sensor ID)—Sensor Location. Returns the level location of the sensor. This variable is assignable.

SensorState (Sensor ID)—Sensor State Indicator. Returns whether the sensor is enabled or disabled (0=disabled, 1=enabled). This variable is assignable.

SensorTank (Sensor ID)—Tank Number Of Sensor. Returns the number of the tank with which the sensor is associated.

SensorIsCovered (Sensor ID)—Sensor Is Covered Indicator. Returns whether the sensor's location is above or below the tank's current level (0=No, 1=Yes). Note that this variable always returns 0 if the sensor is disabled (i.e., SensorState = 0).

Operators

The following table includes mathematical operators and logical operators supported by Arena. Standard math priority rules are used to evaluate complex equations.

Operator	Operation	Priority
Math Operators		
**	Exponentiation	1 (highest)

Operator	Operation	Priority
/	Division	2
*	Multiplication	2
–	Subtraction	3
+	Addition	3

Operator	Operation	Priority
Logical Operators		
.EQ. , ==	Equality comparison	4
.NE. , <>	Non-equality comparison	4
.LT. , <	Less than comparison	4
.GT. , >	Greater than comparison	4
.LE. , <=	Less than or equal to comparison	4
.GE. , >=	Greater than or equal to comparison	4
.AND., &&	Conjunction (and)	5
.OR., ||	Inclusive disjunction (or)	5

Note that == is a logical operator and = is an assignment operator. Use == to test whether two items have the same value; use = to set a value, as in the Assign module.

Math Functions

Arena provides 20 standard mathematical functions. Each function takes a parameter list enclosed in parentheses. These parameters may be specified as constants or expressions when used in a model.

Remarks

The geometric functions (ACOS, ASIN, ATAN, HCOS, HSIN, HTAN, COS, SIN, TAN) take a value specified in radians.

Function	Description
ABS(a)	Absolute value
ACOS(a)	Arc cosine
AINT(a)	Truncate
AMOD(a1 , a2)	Real remainder, returns (a1-(AINT(a1/a2)*a2))
ANINT(a)	Round to nearest integer
ASIN(a)	Arc sine
ATAN(a)	Arc tangent
COS(a)	Cosine
EP(a)	Exponential (e_a)
HCOS(a)	Hyperbolic cosine
HSIN(a)	Hyperbolic sine
HTAN(a)	Hyperbolic tangent
MN(a1 , a2 , ...)	Minimum value
MOD(a1 , a2)	Integer remainder, same as AMOD except the arguments are truncated to integer values first
MX(a1 , a2 , ...)	Maximum value
LN(a)	Natural logarithm
LOG(a)	Common logarithm
SIN(a)	Sine
SQRT(a)	Square root
TAN(a)	Tangent

SIMAN Constructs Variables

For each construct type, a variable is provided that returns the number of constructs defined in a simulation model. Additional variables return the number of blocks in the model, number of active entities, etc. Unless otherwise noted, these variables remain

constant throughout a simulation run. Each of these variables returns an integer quantity. They are not user-assignable.

Variable	Element Name or Description
MXARR	Number of ARRIVALS
MXASM	Number of Named Attribute Symbols
MXATT	Number of ATTRIBUTES
MXBKG	Number of BLOCKAGES
MXBLK	Number of Blocks in SIMAN model
MXCNT	Number of COUNTERS
MXCNV	Number of CONVEYORS
MXCST	Number of CSTATS
MXDSB	Number of DISTRIBUTIONS
MXDST	Number of DSTATS
MXENT	Maximum number of entities that can be active for a given RSET size
MXEXP	Number of EXPRESSIONS
MXFAL	Number of FAILURES
MXFIL	Number of FILES
MXFRQ	Number of FREQUENCIES
MXINX	Number of INTERSECTIONS
MXLEV	Number of LEVELS
MXLNK	Number of LINKS
MXNET	Number of NETWORKS
MXOUT	Number of OUTPUTS
MXPAR	Number of PARAMETERS
MXQUE	Number of QUEUES
MXRAT	Number of RATES
MXREC	Number of RECIPES
MXRES	Number of RESOURCES

