Oasys Manual

Klaus Didrich

1 Introduction

From version 2.3a onwards, the Opal compilation system provides an alternative to the `ocs` command for developing Opal applications: the `oasys` tool. Oasys supports functionality similar to `ocs`, but extends the latter's possibilities by incorporating an interpreter-debugger which allows expressions to be evaluated 'on the fly', and the execution of functions to be monitored.

Oasys was developed by Wolfgang Grieskamp and Eckard Lehmann and the latter programmed the interpreter-debugger as part of his master's thesis. Further improvements were added by Klaus Didrich.

Note: This manual is still incomplete as regards the debugger functionality.

2 Configuring Oasys

2.1 Global Configuration

The global configuration normally need not be edited by the user.

The global configuration is a Tcl script which is read when Oasys is invoked. It is looked for at the following locations:

1. '`$OASYSGLOBALRC`'
2. '`$OASYSLIB/startup.oasys`'
3. '`$OCSDIR/lib/oasys/startup.oasys`'
4. '`/usr/ocs/lib/oasys/startup.oasys`'

2.2 User Configuration

A user may customize Oasys. The user configuration is a Tcl script which is read after the global configuration has been read. It is read from one of the following locations:

1. '`$OASYSRC`'
2. '`$HOME/.oasysrc`'

The user configuration may contain any Oasys or Tcl commands.

2.3 Project Configuration

After having read the global and the user configuration, Oasys tries to read the project configuration from the file '`config.oasys`' in the current directory.

Whenever a directory is added to the search path of Oasys (see Section 5.17 [oasys-path], page 15), the '`config.oasys`' file of that directory will be loaded if present.

2.4 Defining New Oasys Commands

The Oasys command-line interpreter is a Tcl interpreter, so the user can use any feature of Tcl when writing global or local configuration files.

For defining new procedures, however, the Oasys command `oasys-proc` (see Section 5.19 [oasys-proc], page 16) should be used. This has two advantages: input of unit names or file names is supported by completion and the procedure is registered with the Oasys help system.

As a short example, we present the definition of a command `A`, which registers all units in the current directory (see Section 4.1 [a], page 6 for the simple variant).

```
oasys-proc {A} \
           {Add all units in current directory.} \
           {CMD} \
           { foreach s [glob *.sign] {a $s} }
```

3 Basic Usage of Oasys

3.1 Basic Commands

3.1.1 Managing Units

Every unit which is to be used in Oasys must be registered before the first use.

a UNIT Add *UNIT* to the set of registered units. *UNIT* must be found in the current search path (see Section 5.17 [oasys-path], page 15).

l
ll List the set of known units in short or in long format.

3.1.2 Evaluating Expressions

Expressions are always evaluated in a so-called *current context*. This context is one of the registered units. It is called the *focus*. (See also Section 5.1 [oasys-additional-context], page 12.)

Before evaluating expressions, any units whose source code have been changed are checked and compiled.

All expressions are stored. The history can be retrieved (see Section 5.34 [oasys-values], page 19). Element *I* can be referred to by the identifier `RES`*I*, e.g. `RES2` for expression number 2. The last expression can always be referred to with `RES`.

The evaluation of expressions of the built-in monadic type `com[data]` yields a constant, but does not execute the command.

f UNIT Set focus to *UNIT*.

e EXPR Evaluate expression in current focus.

ex EXPR Evaluate expression in current focus. Execute the resulting command.

x NAME Execute command.

t Switch printing of result type on or off.

3.1.3 Information

h [CMD] Print information on *CMD* or all commands.

i Print brief information on Oasys commands.

3.1.4 Compiler Functionality

Every unit can have one of three states:

loaded The source code has been loaded.

checked Abstract syntax of the unit is available.

compiled Object code of the unit is available. (Signature units can never have this state.)

The `r` and `c` commands without arguments refer to all structures whose source code has been changed.

r Reload structures.

c Reload structures, then check structures.

ld COM Reload structures, then create executable named *COM* which executes the command *COM*.

ar FOLDER
Reload structures, then archive structures in *FOLDER*

3.2 Printing Results

In order to print the result of an evaluation, Oasys must know how to transform a given data type into textual representation. The Opal library contains two data types for texts: `denotation` and `fmt`, the latter being a structured text which supports pretty printing.

