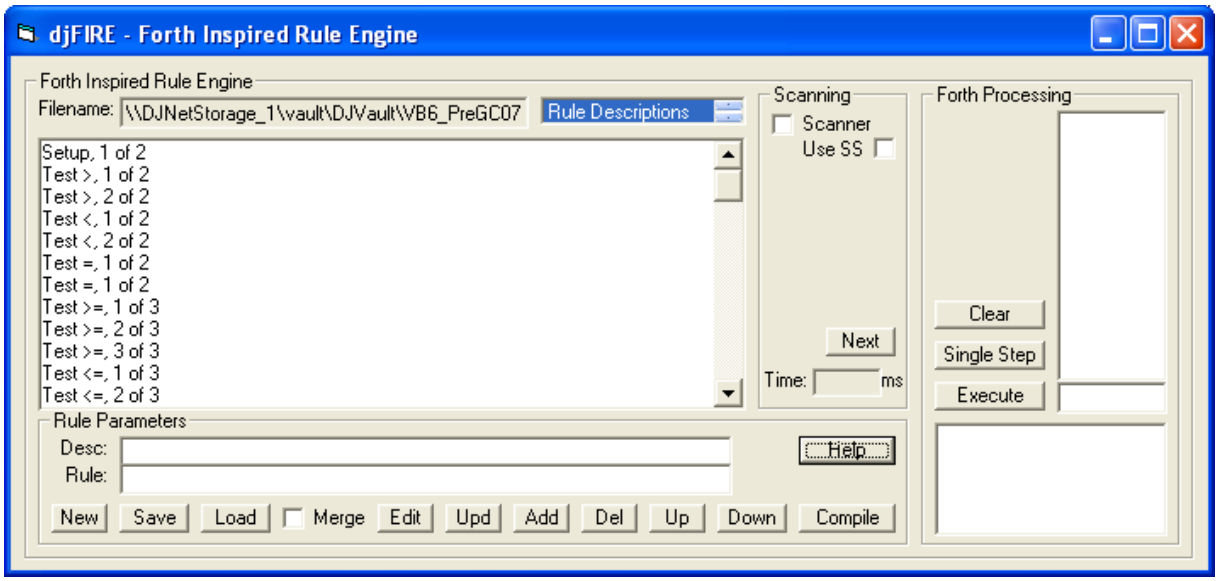
Kairos Autonomi
498 W. 8360 S.
Sandy, Utah 84070
801-255-2950 (office)
801-907-7870 (fax)
www.kairosautonomi.com

BULLETIN
BUL-015

djFire Interface Control Documentation



The djFIRE interpreter by Kairos Autonomi may be used as a stand-alone program or may be included inside of another program. The window or frame shown above is the execution console of the djFIRE scripting language.

Window/Program Operation

Filename — Filename of loaded rule, remembered on program cycles
Desc — Human language description of rule
Rule — FIRE language description of rule, machine read
List — Scrolling list of all active rules

Rule Descriptions

Scrolling list of methods for display of rule list, some methods cannot be edited
Rule Descriptions — List of human readable rules
Rule djFIRE logic — List of machine readable
Compiled Rules — List of tokenized rules
Live Rule Status — Active display of rules as they manipulate the system. Values are shown in [ ]'s
Fired Rule Descriptions — Only show rules that are active, 5 second pause before removal when inactive
Fired Rule Logic — similar as above
Fired Live Rules — similar as above

New	— Clear entries first press, then clear list to make new list
Save	— Save current list to file
Load	— Load previously saved list from file
Merge	— When checked the current list is not cleared before a new list is loaded
Edit	— Bring the selected list up in NOTEPAD. List must be reloaded is changes made
Upd	— Update or change the selected list item with new Desc and Rule; list item is uncompiled
Add	— Add new entry to list, not compiled
Up	— Move selected item up in list
Down	— Move selected item down in list
Compile	— Compile rule, convert tags to tokens
Help	— Show this file
Scanner	— Execute a scanner that executes all listed rules in sequence, continuously — Scanner operates asynchronously at 25ms scans, all rules every 25ms
Sync to Host	— Execute all rules in list, synchronized with host operation
Disable Output	— Disable the output or writing to values
Use SS	— Same as scanner but rule is executed in automatic single ste mode
Next	— Move to next rule and execute it, if no rule selected selects top rule and executes
Time	— Displays the time to execute scanner for all listed rules, average of 40 passes
Clear	— Clear forth stack
Single Step	— Single step through elements of djFIRE command line
Execute	— Execute djFIRE command line
Command Line	— Location of language commands to execute — Pressing enter after a rule is entered will execute the rule and the clear the command line
State	— Display of rule state

Qualification

There is a file, djFIREQualification.fire that can be executed to determine if the script execution is functioning properly. Whenever an error is found in the djFIRE, this qualification file should be updated to demonstrate the error and then to show that it is corrected.