Variable	Element Name or Description
MXRLN	Number of REPORTLINES
MXRPT	Number of REPORTS
MXRUL	Number of RULES
MXSCH	Number of SCHEDULES
MXSEE	Number of SEEDS
MXSEQ	Number of SEQUENCES
MXSET	Number of SETS
MXSTA	Number of STATIONS
MXSTR	Number of STORAGES
MXSTS	Number of STATICS
MXSTT	Number of STATESETS
MXTAB	Number of TABLES
MXTAL	Number of TALLIES
MXTRN	Number of TRANSPORTERS
MXVAR	Number of VARIABLES
MXVSM	Number of Named Variable Symbols

Summary Table of Variables

Attributes and entity-related variables

GENERAL ATTRIBUTES VARIABLES

Variable	Arguments	Description
oAttribute Name	Index 1, Index 2	General-purpose entity attribute
oA	Attribute Number [, Entity Number]*	General-purpose entity attribute
oEntity.Type	[Entity Number]	Entity-type attribute
oEntity.Picture	[Entity Number]	Entity animation attribute
Entity.SerialNumber	—	Entity serial number
oEntity.Jobstep	[Entity Number]	Entity jobstep (sequence index) attribute
oEntity.Sequence	[Entity Number]	Entity sequence attribute
oEntity.Station	[Entity Number]	Entity station location attribute
Entity.CurrentStation	—	Entity current station location attribute
Entity.PlannedStation	—	Entity next planned station location attribute

o Assignable

* Entity number is an optional argument that permits referencing and assigning attributes of remote entities.

TIME ATTRIBUTES VARIABLES

Variable	Arguments	Description
Entity.CreateTime	—	Entity creation time attribute
Entity.StartTime	—	Entity start time attribute
Entity.VATime	—	Entity value-added time attribute
Entity.NVATime	—	Entity non-value-added time attribute
Entity.WaitTime	—	Entity waiting time attribute
Entity.TranTime	—	Entity transfer time attribute
Entity.OtherTime	—	Entity other time attribute

COST ATTRIBUTES VARIABLES

Variable	Arguments	Description
[o]Entity.HoldCostRate	—	Entity holding cost rate
Entity.VACost	—	Entity valued-added cost attribute
Entity.NVACost	—	Entity non-valued-added cost attribute
Entity.WaitCost	—	Entity waiting cost attribute
Entity.TranCost	—	Entity transfer cost attribute
Entity.OtherCost	—	Entity other cost attribute

[o] Assignable

ENTITY-TYPE VARIABLES

Variable	Arguments	Description
EntitiesIn	Entity Type	Number of entities in
EntitiesOut	Entity Type	Number of entities out
EntitiesWIP	Entity Type	Number of entities in process
InitialPicture	Entity Type	Initial picture
InitialHoldCostRate	Entity Type	Initial hold cost rate
InitialVACost	Entity Type	Initial value-added cost
InitialNVACost	Entity Type	Initial non-value-added cost
InitialWaitCost	Entity Type	Initial waiting cost
InitialTranCost	Entity Type	Initial transfer cost
InitialOtherCost	Entity Type	Initial other cost

GROUP MEMBER VARIABLES

Variable	Arguments	Description
AG	Rank, Attribute Number	Group member attribute
ENTINGROUP	Rank [, Entity Number]*	Grouped entity number
GRPTYP	[Entity Number]	Group type
ISG	Rank	Grouped entity IS attribute
MG	Rank	Grouped entity M attribute
NSG	Rank	Grouped entity NS attribute
NG	[Entity Number]*	Number of grouped entities
SAG	Attribute Number	Sum of grouped entity attributes

* Entity number is an optional argument that permits referencing and assigning attributes of remote entities.

