APL64 Project Update - March 2020

Summary

This document describes some of the APL64 Project progress since May 2019.

- APL64 Project work is continuing with the full effort of the APL2000 team.
- Additional features needed for excellent compatibility with APL+Win have been implemented.
- New enhancements exclusive to the APL64 Project have been implemented.
- Testing of all project modifications have been performed.
- Performance enhancements, where possible, have been incorporated. This will be ongoing.
- Testing by APL2000 personnel of the APL64 Project with existing APL+Win customer applications has begun.

Contents

APL+Win Compatibility

The APL64 Project has excellent compatibility with APL+Win with the goal of being able to load any APL+Win workspace in the APL64 Project with minimal or no modification to the APL source code.

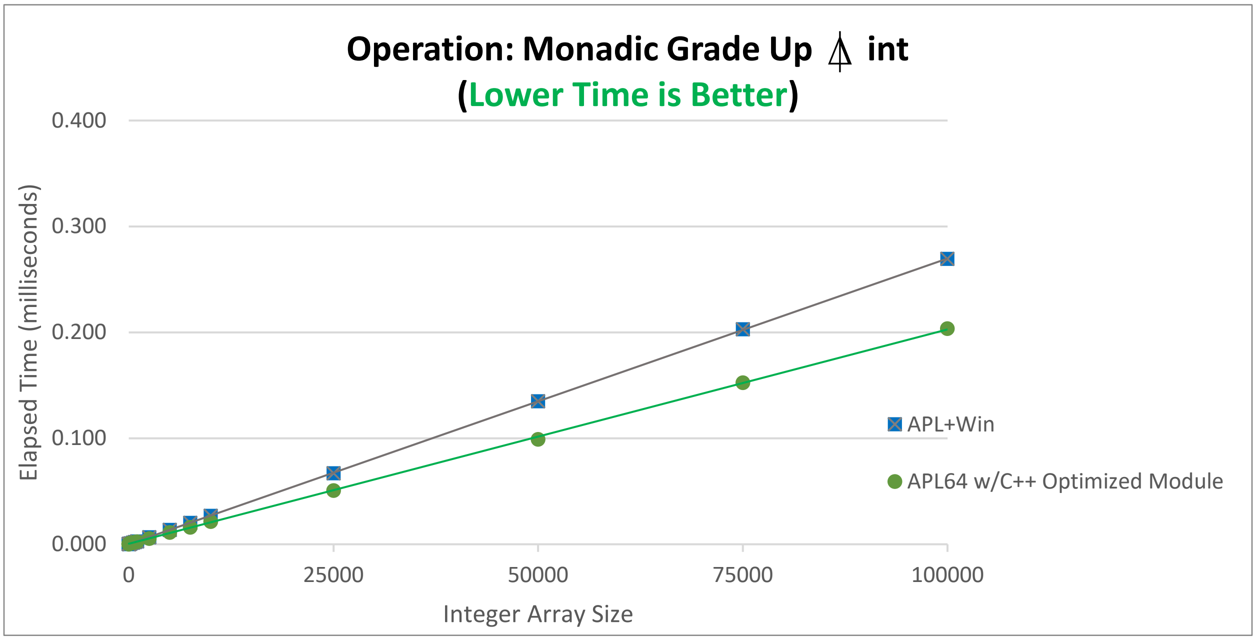
- APL64 Project quote-quad primitive function support
- License and Activation technology in APL64 Project developer version, including a new option for a 'group license' to reduce the need for individual workstation activation
- ⎕NFE (Native File functions with Encoding) system function with improved performance and elimination on the dependency of the ⎕CSE system function
- Option to set ⎕PW based on window width implemented
- More control structures implemented
- Selective specification implemented with enhanced performance
- Enhanced syntax coloring in the Session Command Line, Session History, Function Editor and Debugger
- Numerous system functions implemented: ⎕COPY, ⎕PCOPY, ⎕PSAVE, ⎕SAVE, ⎕LOAD, ⎕QLOAD, ⎕XLOAD, ⎕DEF, ⎕DEFL, ⎕CR, ⎕CRL, ⎕CRLPC, ⎕ERASE, ⎕EX, ⎕IDLIST, ⎕NL, ⎕SS, ⎕SYMB, ⎕EVAL, ⎕VR, ⎕SPLIT, ⎕MIX
- Numerous APL primitive functions implemented: Base value (⊥), Representation (⊤), ⎕SPLIT, ⎕Mix, Catenate, Laminate, Enclose, Partition, Rotate
- Interrupt performance in the APL64 Project session improved
- APL64 Project runtime Executable, ⎕CSE and APL64 Project ActiveX implementations continuing
- APL64 Project product installer under development

