* Builtin PREDICATION

  Unlike a library module, an builtin predicate is called without specifying a liblary-module name.

<alt PRED...>

Alternative predicate execution is shown.
Even if the results of Argument PRED are true, false, and unknown, it is certainly set to true.
It expresses [PRED...] as substitution.It uses by syntax analysis as [] of the method of EBNF.

<assert CLAUSE>
<asserta CLAUSE>
<assertz CLAUSE>

CLAUSE is added.
asserta is added to a head and assertz is added at the end.
assert is the same as asserta and is added to a head.

<cd PATH>

A current directory is moved to the PATH specified by the argument.

<compare EXPRESSION>
<comparef EXPRESSION>

EXPRESSION is evaluated.
compare is an integer and comparef is comparison of a float number.

<dir>

A current directory is displayed.

<edit FILE-NAME>

A FILE-NAME is edited using the editor set as the environment variables DEDITORPATH and EDITOR.

<erase CLAUSE-NAME>

All the programs, variables and objects corresponding to CLAUSE-NAME are deleted.

<exit>

Execution is interrupted on the way.

<f VAR PRED...>
<func VAR PRED...>

Execution evaluation of the function predicate contained
in PRED-LIST is carried out. A value is replaced.
After performing all, a result is set to a variable.

<findall PRED...>

PRED is performed and all the solutions are calculated.

<for (VAR lastvalue) PRED...>
<for (VAR initialvalue lastvalue) PRED...>

The predicate of the specified number of times and an
argument is performed.
When the initial value is not specified, the value
from 0 is set to a VAR.

<foreach (VAR LIST) PRED...>

PRED of arguments are performed for every element
of LIST.
The value of a list is set to a variable in order.

<include FILENAME>

The library of a FILE-NAME is read.

<help>
<help PRED>

HELP manual is displayed.
If it performs without an argument, the list of an built-in
predicate and the predicates of the library of a sys module
will be displayed.

<let VAR = EXPRESSION>
<letf VAR = EXPRESSION>
<letc VAR = EXPRESSION>

Expression is calculated.
A calculation result is substituted when the left
side is a variable.
When the left side is a numerical value, it is
judged whether it is equal to a calculation result.
let calculates an integer and letf is calculation
of floating number.
letc calculates a complex number.

<list [VAR]>

A program list is displayed.

<load FILE-NAME>

The program of a FILE-NAME is read.

<loop PRED...>

A repetition of predicate execution is shown.
It was called inside. Failure of a predicate
will escape from a loop.
It is the predicate which is surely successful
by true,
It expresses {PRED...} as substitution.
It uses by syntax analysis as {} of the method
of EBNF.

<ls>

A current directory is displayed.

<map (VAR LIST) PRED...>

PRED of arguments are performed for every element
of LIST.
The value of a list is set to a variable in order.

<module VAR>

The library module used now is set to a variable.

<new>

All log rum is cleared.
<not PRED...>
false is returned when predicate execution is true.
In false, true is returned.
In unknown, unknown is returned.

<obj OBJECTNAME PRED...>

A predicate is performed using the specified object
or module.
A module is the same as an object.
It can be described as ::object <PRED...> as an
omitted type.

<or PRED PRED PRED ...>

In the place which performed the predicate of the
argument from the head and was set to true Processing
is closed and true is returned.
In the place which performed the predicate of the argument
from the head and was set to false Processing is closed
and false is returned.
unknown is returned when all the predicates are unknown.
It uses by syntax analysis as | of the method
of EBNF.

<print LIST>

After outputting a list, a new line is started.
<writenl LIST> is the same as <print LIST>.

<printf LIST>

LIST is outputted.
The differences from print are a blanks not entering between the elements of a list, and not starting a new line automatically.

<printlist LIST>

A parenthesis is removed and LIST is outputted.

<printlistnl LIST>

LIST is outputted removing a parenthesis and starting a new line for every element.

<pwd [VAR]>

The current directory is set to VAR.

<quit>

Execution of a program is stopped and it ends.

<quote PRED>

Evaluation of the function predicate of an argument is deterred.

<retract CLAUSE-NAME>

All the programs, variables and objects corresponding to CLAUSE-NAME are deleted.

<retractpred HEAD-NAME>

All the programs corresponding to HEAD-NAME are deleted.

