# The Source Stepper in Allegro CL 8.2

David Margolies (dm@franz.com)
Duane Rettig

Questions and comments to
support@franz.com

## Source Stepper Availability

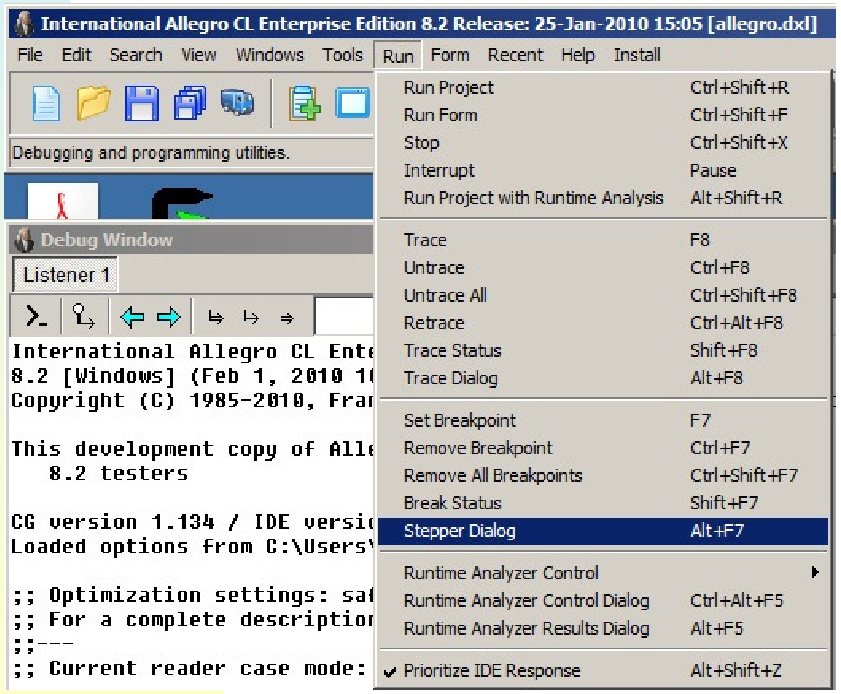
- Available on platforms that support the IDE (Integrated Development Environment)
- Available as a tty interface when IDE is not available (as we will show)
- Not available on Solaris or AIX at this time

## Requirements for source stepping

- Code must be in a file
- The file must be compiled and loaded into Lisp
- The compilation must be done while the compiler:save-source-level-debug-info-switch is true, which it is when the debug optimization quality is 3
- *load-source-file-info* should be true (as it is initially)

## Source Stepping in the IDE

When the Stepper Dialog is visible, compilation is done right (display it with the **Stepper Dialog** command on the IDE's **Run** menu):


International Allegro CL Enterprise Edition 8.2 Release: 25-Jan-2010 15:05 [allegro.dxl]
File Edit Search View Windows Tools Run Form Recent Help Install
Debugging and programming utilities.
Debug Window
Listener 1
Run Project Ctrl+Shift+R
Run Form Ctrl+Shift+F
Stop Ctrl+Shift+X
Interrupt Pause
Run Project with Runtime Analysis Alt+Shift+R
Trace F8
Untrace Ctrl+F8
Untrace All Ctrl+Shift+F8
Retrace Ctrl+Alt+F8
Trace Status Shift+F8
Trace Dialog Alt+F8
Set Breakpoint F7
Remove Breakpoint Ctrl+F7
Remove All Breakpoints Ctrl+Shift+F7
Break Status Shift+F7
Stepper Dialog Alt+F7
Runtime Analyzer Control
Runtime Analyzer Control Dialog Ctrl+Alt+F5
Runtime Analyzer Results Dialog Alt+F5
✓ Prioritize IDE Response Alt+Shift+Z