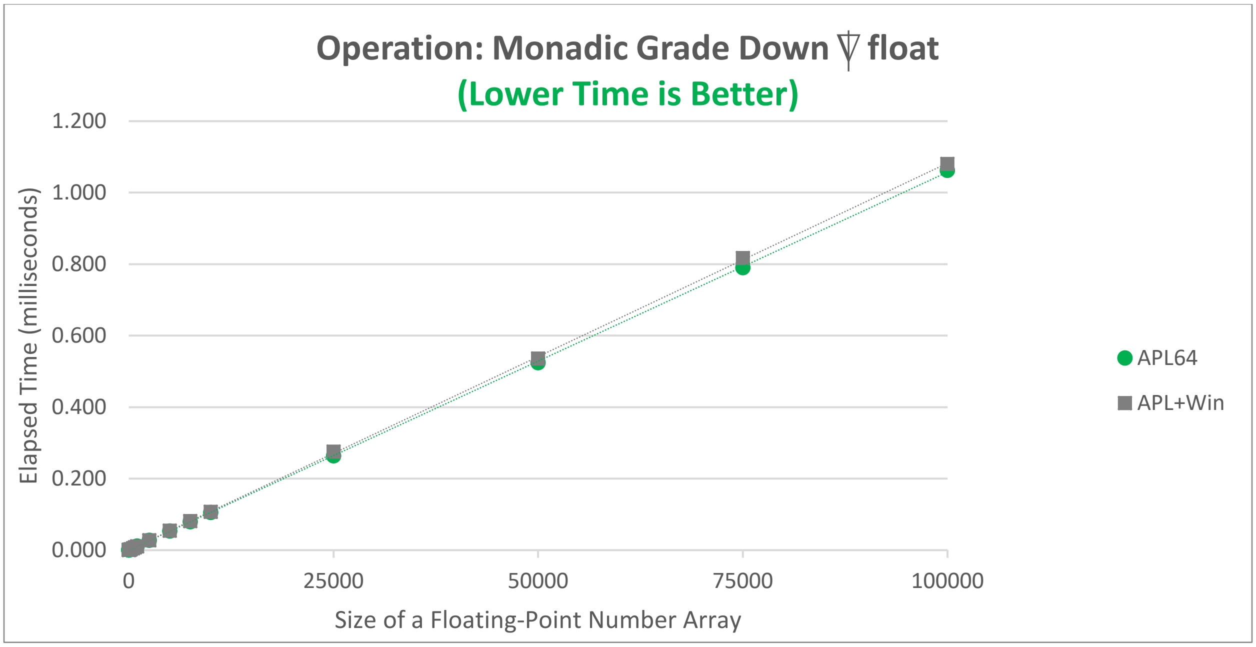
Performance Optimization

After an APL64 Project primitive or system function is implemented and tested, it is considered for performance optimization. The optimization process requires considerable effort and skill. Previous optimizations were performed for the arithmetic primitive functions (+-×÷). Some of the additional APL64 Project optimizations are illustrated here. Performance results will vary depending on the user provided environment.

Grade Up & Grade Down Performance Optimization

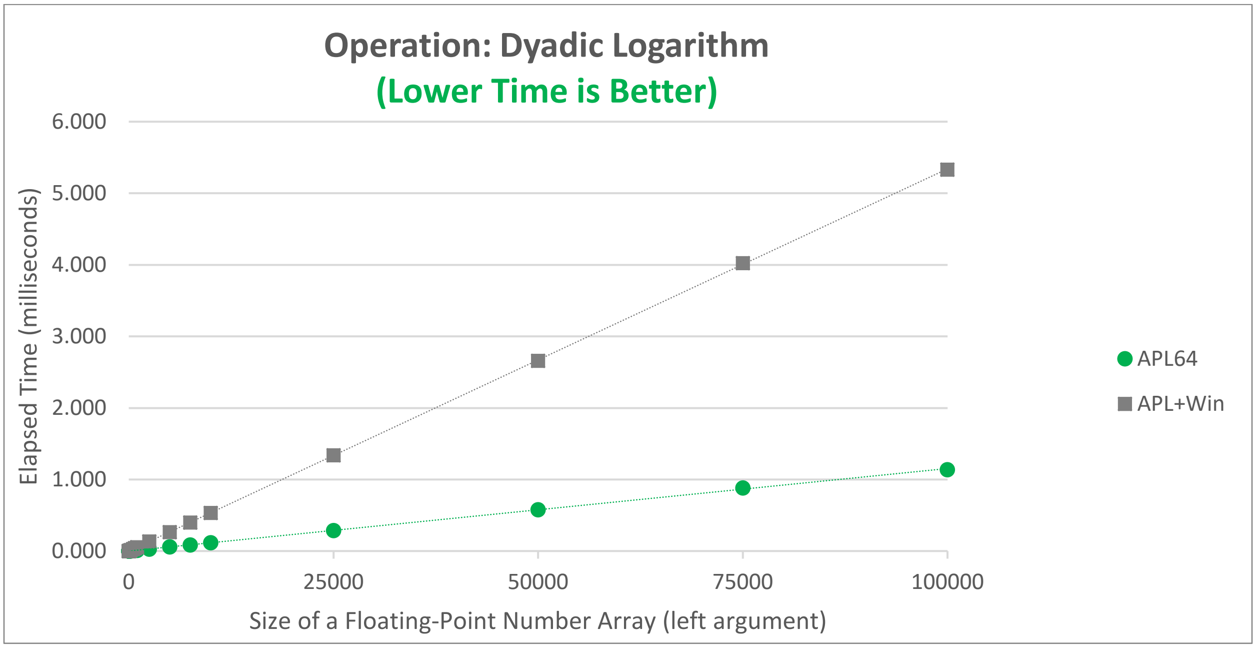
The APL64 Project grade up and grade down primitive system functions provide for an optional, automatically-loaded-if-appropriate, Windows-native C++ module to improve performance over that of APL+Win.