OTHER ENTITY VARIABLES

Variable	Arguments	Description
[o]ATTR	Attribute Number [, Index 1, Index 2]	Attribute value
IDENT	—	Active entity number
NUMENT	—	Number of active entities

Activity area variables

Variable	Arguments	Description
AreaVATime	Activity Area Name	Total value-added time
AreaNVATime	Activity Area Name	Total non-valued-added time
AreaTranTime	Activity Area Name	Total transfer time
AreaOtherTime	Activity Area Name	Total other time
AreaWaitTime	Activity Area Name	Total wait time
AreaTotalTime	Activity Area Name	Total time
AreaVACost	Activity Area Name	Total value-added cost
AreaNVACost	Activity Area Name	Total non-valued-added cost
AreaTranCost	Activity Area Name	Total transfer cost
AreaOtherCost	Activity Area Name	Total other cost
AreaWaitCost	Activity Area Name	Total wait cost
AreaTotalCost	Activity Area Name	Total cost

[o] Assignable

Event calendar variables

Variable	Arguments	Description
FIRSTINCAL	—	First entity on the calendar
NEXTINCAL	Entity Number	Next entity on the calendar

Continuous variables

LEVEL VARIABLES

Variable	Arguments	Description
oLevel Name	Index 1, Index 2	Named level variable
oS	Level Number	Level variable

RATE VARIABLES

Variable	Arguments	Description
oRate Name	Index 1, Index 2	Named rate variable
oD	Rate Number	Rate variable

o Assignable

Conveyor variables

GENERAL

Variable	Argument	Description
ICS	Conveyor ID	Conveyor status indicator (0 = idle, 1 = moving, 2 = blocked, 3 = inactive)
MLC	Conveyor ID	Conveyor length
[o]VC	Conveyor ID	Conveyor velocity

CONVEYING ENTITY VARIABLES

Variable	Argument	Description
CLA	Conveyor ID	Length of accumulated entities
LC	Conveyor ID	Number of occupied cells
LEC	Conveyor ID	Length of conveying cells
NEA	Conveyor ID	Number of accumulated entities
NEC	Conveyor ID	Number of conveying entities
CNVDST	Conveyor Number, Entity Number	Entity location on conveyor

[o] Assignable

Queue variables

GENERAL QUEUE VARIABLES

Variable	Arguments	Description
ENTATRANK	Rank, Queue ID	Entity number of queued entity
FIRSTINQ	Queue ID	First entity number in queue
LASTINQ	Queue ID	Last entity number in queue
NQ	Queue ID	Number in queue

QUEUED ENTITY VARIABLES

Variable	Arguments	Description
AQUE	Queue ID, Rank, Attribute Number	Attribute of queued entity
ISQUE	Queue ID, Rank	Sequence index attribute of queued entity
MQUE	Queue ID, Rank	Station attribute of queued entity
NSQUE	Queue ID, Rank	Sequence number attribute of queued entity
PREDECESSOR	Entity Number	Queued entity predecessor
SAQUE	Queue ID, Attribute Number	Sum of attributes of queued entities
SUCCESSOR	Entity Number	Queued entity successor

Resource variables

GENERAL RESOURCE VARIABLES

Variable	Argument	Description
IRF	Resource ID	Resource failure
oMR	Resource ID	Resource capacity
NR	Resource ID	Number of busy resource units
RESUTIL	Resource ID	Resource utilization
RESSEIZES	Resource ID	Number of seizes
RTYP	Resource ID	Resource type
LR	Resource ID	Resource location
oSTATE	Resource ID	Resource state
STATEVALUE	Resource Expr, StateString	Resource stateset
IDLE_RES	—	Idle resource state constant
BUSY_RES	—	Busy resource state constant
INACTIVE_RES	—	Inactive resource state constant
FAILED_RES	—	Failed resource state constant

Replication variables

Variable	Argument	Description
oMREP	—	Maximum replications
NREP	—	Replication number

o Assignable

Date and time variables

CALENDAR DATES AND TIMES VARIABLES

Variable	Argument	Description
CalYear	Expression	Simulation time in year with century
CalMonth	Expression	Simulation time as integer for calendar month integer
CallWeek	Expression	Simulation time as integer for calendar week of year
CalDayOfYear	Expression	Simulation time as integer for day of year
CalDayOfMonth	Expression	Simulation time as integer for day of month
CalHour	Expression	Simulation time as integer of hour of 24-hour day
CalMinute	Expression	Simulation time as integer of minute of calendar time
CalSecond	Expression	Simulation time as integer of second of calendar time
CalDateToBaseTime	Expression	Simulated time in base units corresponding to midnight of calendar date specified