A *print function* for type *data* is a function of type *data* `->` `denotation` or *data* `->` `fmt`. If the type is parameterized, the print function must have one additional higher-order parameter for the print functions of the parameter types. For example, a print function for type `map[dom, <, codom]` must have either functionality `(dom -> denotation) ** (codom -> denotation) -> map[dom, <, codom] -> denotation` or `(dom -> fmt) ** (codom -> fmt) -> map[dom, <, codom] -> fmt` (you may not mix `denotation` and `fmt`).

The user may define print functions explicitly (see Section 5.18 [oasys-print-method], page 16). If no print function is defined for type *type* in structure *struct*, Oasys tries to find one. If one of the structures *struct*, `Fmt`*struct*, *struct*`Conv` (in this order) contains possible candidates for print functions, one of these candidates is chosen. Selector functions are not considered.

If no print function can be found, Oasys prints `<some>` to indicate that 'some' result has been obtained, but cannot be represented.

3.2.1 Printing Functions

Functions, i.e. expressions with function functionality, are represented by their internal representation. For known functions their (internally unique) name is used, otherwise a lambda expression is printed.

Example:

```
Denotation.sign>e {+/+(",")}
⊣ +/+'Denotation,1(,)
```

Names are annotated with their origin. In this example, there is also an additional number to distinguish overloaded functions.

Local variables of lambda expressions are renamed, and infix applications are converted into prefix applictions.

Example:

```
Denotation.sign>e {IF false THEN \\a, b. a ++ b
                            ELSE \\a, b. a ++ "::" ++ b FI}
⊣ \\ a0, a1. ++'Denotation(a0, ++'Denotation("::", a1))
```

In some cases local variables are not treated properly.

3.3 Debugging

Commands starting with the letter 'd' are used for debugging. Note that the debugging features are experimental and not stable yet.

dm NAME
: Enable monitoring of execution of function *NAME*. A monitored function execution can be interrupted by hitting *C-C*. Technically, a monitored function is going to be interpreted.

db NAME [EXPR]
: Set a breakpoint for function *NAME*. Whenever this function is entered, execution is interrupted. The actual arguments of the function can be inspected by using the formal parameter names in expression evaluation. If *EXPR* is given, it must denote a boolean expression over the formal parameters of the function; execution is only interrupted if the guard expression is true.

do NAME
: Disable debugging for function *NAME* (cancelling a previous monitor or breakpoint command).

ds
: Show which functions are monitored or which have a breakpoint

dt
: Show backtrace.

dc
: Continue an interrupted execution

dn NUMBER
: Perform a number of steps after interrupted evaluation

df NUMBER
: Select frame

4 Alphabetic Listing of Oasys Shortcut Commands

4.1 a

a *STRUCTURE* synopsis

Add structure *STRUCTURE* and all imported structures to the set of registered structures. *STRUCTURE* may contain a directory path or a file suffix; both will be removed. *STRUCTURE* and its imported structures must be found in the Oasys search path (see Section 5.17 [oasys-path], page 15).

Example:

```
> a Rational.
⊣ loading Rational.sign
⊣ loading Rational.impl
```

See also Section 4.18 [l], page 10, Section 4.20 [ll], page 10, Section 5.22 [oasys-register], page 17.

4.2 ar

This section is still draft.

ar *FOLDER* synopsis

Reload, then create object archive for folder*FOLDER* to be used with the `ocs` command.

4.3 c

c synopsis

Reload all registered structures whose source has changed, then check these structures.

This command is rarely needed, because all actions are also part of the `e` command (see Section 4.13 [e], page 8).

Example:

```
> c
⊣ loading Rational.impl
⊣ checking Rational.impl
⊣ ERROR [Rational.impl at 43.5]:
⊣ improperly named function definition target or parameter foo
⊣ ERROR [check]: language error
```

See also Section 5.7 [oasys-check], page 13, Section 5.35 [oasys-verbosity], page 20.

4.4 db

This section is still draft.

db *NAME* [*EXPR*] synopsis
Set breakpoint for function *NAME* with optional guard *EXPR*.

4.5 dc

This section is still draft.

dc synopsis
Continue after interrupted evaluation.

4.6 df

This section is still draft.

df *NUMBER* synopsis
Select frame *NUMBER*.

4.7 di

This section is still draft.

di synopsis
Introduction to oasys debugging commands.

4.8 dm

This section is still draft.

dm *NAME* synopsis
Enable monitoring of execution of function *NAME*.

