Open Issues

Takahiro Ueda

Juntendo U.

23 June 2026

FORM Developer's Workshop 2026

Background and aim

There are many unresolved topics around FORM

Many are recorded in GitHub Issues, Discussions, or Pull Requests

Current tracker counts:

- open Issues: 238
- open Pull Requests: 13
- open Discussions: 20

This talk gives a shared map for discussion and hands-on sessions during the workshop

Background and aim

There are many unresolved topics around FORM

Many are recorded in GitHub Issues, Discussions, or Pull Requests

Current tracker counts:

- open Issues: 238
- open Pull Requests: 13
- open Discussions: 20

This talk gives a shared map for discussion and hands-on sessions during the workshop

Background and aim

There are many unresolved topics around FORM

Many are recorded in GitHub Issues, Discussions, or Pull Requests

Current tracker counts:

- open Issues: 238
- open Pull Requests: 13
- open Discussions: 20

This talk gives a shared map for discussion and hands-on sessions during the workshop

Closing Issues

#844 Closing issues

Many issues filed on GitHub

Important reports can get buried in a long list of issues

Should we use a stale/closing workflow?

Idea: long inactive issue, no important label → mark as stale
→ still no response → close with reason

Good : make important issues easier to find ☺

Bad : initial stale notification burst ☹

Closing Issues

#844 Closing issues

Many issues filed on GitHub

Important reports can get buried in a long list of issues

Should we use a stale/closing workflow?

Idea: long inactive issue, no important label → mark as stale
→ still no response → close with reason

Good : make important issues easier to find ☺

Bad : initial stale notification burst ☹

Closing Issues

#844 Closing issues

Many issues filed on GitHub

Important reports can get buried in a long list of issues

Should we use a stale/closing workflow?

Idea: long inactive issue, no important label → mark as stale
→ still no response → close with reason

Good : make important issues easier to find ☺

Bad : initial stale notification burst ☹

Closing Issues

#844 Closing issues

Many issues filed on GitHub

Important reports can get buried in a long list of issues

Should we use a stale/closing workflow?

Idea: long inactive issue, no important label → mark as stale
→ still no response → close with reason

Good : make important issues easier to find ☺

Bad : initial stale notification burst ☹

Outline

Reports and examples	selected recent issues for discussion
Long-standing issues	hard-to-fix behaviour
Code reuse	namespaces and packages
Maintenance choices	support policy and developer tools
Discussion	next steps for the workshop and beyond

Reports and examples

Selected recent issues for discussion

Recent bug reports

Examples from recent reports:

- #839 Triple dot operator removes 0s
- #836 Terminate with `float_` function after `#EndFloat`
- #833 Broken stats print due to #805

We will see more easy(?) examples in hands-on sessions

#839: Triple dot operator removes 0s

```
On names;
Symbols <x01a1>,...,<x01a3>;
.end
```

```
Symbols
  x1a1 x1a2 x1a3
```

Likely place to inspect: `ExpandTripleDots` in `pre.c`

#839: Triple dot operator removes 0s

```
1 On names;
2 Symbols <x01a1>,...,<x01a3>;
3 .end
```

```
Symbols
  x1a1 x1a2 x1a3
```

Likely place to inspect: `ExpandTripleDots` in `pre.c`

#839: Triple dot operator removes 0s

```
1 On names;
2 Symbols <x01a1>,...,<x01a3>;
3 .end
```

```
 Symbols
   x1a1 x1a2 x1a3
```

Likely place to inspect: `ExpandTripleDots` in `pre.c`

#836: Terminate with float_ function after #EndFloat

```
#StartFloat 10d
Local F = 3.0;
.sort
#EndFloat
multiply 5;
Print;
.end
```

```
Illegal attempt at using a float_ function without proper startup.
Please use #StartFloat <options> first.
Program terminating at float-after-endfloat.frm Line 6 -->
Terminate called from float.c:381 (UnpackFloat)
```

Interestingly, likely AI-generated PR #837 was opened (not a correct fix)

#836: Terminate with float_ function after #EndFloat

```
#StartFloat 10d
Local F = 3.0;
.sort
#EndFloat
multiply 5;
Print;
.end
```

```
Illegal attempt at using a float_ function without proper startup.
Please use #StartFloat <options> first.
Program terminating at float-after-endfloat.frm Line 6 -->
Terminate called from float.c:381 (UnpackFloat)
```

Interestingly, likely AI-generated PR #837 was opened (not a correct fix)

#833: Broken stats print due to #805

The visible change is in a statistics line

FORM 5.0.0:

```
Time =       0.00 sec    Generated terms =        100
bcdefghijklmnopq         Terms in output =        100
                         Bytes used      =       2412
```

master after merging #805:

```
Time =       0.00 sec    Generated terms =        100
abcdefghijklmnopq         Terms in output =        100
                         Bytes used      =       2412
```

Josh's question:

- keep exactly the old truncation?
- or use more of the empty space before `Terms in ...`?