CURRENT AND FINAL SIMULATION TIME VARIABLES

Variable	Argument	Description
[o]TFIN	—	Final simulation time
TNOW	—	Current simulation time

[o] Assignable

CONVERTING DURATIONS TO THE BASE TIME UNITS VARIABLES

Variable	Argument	Description
SecondsToBaseTime	Expression	Time value conversion
MinutesToBaseTime	Expression	Time value conversion
HoursToBaseTime	Expression	Time value conversion
DaysToBaseTime	Expression	Time value conversion

System response variables

THROUGHPUT VARIABLE

Variable	Argument	Description
Total.Throughput	—	Total entity throughput

COSTS VARIABLES

Variable	Argument	Description
Total.VACost	—	Total value-added cost
Total.NVACost	—	Total non-value-added cost
Total.WaitCost	—	Total waiting cost
Total.TranCost	—	Total transfer cost
Total.OtherCost	—	Total other cost
Total.EntityCost	—	Total entity cost
Total.ResUseCost	—	Total resource usage cost
Total.ResBusyCost	—	Total resource busy cost
Total.ResIdleCost	—	Total resource idle cost
Total.ResourceCost	—	Total resource cost
Total.SystemCost	—	Total system cost

Statistics collection variables

COUNTER STATISTICS VARIABLES

Variable	Argument	Description
[o]MC	Counter ID	Count limit
NC	Counter ID	Count value

[o] Assignable

TIME-PERSISTENT STATISTICS (CSTAT) VARIABLES

Variable	Argument	Description
CAVG	Cstat ID	Average value
CMAX	Cstat ID	Maximum value
CMIN	Cstat ID	Minimum value
CSTD	Cstat ID	Standard deviation
CTPD	Cstat ID	Time period
CHALF	Cstat ID	Half width
CVALUE	Cstat ID	Last recorded value
CBATCH	Cstat ID	Average value in batch
CNUMBAT	Cstat ID	Number of batches
CBATSIZ	Cstat ID	Batch size

TIME-PERSISTENT STATISTICS (DSTAT) VARIABLES

Variable	Argument	Description
DAVG	Dstat ID	Average value
DMAX	Dstat ID	Maximum value
DMIN	Dstat ID	Minimum value
DSTD	Dstat ID	Standard deviation
DTPD	Dstat ID	Time period
DHALF	Dstat ID	Half width
DVALUE	Dstat ID	Last recorded value
DBATCH	Dstat ID	Average value in batch
DNUMBAT	Dstat ID	Number of batches
DBATSIZ	Dstat ID	Batch size

FREQUENCIES STATISTICS VARIABLES

Variable	Arguments	Description
FAVG	Frequency ID, Category	Average time in category
FCATS	Frequency ID	Number of categories
FCOUNT	Frequency ID, Category	Frequency category count
oFHILIM	Frequency ID, Category	Frequency category high limit
oFLOLIM	Frequency ID, Category	Frequency category low limit
FSTAND	Frequency ID, Category	Standard category percent
FRQTIM	Frequency ID, Category	Time in category
FRESTR	Frequency ID, Category	Restricted category percent
FTOT	Frequency ID	Total frequency time
FTOTR	Frequency ID	Restricted frequency time
FVALUE	Frequency ID	Last recorded value

o Assignable

TALLY STATISTICS VARIABLES

Variable	Argument	Description
TAVG	Tally ID	Average value
TMAX	Tally ID	Maximum value
TMIN	Tally ID	Minimum value
TNUM	Tally ID	Number of observances
TSTD	Tally ID	Standard deviation
THALF	Tally ID	Half width
TVALUE	Tally ID	Last recorded value
TBATCH	Tally ID	Average value in batch
TNUMBAT	Tally ID	Number of batches
TBATSIZ	Tally ID	Batch size

OUTPUT STATISTICS VARIABLE

Variable	Argument	Description
OVALUE	Output ID	Last recorded value

POST-RUN STATISTICS VARIABLES

Variable	Argument	Description
ORUNAVG	Output ID	Average value
ORUNMAX	Output ID	Maximum value
ORUNMIN	Output ID	Minimum value
ORUNHALF	Output ID	Half width