4.9 dn

This section is still draft.

dn [*STEPS*] synopsis
Perform *STEPS* steps after interrupted evaluation.

4.10 do

This section is still draft.

do *NAME* synopsis
Disable debugging for function *NAME*.

4.11 ds

This section is still draft.

ds synopsis

Show debugged functions.

4.12 dt

This section is still draft.

dt synopsis

Print back-trace.

4.13 e

e *EXPR* synopsis

Reload all registered structures whose source has changed, check these structures and compile them, then evaluate *EXPR* in current focus (see Section 4.15 [f], page 9) and print the result. If *EXPR* contains spaces or quotes, it should be surrounded by curly brackets. Errors in the expressions are reported with line number 0.

Note that monadic expressions (expressions of type `com[data]`) are evaluated but *not* executed.

Example:

```
Rational.impl>e 24 / 8
⊣ [3/1]
Rational.impl>e 34 / 8
⊣ ERROR [at 0.1]: no matching operation for 34
⊣ ERROR [check]: language error
⊣ aborted
```

See also Section 4.14 [ex], page 8, Section 4.15 [f], page 9, Section 4.23 [t], page 11, Section 4.24 [x], page 11, Section 5.11 [oasys-eval], page 14, Section 5.18 [oasys-print-method], page 16, Section 5.28 [oasys-show-type], page 18, Section 5.34 [oasys-values], page 19.

4.14 ex

ex *EXPR* synopsis

Reload all registered structures whose source has changed, check these structures and compile them, then evaluate *EXPR* in current focus (see Section 4.15 [f], page 9), *execute* the resulting command and print the result. The expression must be of type `com[data]`.

If *EXPR* contains spaces or quotes, it should be surrounded by curly brackets. Errors in the expressions are reported with line number 0.

Example:

```
BasicIO.sign>ex writeLine("abc")
⊣ abc
⊣ nil
```

Note that `abc` is the side effect of the execution of the command, and `nil` is the value returned by the command.

See also Section 4.13 [e], page 8, Section 4.15 [f], page 9, Section 4.23 [t], page 11, Section 4.24 [x], page 11, Section 5.12 [oasys-exec], page 14, Section 5.18 [oasys-print-method], page 16, Section 5.28 [oasys-show-type], page 18, Section 5.34 [oasys-values], page 19.

4.15 f

f *UNIT* synopsis

Set focus to *UNIT* (structure name + '`.sign`' or '`.impl`'). *UNIT* is the context in which the following expressions are evaluated. The current focus is included in the oasys prompt.

The focus can not be set to implementation parts of library structures (see Section 5.17 [oasys-path], page 15).

Example:

```
> f Nat.sign
```

See also Section 4.13 [e], page 8, Section 4.14 [ex], page 8, Section 4.24 [x], page 11, Section 5.13 [oasys-focus], page 14.

4.16 h

h [*CMD*] synopsis

Print short help on all commands (default) or the specified *CMD*.

Example:

```
> h a
⊣ Add STRUCTURE to known units.
⊣  A possible directory path and .sign or .impl suffix will be stripped
⊣  from STRUCTURE.
```

See also Section 5.14 [oasys-help], page 15.

4.17 i

i synopsis

Print a short summary (introduction) of the most important Oasys commands.

4.18 l

l synopsis

List all registered units.

Example:

```
> l
...
String.sign String.impl OptionConv.sign OptionConv.impl Com.sign
Com.impl ComConv.sign ComConv.impl
```

See also Section 4.20 [ll], page 10, Section 5.33 [oasys-units], page 19.

4.19 ld

ld *COM* synopsis

Reload all registered structures whose source has changed, check these structures and compile them, then create an executable file which will execute *COM*. *COM* must be a constant of type `com[void]`. The executable file will have the name *COM* as well.

Example:

```
HelloWorld.sign> ld hello
⊣ linking hello'HelloWorld.sign
```

4.20 ll

ll synopsis

List all registered units, together with their internal number and state.

Example:

```
HelloWorld.sign>ll
⊣ ...
⊣ HelloWorld.impl, (#43), checked
⊣ BasicIO.sign   , (#44), checked, library
⊣ BasicIO.impl   , (#45), checked, compiled, library
⊣ Pair.sign      , (#46), checked, library
⊣ Pair.impl      , (#47), checked, compiled, library
```

See also Section 4.18 [l], page 10, Section 5.33 [oasys-units], page 19.

4.21 q

q synopsis

Quit Oasys.