Long-standing issues

Hard-to-fix behaviour

Well-known, documented limitation on MaxTermSize

`MaxTermSize` sets the maximum size of a term

Problem: this value cannot be changed in the middle of a run

If your expression reaches it, you need to restart the run with a larger `MaxTermSize`

Question: is it possible to remove this limit?

Well-known, documented limitation on MaxTermSize

`MaxTermSize` sets the maximum size of a term

Problem: this value cannot be changed in the middle of a run

If your expression reaches it, you need to restart the run
with a larger `MaxTermSize`

Question: is it possible to remove this limit?

Well-known, documented limitation on MaxTermSize

`MaxTermSize` sets the maximum size of a term

Problem: this value cannot be changed in the middle of a run

If your expression reaches it, you need to restart the run
with a larger `MaxTermSize`

Question: is it possible to remove this limit?

Well-known, documented limitation on MaxTermSize

`MaxTermSize` sets the maximum size of a term

Problem: this value cannot be changed in the middle of a run

If your expression reaches it, you need to restart the run
with a larger `MaxTermSize`

Question: is it possible to remove this limit?

Short answer: no

Well-known(?) undocumented issues on pattern matching

```
1  Symbols x,y,a,b;
2  Local F = a^2*b;
3
4  id y?^2*x? = 0;
5  Print;
6  .sort
7
8  id x?^2*y? = 0;
9  Print;
10 .end
```

```
F =
   a^2*b;
```

```
F = 0;
```

Backtracking of pattern matching is not sufficient in some cases

Well-known(?) undocumented issues on pattern matching

```
1  Symbols x,y,a,b;
2  Local F = a^2*b;
3
4  id y?^2*x? = 0;
5  Print;
6  .sort
7
8  id x?^2*y? = 0;
9  Print;
10 .end
```

```
F =
   a^2*b;
```

```
F = 0;
```

Backtracking of pattern matching is not sufficient in some cases

This is a family of frequently asked questions

Related reports:

- `#12` Wrong circular pattern match
- `#98` Pattern `?a,p?,?b` not matching in nested functions
- `#300` Pattern matching for products of vectors [good first issue]
- `#376` strange behavior of result of order of vector definitions
- `#832` vector product substitutions with powers

Complete rewrite at some point?

Document the current matching rules?

This is a family of frequently asked questions

Related reports:

- `#12` Wrong circular pattern match
- `#98` Pattern `?a,p?,?b` not matching in nested functions
- `#300` Pattern matching for products of vectors [good first issue]
- `#376` strange behavior of result of order of vector definitions
- `#832` vector product substitutions with powers

Complete rewrite at some point?

Document the current matching rules?

This is a family of frequently asked questions

Related reports:

- `#12` Wrong circular pattern match
- `#98` Pattern `?a,p?,?b` not matching in nested functions
- `#300` Pattern matching for products of vectors [good first issue]
- `#376` strange behavior of result of order of vector definitions
- `#832` vector product substitutions with powers

Complete rewrite at some point?

Document the current matching rules?

Code reuse

Namespaces and packages

Library authors need private names

#236 FORM namespace?

Example:

```
1  Symbols x, tmp;
2
3  #procedure Lib()
4    id tmp = 0;
5  #endprocedure
6
7  Local F = x + tmp;
8
9  #call Lib()
10
11 Print +s;
12 .end
```

Problem:

- The library `tmp` and the user `tmp` are the same name
- Reusable libraries need a way to hide internal names

Library authors need private names

#236 FORM namespace?

Example:

```
1  Symbols x, tmp;
2
3  #procedure Lib()
4    id tmp = 0;
5  #endprocedure
6
7  Local F = x + tmp;
8
9  #call Lib()
10
11 Print +s;
12 .end
```

Problem:

- The library `tmp` and the user `tmp` are the same name
- Reusable libraries need a way to hide internal names

FORM package manager

#322 FORM package manager Solved by Vitaly Magerya's Talk?