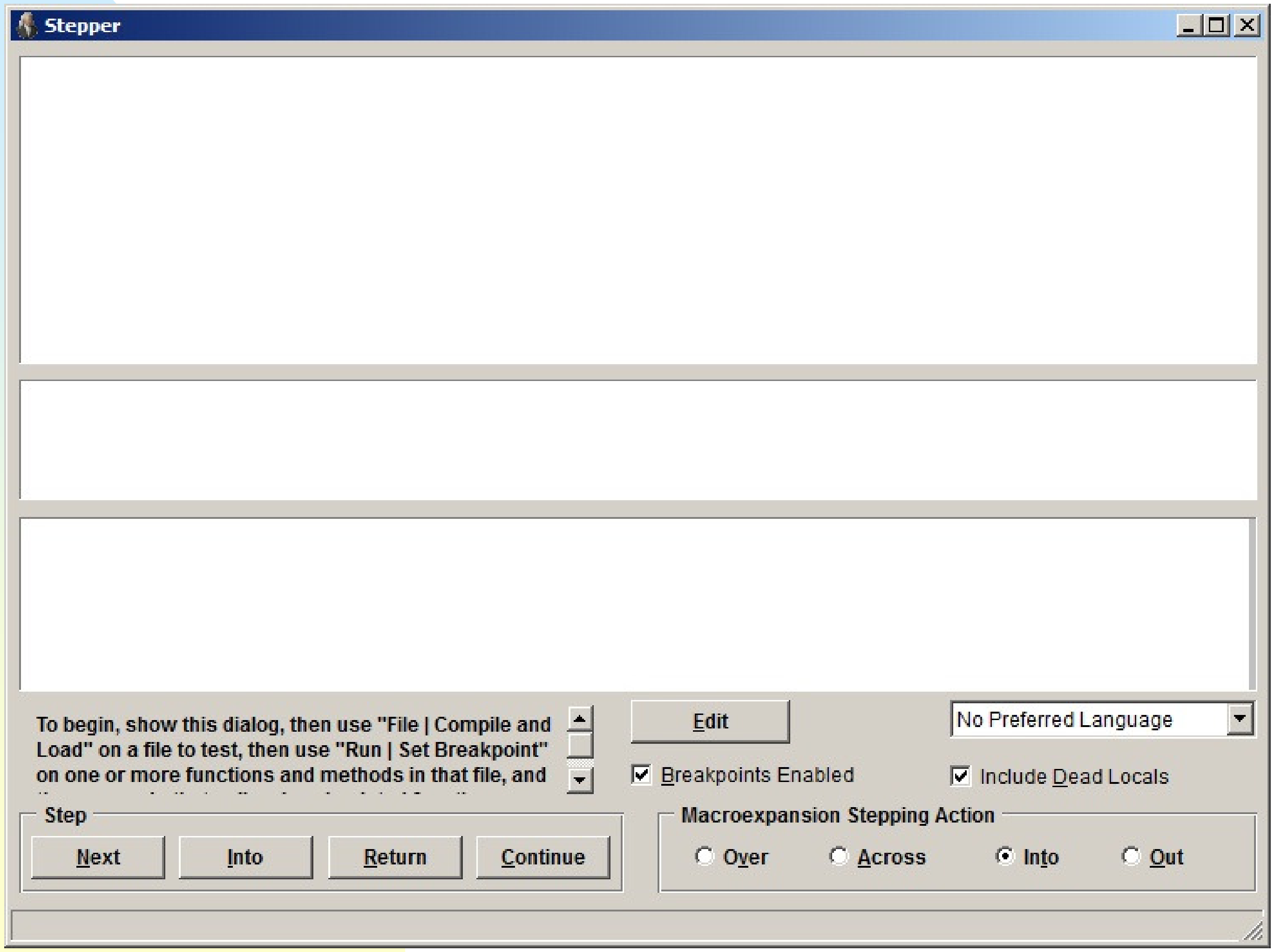
```
International Allegro CL Ente
8.2 [Windows] (Feb 1, 2010 1(
Copyright (C) 1985-2010, Fran

This development copy of Alle
   8.2 testers

CG version 1.134 / IDE versio
Loaded options from C:\Users\

;; Optimization settings: saf
;; For a complete descriptio
;;---
;; Current reader case mode:
```

# Stepper Dialog



To begin, show this dialog, then use "File | Compile and Load" on a file to test, then use "Run | Set Breakpoint" on one or more functions and methods in that file, and

No Preferred Language

[✓] Breakpoints Enabled

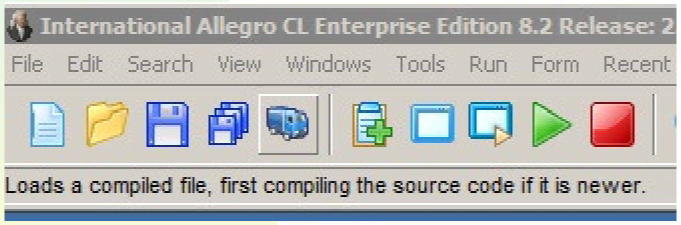
[✓] Include Dead Locals

Macroexpansion Stepping Action: Over Across [✓ Into] Out

2/25/2010

## Once Stepper Dialog is displayed, files will be compiled and loaded with info stored

- If you use IDE tools to compile and load file, for example using the Compile/Load button, the file will be compiled suitably for source stepping.


International Allegro CL Enterprise Edition 8.2 Release: 2
File Edit Search View Windows Tools Run Form Recent

Loads a compiled file, first compiling the source code if it is newer.