Rule Language Details

- Definitions
  - TOS - Top of Stack
- Stack order
  - v4 v3 v2 v1 <--TOS
  - v1 - top of stack
  - v2 - 2nd on stack
  - ..
  - vn - further down the stack
- Numerical entry
  - All values are decimal
  - May be signed, whole or decimals
  - Prefix "&H" for hexidecimal values
- Command Entry
  - Commands are case insensitive
  - Shared variables or tags are case sensitive
  - Tags must have prefixes to be recognized
  - Prefixes determine the usage of the tag, read or write
  - Tags must be longer than 2 characters
  - All language elements must be separated by spaces
  - Extra characters are not allowed
  - djFIRE is a Forth Language-like scripting language, it is not a Forth implementation
- Branch
  - IF — Execute following elements if 1 on top of stack
  - ELSE — Execute following elements if 0 was on TOS before IF
  - THEN — End of conditional execution
  - SKIP — Skip the next x rules as defined by TOS
    - — Not implemented as of 5/6/2007
- Values
  - @ — Get value from tag
  - ! — Save value to tag
  - # — Direct value
  - / — Zero tag
  - * — Set tag to 1
  - + — Increment tag, plus
  - - — Decrement tag, minus
- Conditional
  - > — 1 on stack if v2 > v1 is true
  - < — 1 on stack if v2 < v1 is true
  - >=, => — 1 on stack if v2 >= v1 is true
  - <=, =< — 1 on stack if v2 <= v1 > is true

=, == — 1 on stack if v2 = v1 > is true
<>, !=, =! — 1 on stack if v2 <> v1 > is true
0< — 1 on stack if v1 < 0 is true
0> — 1 on stack if v1 > 0 is true
0= — 1 on stack if v1 = 0 is true

Math

+ — result on stack from v2 + v1
* — result on stack from v2 * v1
^ — result on stack from v2 ^ v1
- — result on stack from v2 - v1
/ — result on stack from v2 / v1
1+ — result on stack from v1 + 1
1- — result on stack from v1 - 1
Min — result on stack from min of v2 or v1
Max — result on stack from max of v2 or v1
ABS — result on stack from absolute value of v1
NEGATE — result on stack from v1 * -1
MOD — result on stack from v2 MOD v1

Stack

PICK — picked value on stack v4 v3 v2 v1 3 PICK leaves v4 v3 v2 v1 v3 on stack
ROLL — rolled value to top of stack v4 v3 v2 v1 3 ROLL leaves v4 v2 v1 v3 on stack
DROP — drop value from stack v2 v1 DROP leave v2 on stack
SWAP — swap top two values on stack v2 v1 SWAP leave v1 v2 on stack
DUP — duplicate top value on stack v1 DUP leave v1 v1 on stack
?DUP — duplicate top value on stack if nonzero v1 DUP leave v1 v1 on stack if v1 <> 0
ROT — roll 3rd value to top of stack v4 v3 v2 v1 ROT leaves v4 v2 v1 v3 on stack
OVER — place 3rd value to top of stack v4 v3 v2 v1 OVER leaves v4 v3 v2 v1 v3 on stack
DEPTH — place number of stack elements on TOS
EMPTY — clear all entries from stack

Rule States

TRUE — set rule state to true, not same as 1 or 0 used by IF ELSE THEN
FALSE — set rule state to false, not same as 1 or 0 used by IF ELSE THEN
?STATE — place 1 on stack if rule true, 0 if false
GREEN — invert rule reporting logic, true is false, false is true

RED — restore rule reporting logic

Logic

NOT — 2 complement of TOS
AND — result on stack from v2 AND v1
OR — result on stack from v2 OR v1
XOR — result on stack from v2 XOR v1

Execution

(— ignore Line if at beginning, no rule state update
— ignore rest of line if in mid line, performs rule state update
— used to comment out single rule line when placed as first character
QUIT — terminate rule execution, if in list then no more rules after
— stack not cleared
ABORT — terminate rule execution, if in list then no more rules after
— stack cleared
— used to comment out all following rules

System

VERSION — leaves the version of the djFIRE on stack
CHECKSUM — calculate checksum of rules and leave TOS
— Not implemented as of 5/6/2007
CREATE — Create a shared variable on system
CREATE svname svtype
svname — case sensitive alphanumeric variable name
valid chars (alpha ,numbers ,_!@#$%^&*()_-) must start with alpha
svtype — long, double, string
INIT — Line is executed once upon start of multi-line execution
— Must be first token

Examples:

Initialization of Shared Variables

This line is executed once at the beginning of a multiline execution and then not executed again. It is not even presented to the rule engine again.

INIT CREATE sv_var1 double CREATE sv_var2 long

Simple Rule

This rule checks to see if the segment distance is less than 100ft and if so it sets the direction for speed change to decrease. The @ symbol as a prefix on the shared variable fetches the SV from memory. The ! prefix causes it to store the value of -1 into the shared variable. The state of the rule is reported as TRUE if the conditional is true and executed.

```
@seg_distance 100 < IF -1 !speed_direction ELSE 1
!speed_direction THEN
```