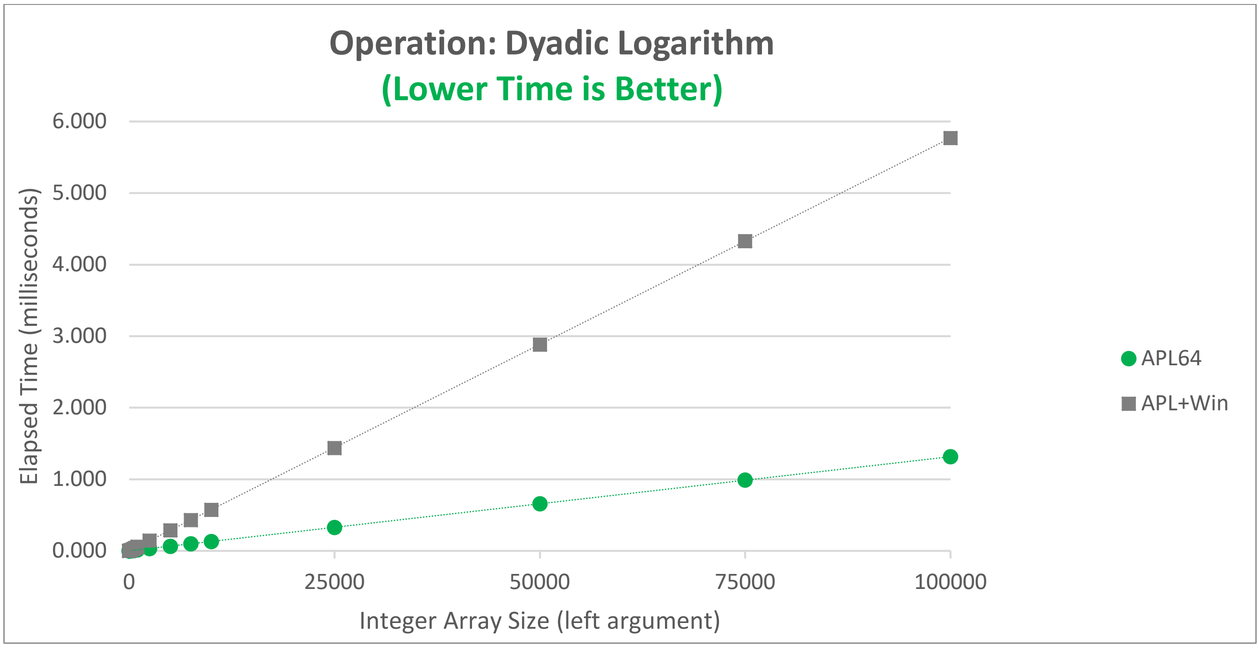




Logarithm Performance Optimization

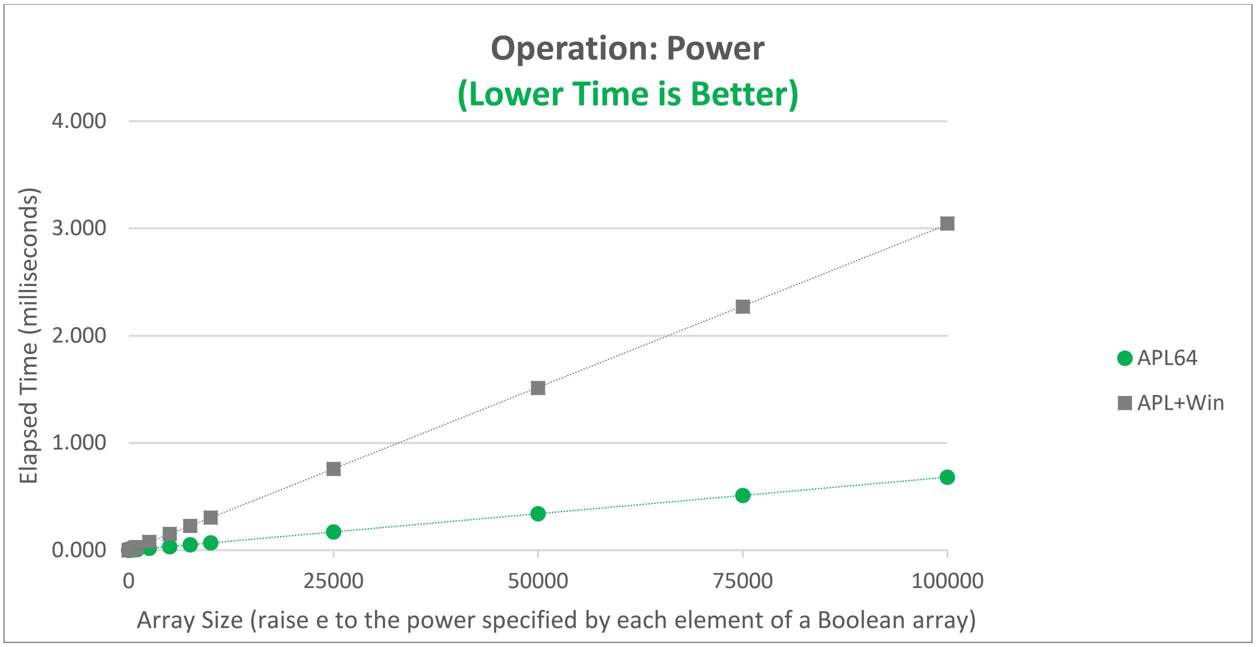
The APL64 Project logarithm primitive function performance, without the need for a Windows-native C++ module, has been improved to exceed that of APL+Win.