Transporter variables

GENERAL STATUS VARIABLES

Variable	Arguments	Description
oIT	Transporter ID, Unit Number	Transporter unit status
MT	Transporter ID	Number of active units
NT	Transporter ID	Number of busy units
oVT	Transporter ID	Transporter set velocity
oVTU	Transporter ID, Unit Number	Transporter unit velocity

FREE-PATH TRANSPORTER VARIABLES

Variable	Arguments	Description
ID	Transporter ID, Station ID	Transporter distance
IDIST	Distance Set ID, Beginning Station ID, Ending Station ID	Distance set value
LT	Transporter ID, Unit Number	Transporter location

o Assignable

GUIDED TRANSPORTER VARIABLES

Variable	Arguments	Description
ACC	Transporter ID	Acceleration
DEC	Transporter ID	Deceleration
ISZT	Transporter ID, Unit Number	Size type
LDL	Transporter ID, Unit Number	Destination link
LDX	Transporter ID, Unit Number	Destination intersection
LDZ	Transporter ID, Unit Number	Destination zone number
LT	Transporter ID, Unit Number	Intersection location
LTL	Transporter ID, Unit Number	Link location
LTZ	Transporter ID, Unit Number	Zone location
NSZT	Transporter ID, Unit Number	Transporter size value
TAZ	Transporter ID, Unit Number	Zone arrival time
TVF	Transporter ID	Turning velocity factor
TWZ	Transporter ID, Unit Number	Time waiting in zone

GUIDED NETWORK VARIABLES

Variable	Arguments	Description
INXNUM	Station ID	Intersection number
IDSNET	Network ID, Beginning Intersection ID, Ending Intersection ID	Network distance
LENZ	Link ID	Zone length
LNKNUM	Beginning Intersection ID, Ending Intersection ID	Connecting link
LTYP	Link ID	Link type
LX	Intersection ID	Intersection length
MZ	Link ID	Number of zones
NDX	Link ID	Destination intersection
NEXTX	Network ID, Beginning Intersection ID, Destination Intersection ID	Next travel intersection
NL	Link ID	Number of occupied zones in link
NX	Intersection ID	Intersection status
NXB	Link ID	Beginning intersection
NXE	Link ID	Ending intersection
NZ	Link ID, Zone Number	Zone status
VL	Link ID	Link velocity factor
VX	Intersection ID	Intersection velocity factor

Miscellaneous variables

BLOCKAGE STATUS VARIABLE

Variable	Argument	Description
NB	Blockage ID	Current blockage quantity

EXPRESSIONS VARIABLES

Variable	Arguments	Description
ED	Expression Number	Expression value
EXPR	Expression Number [Index 1, Index 2]	Expression value
Expression Name	Index 1, Index 2	Expression value

FUNCTIONS VARIABLES

Variable	Arguments	Description
NSYM	Symbol Name	Symbol number
NUMBLK	[Block Label]	Block number
TF	Table ID, X Value	Table function value
UF	User Function Number	User-coded function
STR	(constructType, constructNum [,subConstructNum])	Symbol name

GENERAL-PURPOSE GLOBAL VARIABLES

Variable	Arguments	Description
oV	Variable Number	Variable value
oVAR	Variable Number [Index 1, Index 2]	Variable value
oVariable Name	Index 1, Index 2	Variable value

PARAMETERS VARIABLES

Variable	Arguments	Description
CO	Parameter Set ID	Constant value
NMPAR	Parameter Set ID	Number of parameter values
oP	Parameter Set ID, Parameter Number	Parameter value

RESOURCE COST VARIABLES

Variable	Argument	Description
NSEXPO	Schedule ID	Non-stationary exponential distribution
SchedValue	Schedule ID	Schedule value

o Assignable

J INDEX VARIABLE

Variable	Argument	Description
ₒJ	—	Search index variable

SET VARIABLES

Variable	Arguments	Description
MEMBER	Set ID, Index	Set member
MEMIDX	Set ID, Member ID	Member index in set
NUMMEM	Set ID	Number of members