Most of programming languages have package managers that help people install and manage libraries/dependencies

Examples:
Fortran (`fpm`), C++ (`Conan`, `vcpkg`), Java (`Maven`), Rust (`cargo`), Python (`pip`), Mahtematica (`ResorceFunction`), ...

Why not for FORM?

Namespace may be a prerequisite for a package manager?

FORM package manager

#322 FORM package manager Solved by Vitaly Magerya's Talk?

Most of programming languages have package managers that help people install and manage libraries/dependencies

Examples:
Fortran (`fpm`), C++ (`Conan`, `vcpkg`), Java (`Maven`), Rust (`cargo`), Python (`pip`), Mahtematica (`ResorceFunction`), ...

Why not for FORM?

Namespace may be a prerequisite for a package manager?

FORM package manager

#322 FORM package manager Solved by Vitaly Magerya's Talk?

Most of programming languages have package managers that help people install and manage libraries/dependencies

Examples:
Fortran (`fpm`), C++ (`Conan`, `vcpkg`), Java (`Maven`), Rust (`cargo`), Python (`pip`), Mahtematica (`ResorceFunction`), ...

Why not for FORM?

Namespace may be a prerequisite for a package manager?

FORM package manager

#322 FORM package manager Solved by Vitaly Magerya's Talk?

Most of programming languages have package managers that help people install and manage libraries/dependencies

Examples:
Fortran (`fpm`), C++ (`Conan`, `vcpkg`), Java (`Maven`), Rust (`cargo`), Python (`pip`), Mahtematica (`ResorceFunction`), ...

Why not for FORM?

Namespace may be a prerequisite for a package manager?

Maintenance choices

Support policy and developer tools

Legacy feature support

#627 Considering deprecation of several features in FORM

Legacy features that have been supported for a long time can make the codebase harder to maintain

- #623 native Windows version
- #624 32-bit system version
- #625 ParFORM

Questions:

- Who still uses each feature? The issues have received no responses
- When would it be reasonable to drop support?

Legacy feature support

#627 Considering deprecation of several features in FORM

Legacy features that have been supported for a long time can make the codebase harder to maintain

- #623 native Windows version
- #624 32-bit system version
- #625 ParFORM

Questions:

- Who still uses each feature? The issues have received no responses
- When would it be reasonable to drop support?

Legacy feature support

#627 Considering deprecation of several features in FORM

Legacy features that have been supported for a long time can make the codebase harder to maintain

- #623 native Windows version
- #624 32-bit system version
- #625 ParFORM

Questions:

- Who still uses each feature? The issues have received no responses
- When would it be reasonable to drop support?

Do we still need sequential FORM?

A more extreme idea:
we could drop sequential FORM and keep only TFORM.

In principle, `form` could be equivalent to `tform -w0`

We could make `tform` available as `form`, so that `form -w16` works as expected

If we also drop PARFORM, we could unify FORM around a single execution model

This would greatly reduce the complexity of the codebase and testing

Whether we like it or not, sequential FORM already links against `Pthreads` via `FLINT`

Do we still need sequential FORM?

A more extreme idea:
we could drop sequential FORM and keep only TFORM.

In principle, `form` could be equivalent to `tform -w0`

We could make `tform` available as `form`, so that `form -w16`
works as expected

If we also drop PARFORM, we could unify FORM around
a single execution model

This would greatly reduce the complexity of the codebase and testing

Whether we like it or not, sequential FORM already links
against `Pthreads` via `FLINT`

Do we still need sequential FORM?

A more extreme idea:
we could drop sequential FORM and keep only TFORM.

In principle, `form` could be equivalent to `tform -w0`

We could make `tform` available as `form`, so that `form -w16`
works as expected

If we also drop PARFORM, we could unify FORM around
a single execution model

This would greatly reduce the complexity of the codebase and testing

Whether we like it or not, sequential FORM already links
against `Pthreads` via `FLINT`

Do we still need sequential FORM?

A more extreme idea:
we could drop sequential FORM and keep only TFORM.

In principle, `form` could be equivalent to `tform -w0`

We could make `tform` available as `form`, so that `form -w16`
works as expected

If we also drop PARFORM, we could unify FORM around
a single execution model

This would greatly reduce the complexity of the codebase and testing

Whether we like it or not, sequential FORM already links
against `Pthreads` via `FLINT`

Checkpoints mechanism

#626 Considering deprecating the checkpoint mechanism

Though it is also declared as deprecating, checkpoints are useful if they really work:

- long runs
- recovery after interruptions
- expensive computations

Question:

- repair, redesign, or retire?