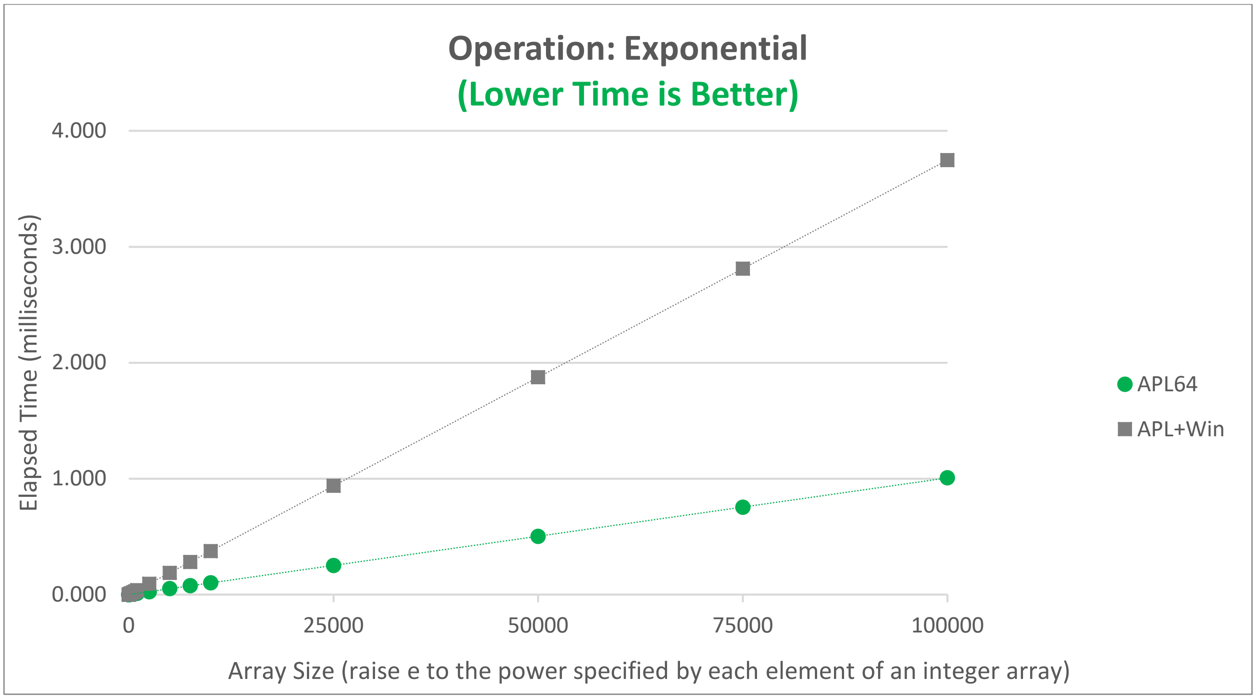


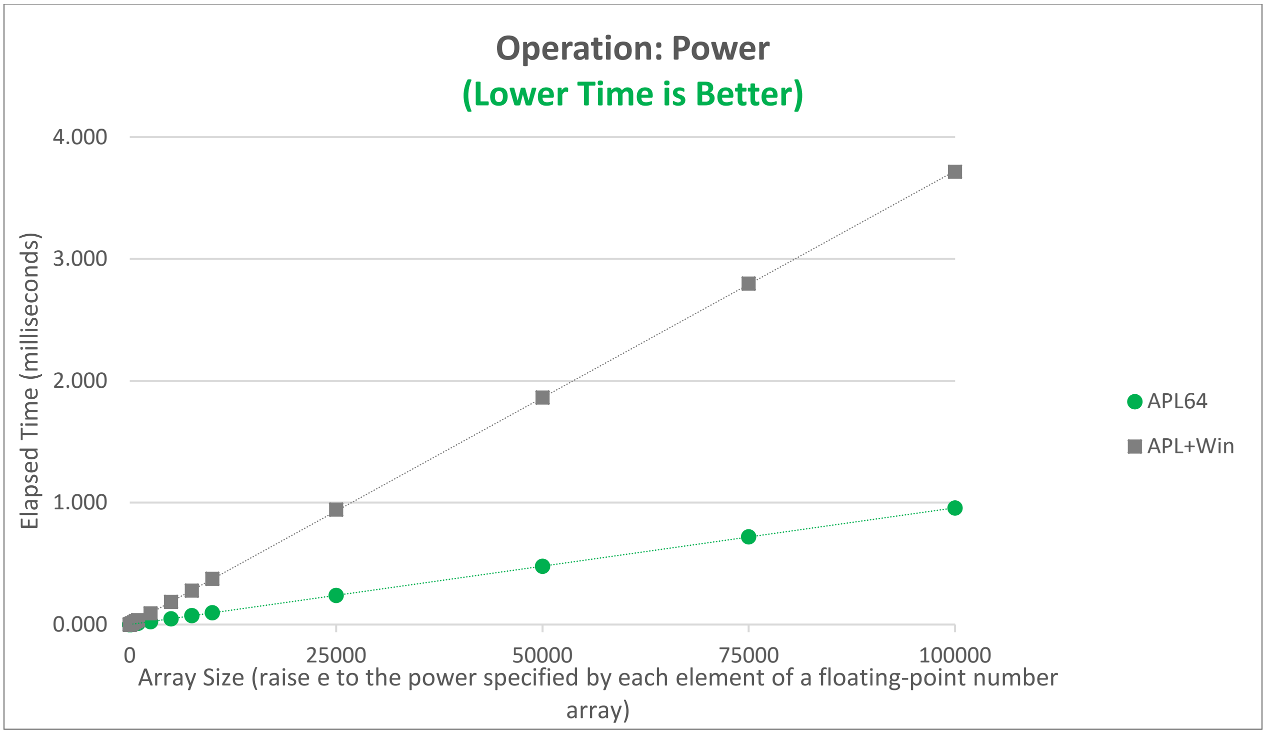


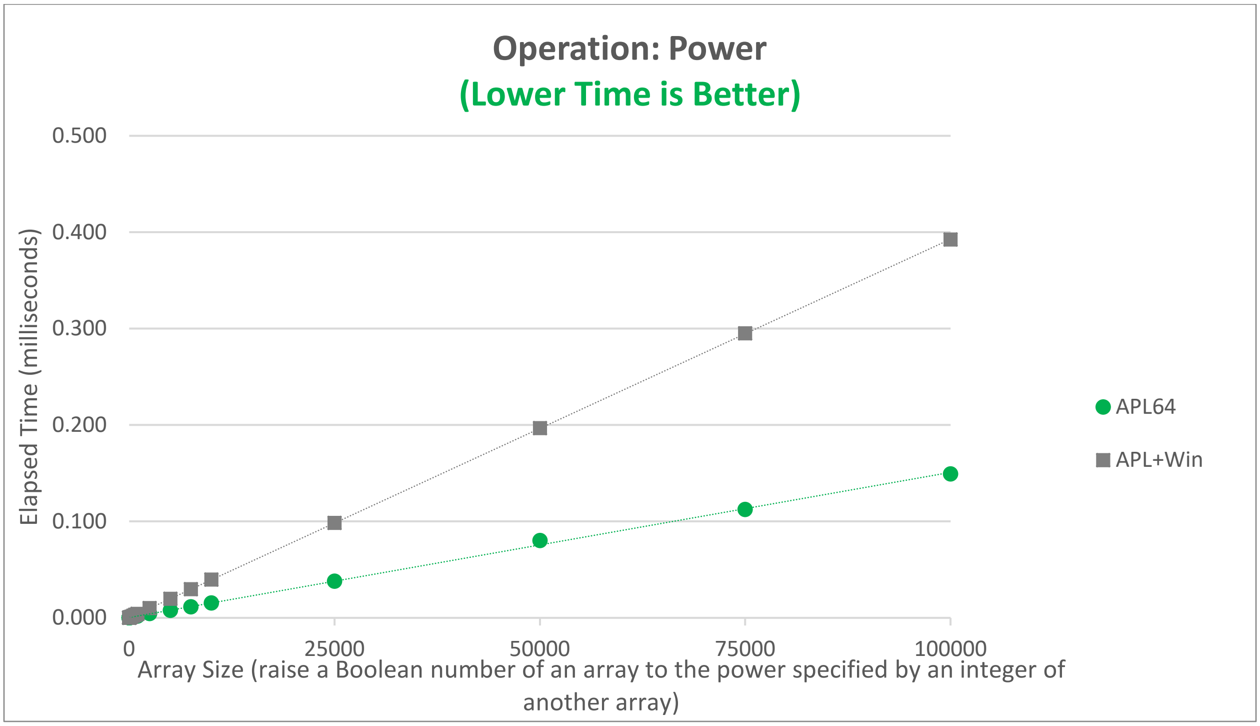
Power (Exponential) Performance Optimization