4.22 r

r synopsis

Reload all registered units whose source has changed.

This command is rarely needed, because it is part of the Section 4.3 [c], page 6 and the Section 4.13 [e], page 8 and Section 4.14 [ex], page 8 commands.

Example:

```
Rational.sign>r
⊣ loading Rational.sign
```

See also Section 5.24 [oasys-reload], page 17.

4.23 t

t synopsis

Control whether the result type of expressions is printed. The command toggles between on and off.

Example:

```
Rational.sign>t
⊣ show types is on
Rational.sign>e +
⊣ +'Rational
⊣ rat'Rational ** rat'Rational -> rat'Rational
```

See also Section 4.13 [e], page 8, Section 4.14 [ex], page 8, Section 5.28 [oasys-show-type], page 18.

4.24 x

x *COM* synopsis

Reload all known structures whose source has changed, then execute *COM* which must be a constant of type `com[void]`. This is faster than Section 4.14 [ex], page 8.

Example:

```
HelloWorld.sign>x hello
⊣ Hello, World!
```

See also Section 4.14 [ex], page 8, Section 5.25 [oasys-run], page 18.

5 Alphabetic Listing of Ordinary Oasys Commands

5.1 oasys-additional-context

oasys-additional-context (add *UNIT** | rm *UNIT** | clear |show) synopsis

Oasys manages a set of units which are always added to the current context when evaluating expressions.

`add` *UNITS*
: Add units to the additional context. Implementation units will be replaced by their corresponding signature parts.

`rm` *UNITS*
: Remove units from the additional context. Implementation units will be replaced by their corresponding signature parts.

`clear`
: Make the additional context the empty set.

`show`
: Print the additional context.

The smaller the set of imported units, the faster expressions will be evaluated. Additional imported units may cause ambiguities and even cyclic dependency errors.

See also Section 5.13 [oasys-focus], page 14, Section 5.11 [oasys-eval], page 14, Section 5.12 [oasys-exec], page 14.

5.2 oasys-all-commands

oasys-all-commands synopsis

Commands from this section are not available for completion on the command line and are hidden from the help command by default. This command makes them available.

See also Section 5.21 [oasys-really-all-commands], page 17

5.3 oasys-archive

This section is still draft.

oasys-archive [*FOLDER-NAME*] synopsis

Create object archive for use with the ocs-command for the given *FOLDER-NAME*. *FOLDER-NAME* is the base name of the source path. If *FOLDER-NAME* is omitted, archives are created for all folders.

See also Section 4.2 [ar], page 6.

5.4 oasys-args

oasys-args { *ARG* } synopsis

Set arguments for next invocation of the evaluator process. This is for debugging purposes only.

See also Section 5.5 [oasys-args-show], page 13.

5.5 oasys-args-show

oasys-args-show synopsis

Print arguments set with last call of Section 5.4 [oasys-args], page 13.

5.6 oasys-bt

This section is still draft.

oasys-bt synopsis

Print back trace.

5.7 oasys-check

oasys-check { *UNITS* } synopsis

Check the given *UNITS* (if no units are given, all registered units are checked). It is not checked, whether the source code of *UNITS* has changed.

Imported units are checked first.

See also Section 4.3 [c], page 6, Section 5.24 [oasys-reload], page 17.

5.8 oasys-compile

oasys-compile { *UNITS* } synopsis

Compile the given *UNITS* (if no units are given, all registered units are checked). Imported units are checked and compiled if necessary.

5.9 oasys-continue

This section is still draft.

oasys-continue synopsis

Continue the last broken evaluation.

5.10 oasys-debug

This section is still draft.

oasys-debug [*monitor NAME* | *off NAME* | *break NAME* `{` *EXPR* `}` | *update*] synopsis

Maintain debugging state.

5.11 oasys-eval

oasys-eval *EXPR* synopsis

Evaluate *EXPR* in current focus (see Section 5.13 [oasys-focus], page 14 and Section 5.1 [oasys-additional-context], page 12) and print the result. If *EXPR* contains spaces, quotes or square brackets, it should be surrounded by curly brackets. Errors in the expressions are reported with line number 0.

Note that monadic expressions (expressions of type `com[data]`) are evaluated but *not* executed.

See also Section 4.13 [e], page 8, Section 5.12 [oasys-exec], page 14, Section 5.13 [oasys-focus], page 14, Section 5.25 [oasys-run], page 18, Section 5.18 [oasys-print-method], page 16, Section 5.28 [oasys-show-type], page 18, Section 5.1 [oasys-additional-context], page 12.

