Event	Abbr.	Occurs when	Notes
entered(S)	en(S)	State S is entered.	This event is used in statecharts only.
exited(S)	ex(S)	State S is exited.	This event is used in statecharts only.
entering	ns	The state is being entered.	This event is used only as a trigger of a reaction in the state.
exiting	xs	The state is being exited.	This event is used only as a trigger of a reaction in the state.
started(A)	st(A)	Activity A is started.	This event is used in statecharts only.
stopped(A)	sp(A)	Activity A is stopped.	This event is used in statecharts only.
started	st	The activity is started	This event is used only as a trigger of a reaction in a reactive activity.
changed(X)	ch(X)	The value of X is changed.	X is a data-item or condition expression or array, which can be an array slice. It can not be structered.
true(C)	tr(C)	This event means that the value of condition C is changed to TRUE.	C can be any condition expression but not an array.
false(C)	fs(C)	This event means that the value of condition C is changed to FALSE.	C can be any condition expression but not an array.
read(X)	rd(X)	X is read by the action read_data.	X is a primitive data-item or condition. X cannot be an alias. X can be an array, not an array slice. X can be an array component, not a bit-array component.
written(X)	wr(X)	X is written by the action write_data or by assignment.	X is a primitive data-item or condition. X cannot be an alias. X can be an array, not an array slice. X can be an array component, not a bit-array component.
timeout(E,N)	tm(E,N)	N clock units are passed from the last time-instant the event E occurred.	E is an event expression, not an array. N is a numeric expression.
all(E)	all(E)	All components of event array E occurred.	E is an event array.
any(E)	any(E)	At least one component of event array E occurred.	E is an event array.

Compound Event	Occurs when
E[C]	E has occurred and the condition C is true.
[C]	Condition C is true.
not E	E did not occur.
E1 and E2	Both E1 and E2 occurred simultaneously.
E1 or E2	Either E1 or E2, or both occurred.

Atomic Conditions	
Constant Literal	A constant literal is either TRUE or FALSE. These are not case sensitive.
Named Single Condition	The named single condition cannot be an array.
C(K)	This expression indicates the K'th component of a condition array C. K can be any integer expression.

Arrays of Conditions	**Notes**
Constant Literal {C1,C2,..., K*CN,...,*CL}	In this example, each Ci is a constant literal condition; K is a constant literal integer.
Named Condition Array	
Array Slice C(K..L)	This example represents an array slice C(K..L) of an array condition C; K and L are integer expressions.

Condition	**Abbr.**	**True when**	**Notes**
in(S)	in(S)	System is in state S.	This condition is used only in statecharts.
active(A)	ac(A)	Activity A is active.	This condition is used only in statecharts.
hanging(A)	hg(A)	Activity A is suspended.	This condition is used only in statecharts.
X1 R X2		The value of X1 and X2 satisfy the relation R.	X1 and X2 are dataitems or condition expressions. When numeric, R can be: =, /=, >, <, >=, <=. When strings, conditions, or arrays, R can be: =, - X1 and X2 are nnot structured
all(C)		All components of condition array C are true.	C is a condition array.
any(C)		At least one component of condition array C is true.	C is a condition array.

Compound Condition	**True when**
not C	C is not true.
C1 and C2	Both C1 and C2 are true.
C1 or C2	Either C1 or C2 are true.

Action	Abbr.	Results	Notes
E		Generate an event E.	E can be a primitive single event, *not* an array element.
make_true(C)	tr!(C)	The value of condition C is made true.	C can be a primitive single event, *not* an array element.
make_false(C)	fs!(C)	The value of condition C is made false.	C can be a primitive single event, *not* an array element.
X:=EXP		Assigns the value of data-item or condition or expression EXP to X.	X is a primitive or alias data-item, array or single, condition or array condition including slices.
start(A)	st!(A)	Start activity A.	This can be used only in statecharts.
stop(A)	sp!(A)	Stop activity A.	This can be used only in statecharts.
stop		Stop the activity.	This can be used only in mini-specs of a reactive activity.
suspend(A)	sd!(A)	Suspend activity A. This results in making condition hanging(A), true.	This can be used only in statecharts.
resume(A)	rs!(A)	Resume activity A. This results in making condition active(A), true.	This can be used only in statecharts.
read_data(X)	rd!(V)	Read the data-item or condition X.	X is a primitive data-item or condition, array or array slice. Bit-array components or slices are not allowed.
write_data(X)	wr!(X)	Write the data-item X.	X is a primitive data-item or condition, array or array slice. Bit-array components or slices are not allowed.
history_clear(S)	hc!(S)	Clear the history information for state S.	This can be used only in statecharts.
deep_clear(S)	dc!(S)	Clear the history information for all decendants of state S.	This can be used only in statecharts.
schedule(A,N)	sc!(A,N)	Performs the action A delayed by N clock units.	N is a numeric expression.

Compound Action	Notes
A1;A2	The actions are performed sequentially. The semi-colon (;) is optional at the end of the list.
if C then A1 else A2 end if	Else-*part* of the statement is optional.
when E then A1 else A2 end when	E is an event expression. Else-*part* of the statement is optional.
for $I in K (to | downto) L loop A end loop	$I is a context variable. K, L are integer expressions. A is an action expression.
while C loop A end loop	C is a condition expression. A is an action expression.
break	break causes the inner most nested loop action to terminate.

Function	**Result**
min(list of integers and reals)	Returns minimum value.
max(list of integers and reals)	Returns maximum value.
rem(integer, integer)	Returns remainder.
abs(integer)	Returns absolute value.
bnand(integer,integer)	Returns result of bit-nand.
bnor(integer,integer)	Returns result of bit-nor.
bxor(integer,integer)	Returns result of bit-exclusive or.
band(integer,integer)	Returns result of bit-and.
bnot(integer)	Returns result of bit-not.
bor(integer,integer)	Returns result of bit-or.
trunc(real)	Returns truncated real number.
round(real)	Returns rounded real number.

Abbrev.	**Full**	**Result**
put!(q,x)	q_put	Adds the value of the expression x to the tail of queue q.
uput!(q,x)	q_urgent_put	Adds to the queue's head element.
get!(q,x[,s])	q_get	Gets x from the head of q and sets s to true. s is false if q is empty.
peek!(q,x[,s])	q_peek	Copies the queue's head without removing it, and return status condition s.
fl!(q)	q_flush	Totally clears the queue.