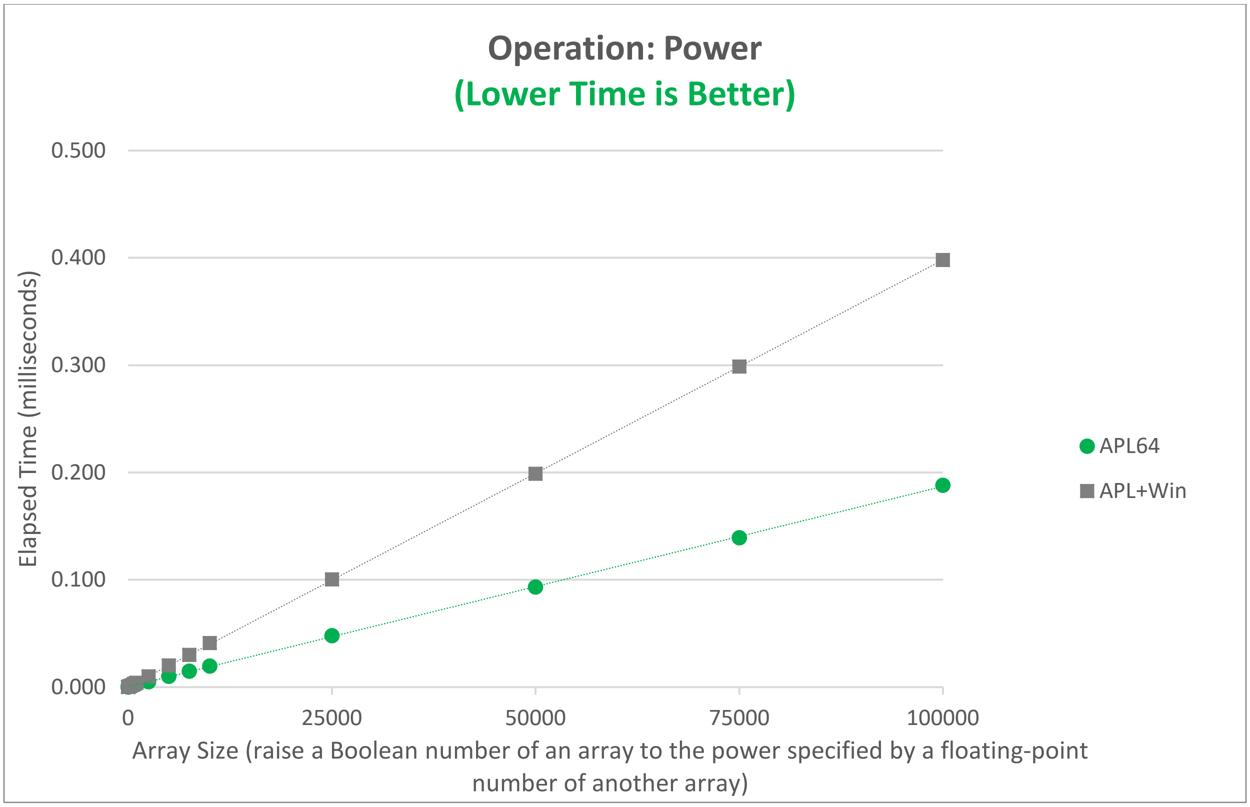
The APL64 Project power primitive function performance, without the need for a Windows-native C++ module, has been improved to exceed that of APL+Win.