5.12 oasys-exec

oasys-exec *EXPR* synopsis

Evaluate *EXPR* in current focus (see Section 5.13 [oasys-focus], page 14 and Section 5.1 [oasys-additional-context], page 12), *execute* the resulting command and print the result. The expression must be of type `com[data]`.

If *EXPR* contains spaces, quotes or square brackets, it should be surrounded by curly brackets. Errors in the expressions are reported with line number 0.

See also Section 4.14 [ex], page 8, Section 5.11 [oasys-eval], page 14, Section 5.13 [oasys-focus], page 14, Section 5.25 [oasys-run], page 18, Section 5.18 [oasys-print-method], page 16, Section 5.28 [oasys-show-type], page 18.

5.13 oasys-focus

oasys-focus [*UNIT*] synopsis

Set focus to *UNIT* (structure name `+` '`.sign`' or '`.impl`'). *UNIT* is the context in which the following expressions are evaluated. (See Section 5.1 [oasys-additional-context], page 12 for extending this context.) The current focus is included in the oasys prompt.

The focus can not be set to implementation parts of library structures (see Section 5.17 [oasys-path], page 15).

If no *UNIT* is given, the command returns the current focus.

See also Section 5.1 [oasys-additional-context], page 12.

5.14 oasys-help

oasys-help [*CMD*] synopsis

Print short help on all commands (default) or the specified *CMD*.

See also Section 4.16 [h], page 9

5.15 oasys-link

This section is still draft.

oasys-link *OBJECTNAME*> [*EXEC-NAME*] synopsis

Link top-level command to executable *EXEC-NAME*. If *EXEC-NAME* is omitted, use name of top-level command. *OBJECTNAME* must be a constant of type `com[void]`.

See also Section 4.19 [ld], page 10.

5.16 oasys-new

This section is still draft.

oasys-new *STRUCTNAME* synopsis

Create a new structure with name *STRUCTNAME*.

5.17 oasys-path

oasys-path [add (ocs | library) *PATH*] synopsis

- `oasys-path` just returns the current search path as a sequence of pairs. The first element of each pair is the path to the sources, the second element is the path to compiler results.
- `oasys-path add ocs` *PATH* adds *PATH* to the search path. If *PATH* contains a file '`config.oasys`', it will be read and executed.
- `oasys-path add library` *PATH* adds *PATH* to the search path. Structures in this path are considered to be *library structures*. If *PATH* contains a file '`config.oasys`', it will be read and executed.

Difference between library and ordinary structures is that library structures are never checked for reloading (library structures are never outdated), and that compiler results are located in a different place.

5.18 oasys-print-method

oasys-print-method *SORT FUN* synopsis

Define *FUN* to be the print function for *SORT*. Any evaluated Opal expression of type *SORT* is passed to this function to obtain a printable representation. *FUN* must have the type *SORT* `->` `fmt` or *SORT* `->` `denotation`.

In case *SORT* is parameterized, *FUN* must have a higher-order parameter of type *data* `->` `fmt` or *data* `->` `denotation` for every parameter sort *data*.

Take care in chosing a total function as print function.

See also Section 3.2 [Printing Results], page 4.

5.19 oasys-proc

oasys-proc *ARGSPEC HELP ARGS BODY* synopsis

Define a new oasys command. As all of the above arguments consist of several words, they should be enclosed in curly brackets.

ARGSPEC specifies the syntax (including the name of the new command). Elements of *ARGSPEC* are

STRING	literal
`@`*VAR*	variable
`@#`*UNIT*	variable with unit completion
`@^`*FILE*	variable with filename completion.

Each of the elements may be prefixed with `?` to indicate optional elements or `*` to indicate repeatitions.

HELP contains the help text printed by the commands Section 4.16 [h], page 9 and Section 5.14 [oasys-help], page 15.

ARGS is a list of Tcl variables for each of the elements of *ARGSPEC* (including literals).

BODY is a sequence of Tcl commands which may refer to the variables in *ARGS*. Optional elements and repetitions are represented as Tcl lists.

See Section 2.4 [Defining New Oasys Commands], page 2 for an example.

5.20 oasys-quit

oasys-quit synopsis

Quit Oasys

See also Section 4.21 [q], page 10.