<rpn VAR RPNEXPRESSION>
<rpnf VAR RPNEXPRESSION>
<rpnc VAR RPNEXPRESSION>

A reverse Poland style is calculated and a result is set to a variable.
rpn calculates an integer and rpnf calculates floating number.
rpnc calculates a complex number.

<save FILE-NAME>

The program of a FILE-NAME is written in.

<self VAR>

An own object name is set as a variable.

<super VAR>

LIST of the object name inherited to VAR is set up.

<timeout limittime PRED>

Within the set-up time, when execution of a predicate is not completed, processing is closed, and unknown is returned. A set period is a micro second bit.

<troff>

Debugging trace is turned OFF.

<tron>

Debugging trace is turned ON.

<true>
<false>
<unknown>

return true, false, unknown

<unify MODULENAME PRED...>

A predicate is performed using the specified object or module.
A module is the same as an object.
It can be described as ::object <PRED...> as an omitted type.

<warn LIST>

LIST is outputted to an stderr error output.
A new line is started after an output.

<x PRED...>

Usually, nothing is performed.
The predicate of an argument is performed when it backtracks.

<!>

cut operator

<% _ FORMAT ARG>

A format is operated for printf.

<¥ _ CHAR>

Escape character

EBNF Syntax analysis

<TOKEN VAR PRED...>

After syntax-analysis PRED execution of an input,
obtained token is set as VAR.

<SKIPSPACE>

The space of an input is skipped.

<C [VAR]>

An input is set as one-character VAR.

<N [VAR]>

When an input is a number, it is set as VAR.
unknown is returned when different.

<A [VAR]>

When an input is the ASCII character, it is set as VAR.
unknown is returned when different.

<AN [VAR]>

When an input is the ASCII character or a number,
it is set as VAR. unknown is returned when different.

<^>

The head of a line is matched.

<$>

The last of a line is matched.

<* [VAR]>

Arbitrary character strings are matched.

<CR>

true is returned when an input is a CR.
unknown is returned when different.

<CNTL [VAR]>

When an input is the CNTL character, it is set as VAR.
unknown is returned when different.

<EOF>

true is returned when an input is a EOF(End Of FIle).
unknown is returned when different.

<SPACE>

true is returned when an input is a space. unknown is
returned when different.

<PUNCT [VAR]>

true is returned when it is characters other than the alphabet and a number.

<WORD [VAR]>

In STRINGS(s) other than the alphabet, a number, and _, unknown is returned by arbitrary STRINGS.

<NUM [VAR]>

The integer of an input is is set as VAR.

<FNUM [VAR]>

The floating point number of an input is set as VAR.

<GETTOKEN VAR>

The token which is a result of the last syntax analysis is set as VAR.

<ID [VAR]>

If an input STRINGS (a head is the alphabet and a number is also good except it) and it agrees, it will be set as VAR.

<RANGE VAR CHAR1 CHAR2>
<NONRANGE VAR CHAR1 CHAR2>

It will be set to true if contained in the range of a character 1 and a character 2.

<NULLLINE>

It is judged whether it is a null line.

<SKIP STR>

Syntax analysis to STR is made to stop and skip.

<SKIPCR>

Syntax analysis to CR is made to stop and skip.

OBJECT operation
<newObj NAME>

create a new NAME object.

<cloneObj NEWOBJECT NAME>

The object of NAME is reproduced and NEWOBJECT is generated.

<delObj NAME>

delete NAME objects.

<method NAME>

The method of a new name is generated

<methoda NAME>

The method of a new name is generated from a head.

<methodz NAME>

The method of a new name is generated at the end.

<delmethod NAME>

The method of the specified name is deleted.

<delmethoda NAME>

The method of the specified name is deleted from a head.

<delmethodz NAME>

The method of the specified name is deleted from an end.

<setVar VAR VAL>

A value is assigned to global VAR. Global VAR
is registered in the form of the following as PRED.
<VAR VAL>;

<delVar VAR>

The variable corresponding to a VAR identifier
is deleted.

<setArray VAR VAL INDEX>

A value is set as global VAR arrangement.
Global VAR is registered in the form of the
following as PRED.
<VAR VAL INDEX>;

<delArray VAR INDEX>

The variable corresponding to a VAR identifier
is deleted.

<VAR VAL INDEX>;

When carrying out specification of the arrangement
of many dimensions, an index is specified as LIST.

<VAR VAL (INDEX1 INDEX2 ...)>

Indexes are a number, a character string, and LIST.