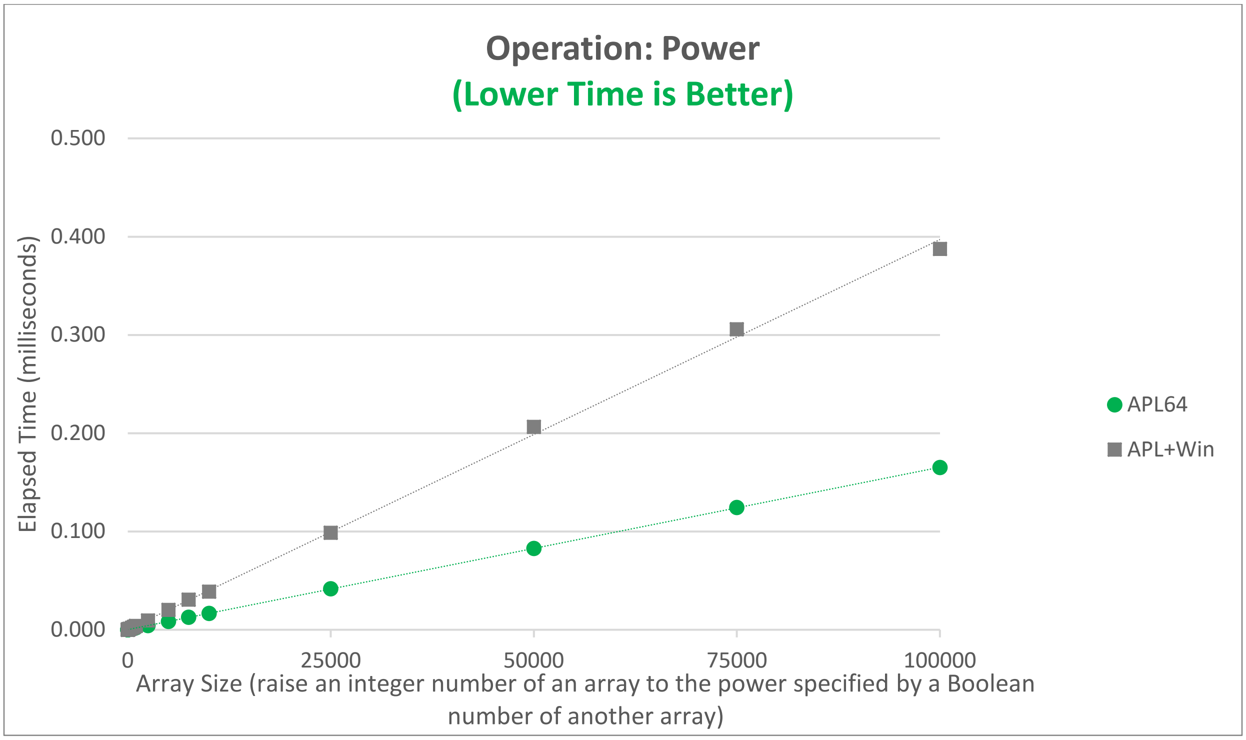

Operation: Power
(Lower Time is Better)
Elapsed Time (milliseconds)
4.000
3.000
2.000
1.000
0.000
0
25000
50000
75000
100000
Array Size (raise e to the power specified by each element of a Boolean array)
● APL64
■ APL+Win

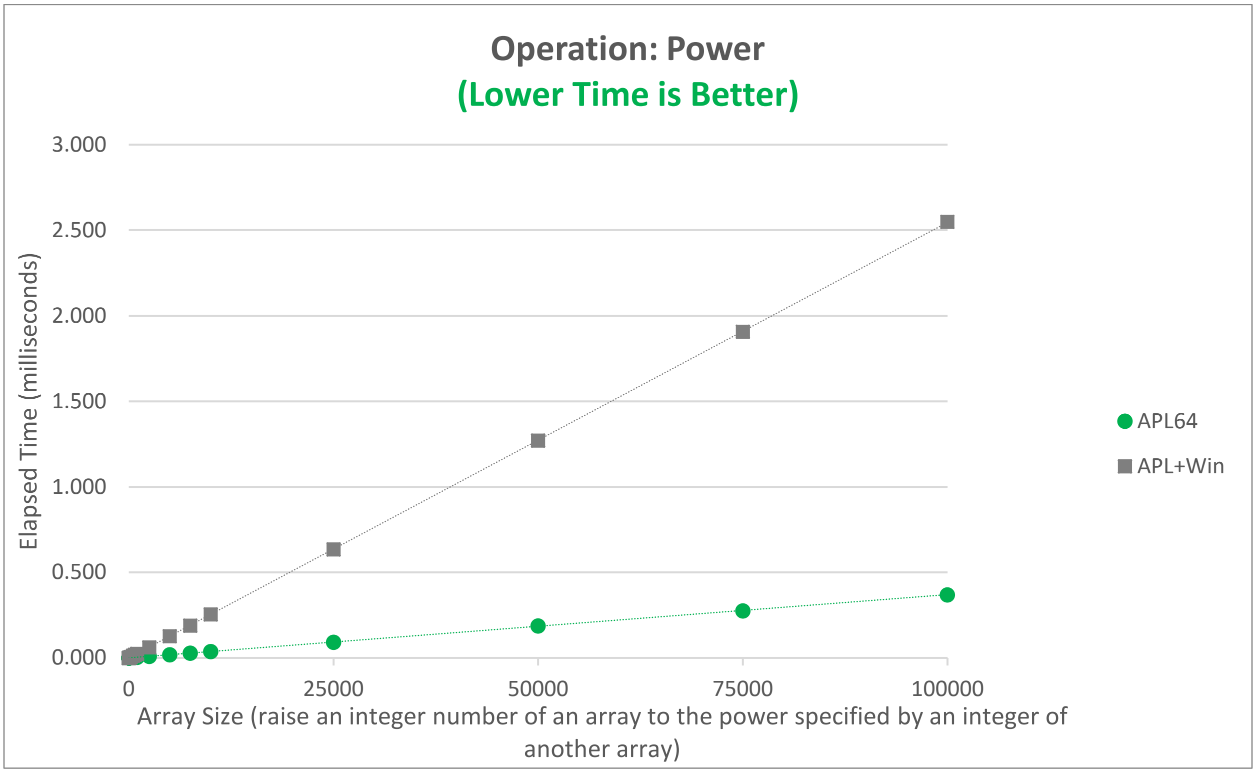

Operation: Exponential
(Lower Time is Better)
Elapsed Time (milliseconds)
4.000
3.000
2.000
1.000
0.000
0
25000
50000
75000
100000
Array Size (raise e to the power specified by each element of an integer array)
● APL64
■ APL+Win