Checkpoints mechanism

#626 Considering deprecating the checkpoint mechanism

Though it is also declared as deprecating, checkpoints are useful if they really work:

- long runs
- recovery after interruptions
- expensive computations

Question:

- repair, redesign, or retire?

Source code formatting

#712 Code formatter?

FORM's source-code conventions are strongly influenced by Jos's style

They have historical and cultural value within the FORM community,
but they are not familiar to many potential contributors

Because this style is not supported by widely used formatters,
formatting still relies on manual work and human review

Questions:

- Can we make FORM source code compatible
  with an existing formatter such as `clang-format`?
- Can AI learn Jos's conventions well enough to help us
  format FORM source code consistently?

Source code formatting

`#712` Code formatter?

FORM's source-code conventions are strongly influenced by Jos's style

They have historical and cultural value within the FORM community,
but they are not familiar to many potential contributors

Because this style is not supported by widely used formatters,
formatting still relies on manual work and human review

Questions:

- Can we make FORM source code compatible
  with an existing formatter such as `clang-format`?
- Can AI learn Jos's conventions well enough to help us
  format FORM source code consistently?

Source code formatting

`#712` Code formatter?

FORM's source-code conventions are strongly influenced by Jos's style

They have historical and cultural value within the FORM community,
but they are not familiar to many potential contributors

Because this style is not supported by widely used formatters,
formatting still relies on manual work and human review

Questions:

- Can we make FORM source code compatible
  with an existing formatter such as `clang-format`?
- Can AI learn Jos's conventions well enough to help us
  format FORM source code consistently?

Source code formatting

`#712` Code formatter?

FORM's source-code conventions are strongly influenced by Jos's style

They have historical and cultural value within the FORM community,
but they are not familiar to many potential contributors

Because this style is not supported by widely used formatters,
formatting still relies on manual work and human review

Questions:

- Can we make FORM source code compatible
  with an existing formatter such as `clang-format`?
- Can AI learn Jos's conventions well enough to help us
  format FORM source code consistently?

Discussion

Next steps for the workshop and beyond

Discussion questions: hide state

#828 PushHide and PopHide together with regular Hide statement

```
Symbols x;
Local F1 = x+1;
.sort
PushHide;
Local F2 = x+2;
.sort
PushHide;
Local F3 = x+3;
.sort
Hide;
.sort
PopHide;   * Unhide F1, F2 and F3
Print;
.end
```

Mixed operations:

- stack operations: `PushHide`, `PopHide`
- ordinary operation: `Hide`

Question:

- what should remain active after `PopHide`?
- how should stack-based hiding interact with regular `Hide`?

Discussion questions: hide state

#828 PushHide and PopHide together with regular Hide statement

```
Symbols x;
Local F1 = x+1;
.sort
PushHide;
Local F2 = x+2;
.sort
PushHide;
Local F3 = x+3;
.sort
Hide;
.sort
PopHide;  * Unhide F1, F2 and F3
Print;
.end
```

Mixed operations:

- stack operations: `PushHide`, `PopHide`
- ordinary operation: `Hide`

Question:

- what should remain active after `PopHide`?
- how should stack-based hiding interact with regular `Hide`?

Discussion questions: dropargs boundary rules

#675 transform dropargs – whole argument list is dropped when having fewer arguments

```
Local F = f(1);
transform f,dropargs(1,2);
```

```
Local F = f(1);
transform f,dropargs(2,last);
```

Question:

- no-op, warning, error, or current behaviour?
- which rule is least surprising for existing programs?

Discussion questions: dropargs boundary rules

#675 transform dropargs – whole argument list is dropped when having fewer arguments

```
Local F = f(1);
transform f,dropargs(1,2);
```

```
Local F = f(1);
transform f,dropargs(2,last);
```

Question:

- no-op, warning, error, or current behaviour?
- which rule is least surprising for existing programs?

Discussion questions: summary

#391 Removing output line length limit

Vitaly's Question

- Can we remove or relax the output line-length limit (255)?

Today, we saw many questions requiring decision, e.g.:

#627 Considering deprecation of several features in FORM

#712 Code formatter?

Not everything must be solved right now

But hopefully big progress in this week

Discussion questions: summary

#391 Removing output line length limit

Vitaly's Question

- Can we remove or relax the output line-length limit (255)?

Today, we saw many questions requiring decision, e.g.:

#627 Considering deprecation of several features in FORM

#712 Code formatter?

Not everything must be solved right now

But hopefully big progress in this week