[o] Assignable

STATION VARIABLES

Variable	Arguments	Description
INXNUM	Station ID	Intersection number
MSQ	Sequence ID, Sequence Index	Sequence station
NE	Station ID	Number of entities transferring
StnVATime	Station Name	Station total value added time
StnNVATime	Station Name	Station total non-value added time
StnTranTime	Station Name	Station total transfer time
StnOtherTime	Station Name	Station total other time
StnWaitTime	Station Name	Station total wait time
StnTotalTime	Station Name	Station total time
StnVACost	Station Name	Station total value added cost
StnNVACost	Station Name	Station total non-value added cost
StnTranCost	Station Name	Station total transfer cost
StnOtherCost	Station Name	Station total other cost
StnWaitCost	Station Name	Station total wait cost
StnTotalCost	Station Name	Station total cost

STORAGE VARIABLE

Variable	Argument	Description
NSTO	Storage ID	Number of entities in storage

STACK VARIABLES

Variable	Arguments	Description
Diff.StartTime	—	Difference in saved start time
Diff.VATime	—	Difference in value-added time
Diff.VACost	—	Difference in value-added cost
Diff.NVATime	—	Difference in non-value-added time
Diff.NVACost	—	Difference in non-value-added cost
Diff.WaitTime	—	Difference in waiting time
Diff.WaitCost	—	Difference in waiting cost
Diff.TranTime	—	Difference in transfer time
Diff.TranCost	—	Difference in transfer cost
Diff.OtherTime	—	Difference in other time
Diff.OtherCost	—	Difference in other cost

OPERATIONPARAMETER VARIABLE

Variable	Argument	Description
OpParamVal	OperationParameter Name	Operation Parameter value

Flow variables

TANK VARIABLES

Variable	Argument	Description
[o]TankCapacity	Tank ID	Tank capacity
[o]TankLevel	Tank ID	Tank level
TankNetRate	Tank ID	Tank net rate
TankQtyAdded	Tank ID	Quantity added to tank
TankQtyRemoved	Tank ID	Quantity removed from tank

REGULATOR VARIABLES

Variable	Argument	Description
[o]RegulatorMaxRate	Regulator ID	Regulator maximum rate
RegulatorState	Regulator ID	Regulator state
RegulatorRate	Regulator ID	Regulator rate
RegulatorQtyAdded	Regulator ID	Quantity added by regulator
RegulatorQtyRemoved	Regulator ID	Quantity removed by regulator
RegulatorTank	Regulator ID	Tank number of regulator
FlowRate	Source Regulator ID, Destination Regulator ID	Flow rate between regulators

[o] Assignable

SENSOR VARIABLES

Variable	Argument	Description
[o]SensorLocation	Sensor ID	Sensor location
[o]SensorState	Sensor ID	Sensor state indicator
SensorTank	Sensor ID	Tank number of sensor
SensorIsCovered	Sensor ID	Sensor is covered indicator

[o] Assignable

2 Strings in Arena

Introduction

In Arena, a string is text composed of a sequence of zero or more characters. A string can contain alphabetic characters, numbers, and symbols. Arena supports single-byte characters and thus supports the standard ASCII character set and the extended ASCII characters. Unicode strings are not supported.

A string value may be specified in any Arena expression using one of the following approaches:

String Value	Example
String constant enclosed in quotation marks (“ ”)	“Blue”
String variable name[(row, column)]	strVar==“Blue”
String attribute name[(row, column)]	strAttr==“Blue”
Result from a string function	Str(RESOURCES,3), StrFormat(“Number In Queue = %f”, NQ(MyQueue)), Chr(65), Str(TNOW)

Note: You cannot specify a ` or ^ character within a quoted string. Instead, you must use Chr function for these characters.

The Chr function may be used to return the character for an integer ASCII character code. For more information, see “Chr function” on page 70.

Because the quotation mark character (“) is used to delimit strings in an expression, you can’t use it directly to specify a quote within a string. Instead, place two consecutive quotes to specify a quote in a string (e.g.,):

```
“The resource name is ““Fred””.”
```

Or build a string with imbedded quotes using the character code for a quote (34):

```
StrFormat(“The resource name is %cFred%c.”,34,34)
“The resource name is “ + Chr(34) + “Fred” + Chr(34) + ”.”
```

String/numeric conversions

During a simulation run, an expression evaluated in the model's logic or animation may be expected to return either a numeric or string value depending on the context in which it is used.