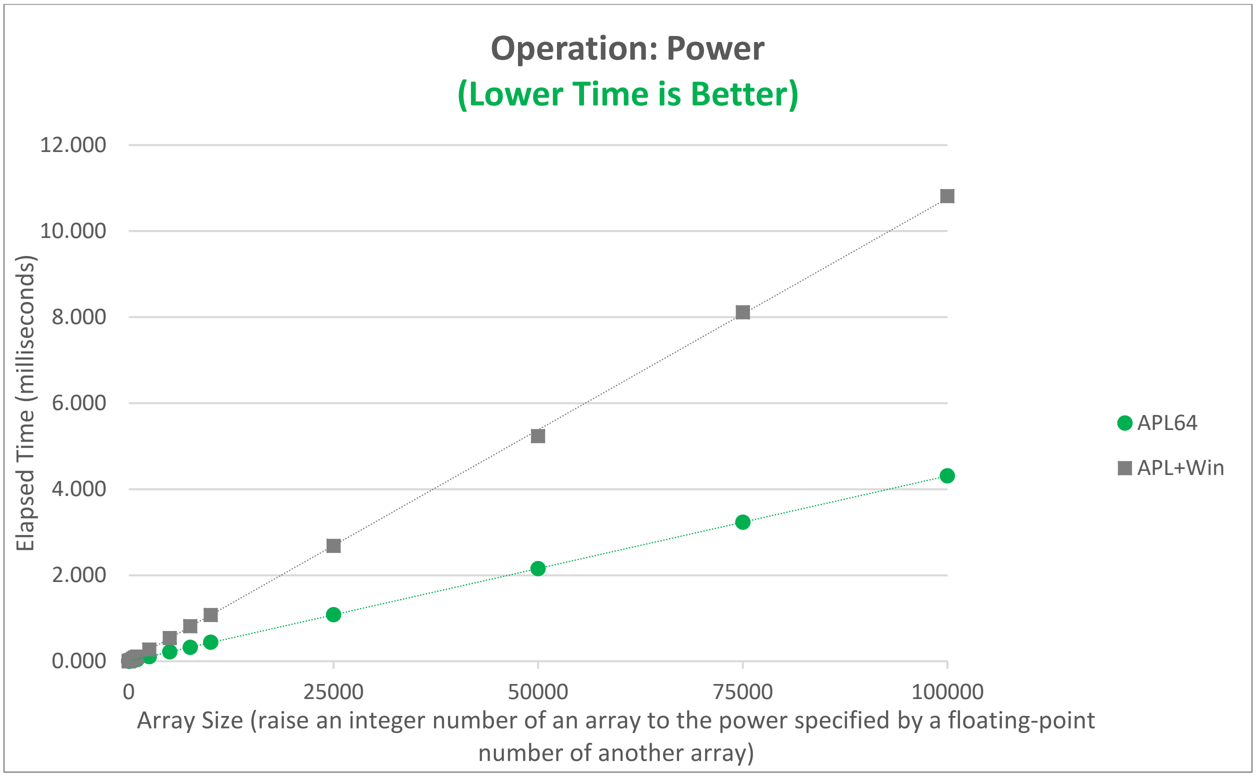

Operation: Power
(Lower Time is Better)
4.000
3.000
2.000
1.000
0.000
Elapsed Time (milliseconds)
0
25000
50000
75000
100000
Array Size (raise e to the power specified by each element of a floating-point number array)
● APL64
■ APL+Win

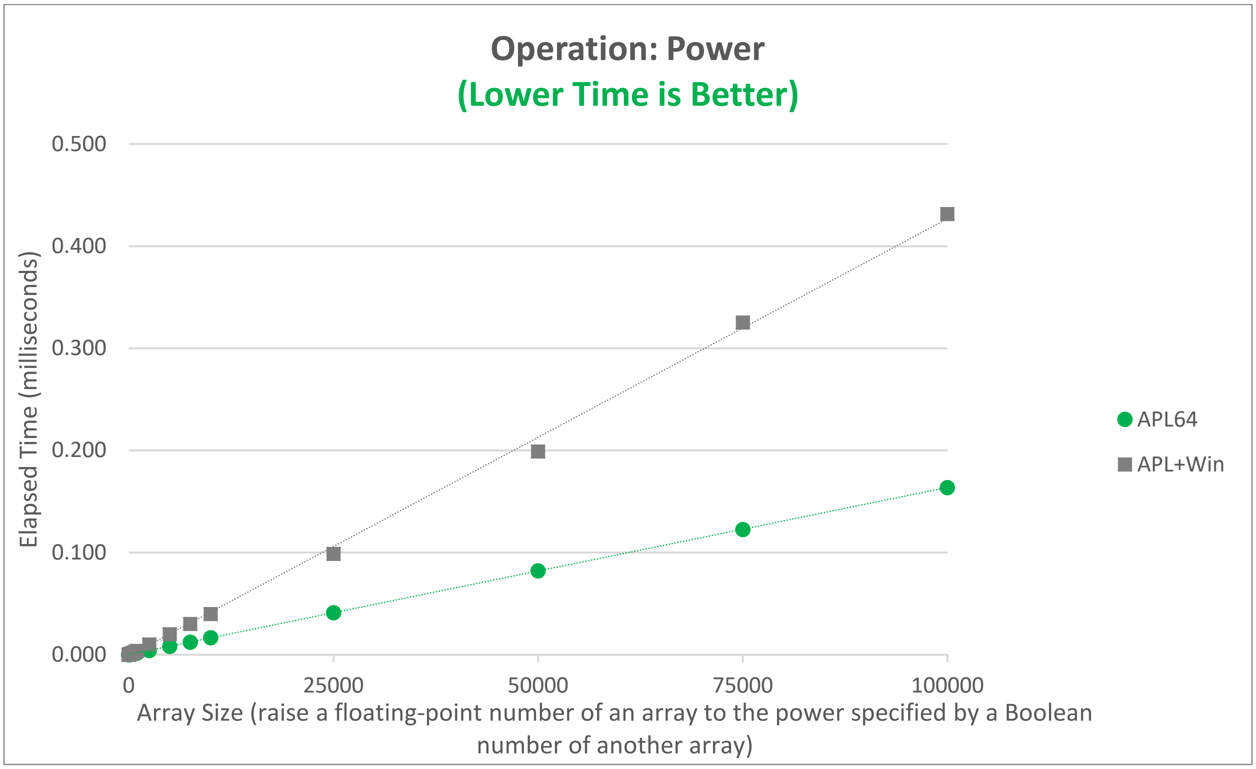

Operation: Power
(Lower Time is Better)
0.500
0.400
0.300
0.200
0.100
0.000
Elapsed Time (milliseconds)
0
25000
50000
75000
100000
Array Size (raise a Boolean number of an array to the power specified by an integer of another array)
● APL64
■ APL+Win