5.21 oasys-really-all-commands

oasys-really-all-commands synopsis

There is a set of commands with prefix `oasys-intern` which are hidden unless this command is given.

See also Section 5.2 [oasys-all-commands], page 12.

5.22 oasys-register

oasys-register `{` *STRUCTURE* `}` synopsis

Register the given structures and load them into the repository unless they are already loaded.

STRUCTURE may contain a directory path or a file suffix; both will be removed. *STRUCTURE* and its imported structures must be found in the Oasys search path (see Section 5.17 [oasys-path], page 15).

See also Section 4.1 [a], page 6, Section 5.17 [oasys-path], page 15.

5.23 oasys-related

oasys-related [*MODE*] [*RELATION*] `{` *UNIT* `}` synopsis

Print all units which related by *RELATION* with a unit from *UNIT*. *MODE* is either `direct` (default) or `transitive` and modifies the *RELATION*.

RELATION is one of

`imports`	gives the imported basic units. If *UNIT* is an implementation, its signature part is regarded as an import.
`importers`	is the reversed relation of `imports`, and gives the basic units which import *UNIT*. If *UNIT* is a signature, its implementation part is regarded as an importer.
`implImports`	is the set of implementation units which are used to realize *UNIT*. If *UNIT* is a signature, then it will be treated identical to the implementation of this signature.
`implImporters`	is the reversed relation of `implImports`, and delivers all units which use *UNIT* (or its implementation part, if *UNIT* is a signature) for their implementation.

The default *RELATION* is `imports`.

5.24 oasys-reload

oasys-reload `{` *UNIT* `}` synopsis

Reload *UNIT* if its source has changed.

See also Section 4.22 [r], page 11.

5.25 oasys-run

oasys-run synopsis

Execute *COM* which must be a constant of type `com[void]`. This is faster than Section 5.12 [oasys-exec], page 14.

See also Section 4.24 [x], page 11, Section 5.12 [oasys-exec], page 14.

5.26 oasys-save

oasys-save [*PATHNAME*] synopsis

Save current state of the repository to *PATHNAME* (default is '`default.repo`').

If *PATHNAME* is given to Oasys as argument when it is called, it will read the saved repository.

5.27 oasys-select

This section is still draft.

oasys-select *NUMBER* synopsis

Select an item in evaluation stack (see Section 5.6 [oasys-bt], page 13).

5.28 oasys-show-type

oasys-show-type [on | off | toggle | show] synopsis

Control whether the result type of expressions is printed.

on	result type is printed
off	result type is not printed
toggle	current state is toggled
show	current state is printed

See also Section 4.23 [t], page 11, Section 5.11 [oasys-eval], page 14, Section 5.12 [oasys-exec], page 14.

5.29 oasys-source

This section is still draft.

oasys-source [*UNIT*] synopsis

This command is not implemented.

5.30 oasys-step

This section is still draft.

oasys-step [*STEPS*] synopsis
Evaluate a *STEPS* steps in current evaluation.

5.31 oasys-unit-compiled

This section is still draft.

oasys-unit-compiled `{` *UNIT* `}` synopsis
Declare units to be compiled.

5.32 oasys-unit-interpreted

This section is still draft.

oasys-unit-interpreted `{` *UNIT* `}` synopsis
Declare units to be interpreted.

5.33 oasys-units

oasys-units [full | name | file] synopsis
Print information about the registered units. For every unit are printed:

- full: unit name, internal number, state.
- name: unit name
- file: path name

See also Section 4.18 [l], page 10, Section 4.20 [ll], page 10.

5.34 oasys-values

oasys-values synopsis
Show a list of expressions which have been evaluated.

The printed representation is the user input. The stored value is the result of the evaluation in the environment which was current at the time the expression was evaluated for the first time.

The expressions can be accessed by the identifiers `RES0`, `RES1`, ... The last evaluated expression can also be accessed as `RES`.

See also Section 3.1.2 [Evaluating Expressions], page 3.

5.35 oasys-verbosity

oasys-verbosity (diag (hint | warn | error) | message *LEVEL*) synopsis

Control the verbosity of Oasys.

diag hint	Show all diagnostics from checking Opal code.
diag warn	Show warnings and errors from checking Opal code.
diag error	Show only errors from checking Opal code.
message *LEVEL*	Show messages of level *LEVEL* and below. (Currently levels 0 and 3 are used.)

Index

(Index is nonexistent)

Table of Contents