Arena's SIMAN simulation language is strongly typed and does not support implicit data type conversions between numbers and strings. All data type conversions must be explicitly performed using the Str and Val conversion functions. For more information, see "Str function" on page 67 and "Val function" on page 68.

If an expression is evaluated and the data type of the value returned by the expression is different than the expected data type, then a "Data type mismatch" runtime error will occur.

Comparing strings

Two strings may be compared in an expression using the standard logical operators:

Logical Operator	Description
EQ. , = =	Equality comparison
.NE. , <>	Non-equality comparison
.LT. , <	Less than comparison
.GT. , >	Greater than comparison
.LE. , <=	Less than or equal to comparison
.GE. , >=	Greater than or equal to comparison

Strings are compared by character using character codes (a case sensitive comparison). A logical expression returns 1 if True and 0 if False.

For example, the logical expression strColor= ="Blue" will return true (1) if the string variable strColor contains the value "Blue". Otherwise, the expression will return false (0).

A "Data type mismatch error" occurs if an attempt is made to logically compare a string value to a real number.

The StrCompare function may also be used to perform string comparisons. In particular, this function is useful in that it provides an option for performing textual comparisons that are not case sensitive. For more information, see "StrCompare function" on page 69.

Building strings

The addition operator (+) may be used as a string operator that concatenates the two strings. For example, the expression "1" + "5" will return the string "15".

The StrFormat function is also available for building and returning a string in a specified format. For more information, see "StrFormat function" on page 69.

Str function

The Str function may be used to convert a numeric expression to a string, or to return the symbol name of a simulation construct.

Str*(*Numeric Expression*)—*String Conversion. This form of Str may be used to convert a numeric expression to a string.

Examples:

- **Str(123)** returns "123"
- **Str(1+3.2)** returns "4.2"

Str*(*Element Type, Element Symbol Number* [,*subConstructNum*])—*Element Symbol Name. This form of Str may be used to return the symbol name of a simulation construct.

When using Str in this form, the function's arguments are defined as follows:

- Element Type = the element type (FILES, RESOURCES, etc.) or a keyword (see list below)
- Element Symbol Number = number of the construct within the element (for example, use 5 for the 5th resource)
- subConstructNum = number of construct within another (for example, a step within a sequence element or a state within a stateset element)

In addition to element names for Element Type, the following keywords may be used to return some additional special-purpose information:

- ANALYST – Analyst name
- BASEFILE – Program file name without extension
- COMPANY – Company name
- MODDATE – Model revision date
- PRGFILE – Program file name with extension
- PRJNAME – Project name

- RUNDATE – Execution date
- STRING – User-defined string
- TIME – Current system time

Only the STRING keyword requires the second (Element Symbol Number) argument. Enter a string argument.

Examples:

- **Str(RESOURCES,5)** returns the symbol name of the 5th resource element defined in the model.
- **Str(ANALYST)** returns the analyst name specified for the model.
- **Str(STATESETS,2)** returns the symbol name of the 2nd stateset element defined in the model.
- **Str(STATESETS,2,1)** returns the name of the first state specified in the 2nd stateset element defined in the model.

Val function

Val*(*String Expression*)—*Numeric Conversion. The Val function may be used to convert a string value to a real number. All blank spaces, line feed characters, and tab characters are first stripped from the string argument. An attempt is then made to convert as much of the string into a number as possible. The conversion stops when a character in the string is not recognized as part of a number. Symbols and characters that are often considered parts of numeric values, such as dollar signs and commas, are not recognized. Only a period (.) is recognized as a valid decimal separator.

If none of the string is recognized as a numeric value, then a 0 is returned for that string.

Examples:

- **Val(“123”)** returns 123
- **Val(“1.3”)** returns 1.3
- **Val(“123abc”)** returns 123
- **Val(“3E-3”)** returns 0.003
- **Val(“junk”)** returns 0
- **Val(“1 2 3”)** returns 123
- **Val(“123abc123”)** returns 123
- **Val(“$100”)** returns 0

- **Val(“1,234”)** returns 1

StrCompare function

***StrCompare*(*StringExpression1, StringExpression2*[,*CompareOption*])**—String Comparison. The StrCompare function returns -1 if StringExpression1 is less than StringExpression2, 0 if StringExpression1 equals StringExpression2, and 1 if StringExpression1 is greater than StringExpression2.