## A first example

- The following function is defined in foo.cl:

```
(defun foo (path n)
  (with-open-file (s path :direction :input
                          :if-does-not-exist nil)
    (let (line)
      (dotimes (i n)
        (setq line (read-line s nil s))
        (if (eq line s) (return))
        (print s)))))
```

## You must set at least one breakpoint using the :br top-level command

- :br foo

This sets a breakpoint at foo. When a call is made to the function foo, computation will stop and information will be displayed in the stepper dialog (we are not doing this yet)

```
:br nil  ;;  clears breakpoints
```

## The function foo reads some lines of a file and prints them

- The idea is you specify a file and a number of lines, that number of lines read and printed.
- There is an error in the function: the stream object is printed rather than the line.

## We compile and load the file and call foo:

```
cg-user(6): (foo "foo.cl" 10)

#<file-simple-stream #P"foo.cl" for input pos 23 @ #x210f0a6a>
#<file-simple-stream #P"foo.cl" for input pos 25 @ #x210f0a6a>
#<file-simple-stream #P"foo.cl" for input pos 46 @ #x210f0a6a>
#<file-simple-stream #P"foo.cl" for input pos 94 @ #x210f0a6a>
#<file-simple-stream #P"foo.cl" for input pos 149 @ #x210f0a6a>
#<file-simple-stream #P"foo.cl" for input pos 169 @ #x210f0a6a>
#<file-simple-stream #P"foo.cl" for input pos 196 @ #x210f0a6a>
#<file-simple-stream #P"foo.cl" for input pos 244 @ #x210f0a6a>
#<file-simple-stream #P"foo.cl" for input pos 284 @ #x210f0a6a>
#<file-simple-stream #P"foo.cl" for input pos 310 @ #x210f0a6a>
nil
cg-user(7):
```

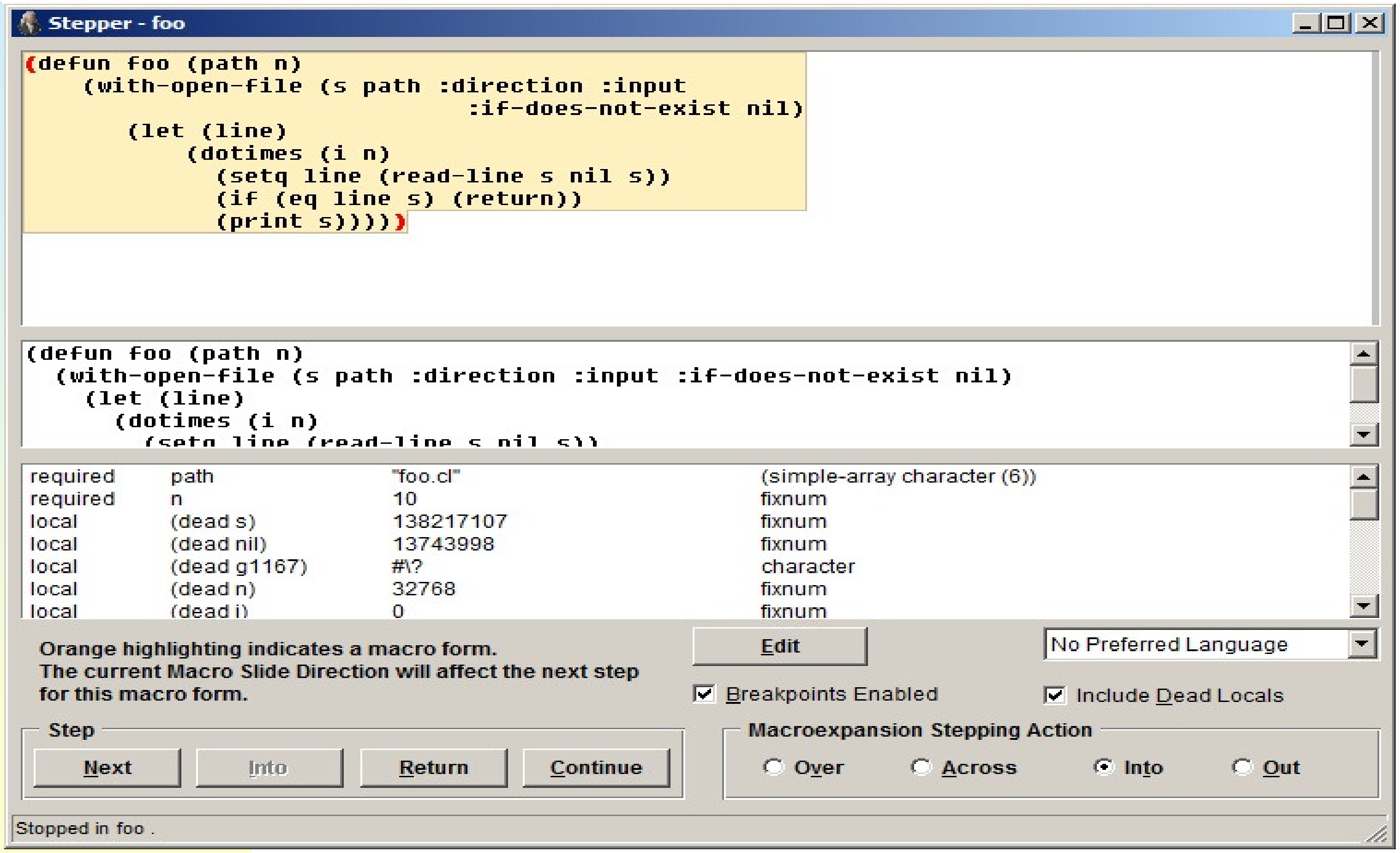
## Not what we wanted!