If the CompareOption value is specified as 0 then a binary, case sensitive string comparison is performed using character codes. Otherwise, the string comparison is textual. A binary comparison is performed by default.

Examples:

Comparison	Result
StrCompare("A", "B")	-1 (string1 is less than string2)
StrCompare("A", "A")	0 (string1 is equal to string2)
StrCompare("B", "A")	1 (string1 is greater than string2)
StrCompare("ABC", "abc", 1)	0 (string1 is equal to string2)
StrCompare("ABC", "abc")	-1 (string1 is less than string2)

StrFormat function

StrFormat*(*Format*[,*Parameters*])—*Formatted String. The StrFormat function returns a formatted string value.

The arguments for StrFormat are defined as follows:

- Format = A valid C-style format string
- Parameters = Optional format parameters. A parameter may be any valid Arena expression. Multiple format parameters are separated by commas.

Examples:

StrFormat Expression	Returns
StrFormat(“Hi %c %d %s”, 65, 10, “there!”)	“Hi A 10 there!”
StrFormat(“The value is %06.2f”,1.23434)	“The value is 001.23”

StrFormat Expression	Returns
StrCompare("B", "A")	1 (string1 is greater than string2)
StrCompare("ABC", "abc", 1)	0 (string1 is equal to string2)
StrCompare("ABC", "abc")	-1 (string1 is less than string2)
StrFormat(“The value is %6.2f”,1.23434)	“The value is 1.23”
StrFormat(“The value is %-6.2f”,1.23434)	“The value is 1.23 ”
StrFormat(“The value is %.5s”, “abcdefghi”)	“The value is abcde”
StrFormat(“The value is %6s”, “abc”)	“The value is abc”
StrFormat(“The value is %-6s”, “abc”)	“The value is abc ”

Chr function

Chr*(*Character Code*)—*ASCII Character. The Chr function returns a single byte character string corresponding to the specified ASCII Character Code (0-255). The Character Code argument may be specified as an expression truncated to an integer.

Examples:

Character Code	Returns
34	“
96	`
94	^
65	A
66	B

Eval function

The Eval function is used to evaluate a simulation expression contained in a string argument.

Eval(String Expression)—Evaluate String Expression. Evaluates the simulation expression contained in the string argument String Expression and returns the result.

Examples:

EVAL Expression	Returns
EVAL("2+2+2")	6 The value of the simulation expression, 2+2+2.
EVAL("TNOW")	The value of the simulation expression, TNOW.
EVAL(StrDestStation) where StrDestStation is a string attribute with current value "Station 1"	The value of the simulation expression, Station 1.
EVAL("StrDestStation") where StrDestStation is a string attribute with current value “Station 1"	"Station 1" The value of the simulation expression, StrDestStation.
EVAL("NSYM(" + StrDestStation + ")") where StrDestStation is a string attribute with current value “Station 1”	The value of the simulation expression, NSYM(Station 1).
EVAL("""Hello"" + ""World""")	"HelloWorld" The value of the simulation expression, "Hello" + "World".

Mid function

The Mid function is used to return a sub-string from a string expression.

Mid(String Expression, Start[, Length]) – Return sub-string from string expression. Returns a sub-string from String Expression, starting at Start, of length Length. If Length is not specified, the sub-string is taken to the end of String Expression.

Examples:

MID Expression	Returns	Comments
MID ("Fred",2)	"red"	Starting at the second character in the string "Fred", the MID function returns three characters, that is "red."
MID ("Fred", LEN ("Fred") - 1)	"ed"	The expression LEN ("Fred") - 1 returns 3. Thus, starting at the third character in the string "Fred", the MID function returns two characters, that is "ed."

Len function

The Len function is used to calculate the length of a string expression.

Len(String Expression) – Length of String Expression. Returns the length of the string value of String Expression.

Examples:

LEN Expression	Returns
LEN ("Fred")	4
LEN ("Fred" + "Jones")	9

Index

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V