- So we will step through to see what is going on.
- We display the Stepper Dialog. We must recompile (so source debug info will be displayed).
- We modify foo.cl and save so compile/load will recompile (you can enter a space to change the file).
- We set a breakpoint, :br foo, and call (foo “foo.cl” 10)

## Stepping information makes the compiled (fasl) file bigger

- foo.fasl without stepping info is 3 Kb.
- foo.fasl with stepping info is 8 Kb.

## Stepper Dialog displaying call to foo



```
(defun foo (path n)
    (with-open-file (s path :direction :input
                              :if-does-not-exist nil)
       (let (line)
          (dotimes (i n)
            (setq line (read-line s nil s))
            (if (eq line s) (return))
            (print s)))))
```

```
(defun foo (path n)
  (with-open-file (s path :direction :input :if-does-not-exist nil)
    (let (line)
      (dotimes (i n)
```

```
required    path           "foo.cl"      (simple-array character (6))
required    n              10            fixnum
local       (dead s)       138217107     fixnum
local       (dead nil)     13743998      fixnum
local       (dead g1167)   #\?           character
local       (dead n)       32768         fixnum
local       (dead i)       0             fixnum
```

Orange highlighting indicates a macro form.
The current Macro Slide Direction will affect the next step for this macro form.

[✓] Breakpoints Enabled
[✓] Include Dead Locals

Macroexpansion Stepping Action: Over Across [✓ Into] Out

Stopped in foo .

## We just click Next and watch the forms being evaluated

- When we get to (print s), the stream object is printed and we (presumably) figure out our error:

  (print s) should be (print line)

## Things to note

- Macros are expanded. You see the macro expansion and the individual forms
- Relevant stack values are displayed. Often many are unobvious but some are what you expect
- The form being executed is displayed
- Colors indicate information about a form

## More things to note

- The Return button returns from the current form
- The Continue button usually jumps to the next breakpoint, and so often to the end of the form being evaluated (and clears the dialog)
- Closing the dialog does not stop stepping, but initiates the tty stepper
- Reopening the dialog usually reinitiates dialog stepping (after a return is entered), but closing/reopening is not recommended

## The Edit button

- Clicking on the Edit Button displays the source in a Editor pane
- When a form is highlighted in blue, it is usually the same as a form in the source and that form will be highlighted in the Editor pane
- This allows you to go right to the source of interest

## Dynamically setting breakpoints

- Breakpoints are indicated by red parentheses.
- You can add/remove breakpoints with the mouse
- Then Continue jumps to the next breakpoint

## TTY stepper

- If the IDE is not being used or the Stepper dialog is not displayed, you get the tty source stepper.
- The initial steps are the same (make sure debug is 3, compile the file, set a breakpoint, evaluate a form).
- Using the dialog is preferred because there is a lot of information to display

## The Macro Expansion Stepping  Action option

- This affects how we step through macros and into functions.
- (This is the :slide option in the tty stepper)

## Last notes

- Compiled files can be very much bigger when stepping information is stored.
- The actual running code is unchanged. The extra space comes from the annotations.
- In certain cases, the compiler can take minutes when before it took microseconds.

FRANZ INC.

## Documentation

- The tty source stepper in doc/debugging.htm#source-step-1
- The Stepper Dialog in doc/ide-menus-and-dialogs/stepper-dialog.htm
- Be sure to do updates as we will be making improvements/fixing issues

# The Source Stepper in Allegro CL 8.2

- David Margolies (dm@franz.com)
- Duane Rettig

- Questions and comments to
  support@franz.com