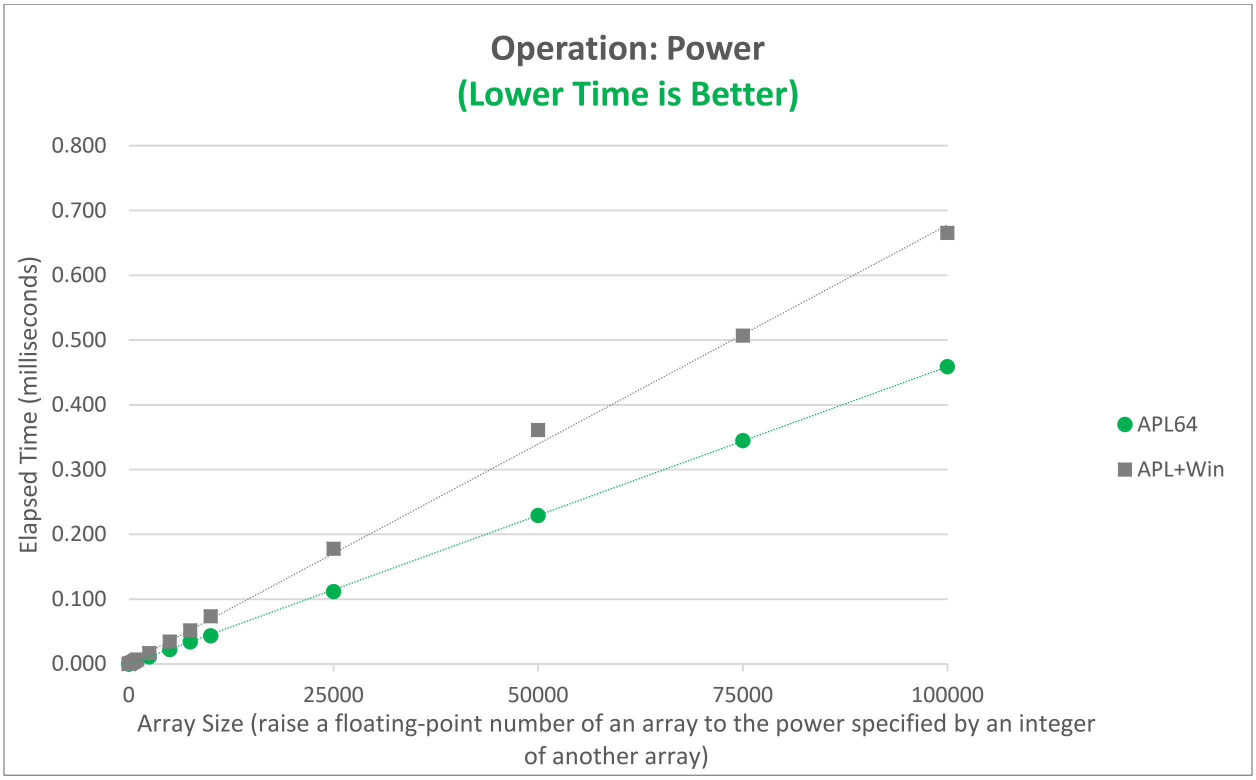

Operation: Power
(Lower Time is Better)
Elapsed Time (milliseconds)
0.500
0.400
0.300
0.200
0.100
0.000
0
25000
50000
75000
100000
Array Size (raise a Boolean number of an array to the power specified by a floating-point number of another array)
● APL64
■ APL+Win


Operation: Power
(Lower Time is Better)
Elapsed Time (milliseconds)
0.500
0.400
0.300
0.200
0.100
0.000
0
25000
50000
75000
100000
Array Size (raise an integer number of an array to the power specified by a Boolean number of another array)
● APL64
■ APL+Win


Operation: Power
(Lower Time is Better)
3.000
2.500
2.000
1.500
1.000
0.500
0.000
Elapsed Time (milliseconds)
0
25000
50000
75000
100000
Array Size (raise an integer number of an array to the power specified by an integer of another array)
● APL64
■ APL+Win

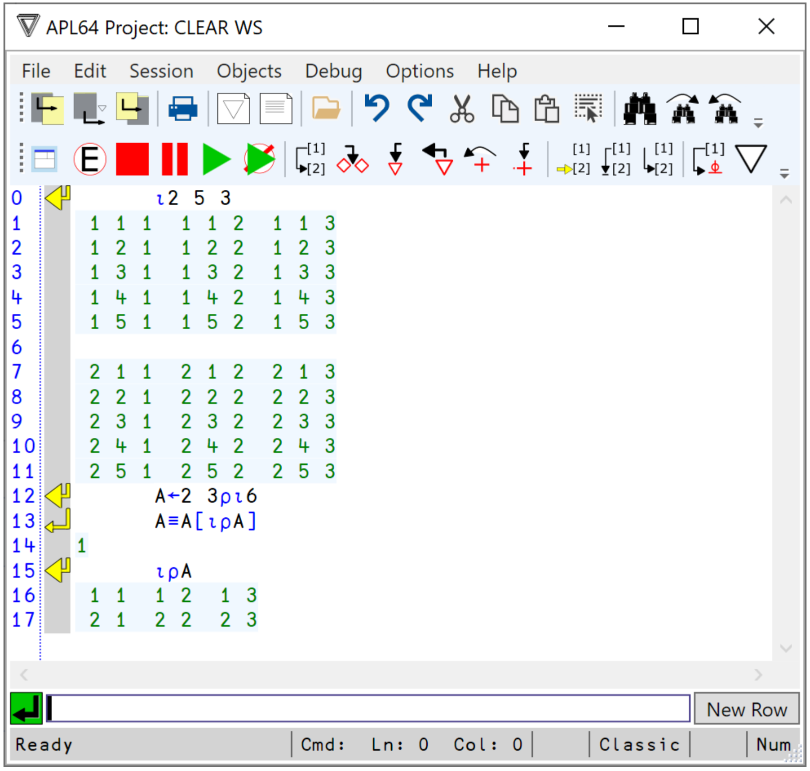

Operation: Power
(Lower Time is Better)
12.000
10.000
8.000
6.000
4.000
2.000
0.000
Elapsed Time (milliseconds)
0
25000
50000
75000
100000
Array Size (raise an integer number of an array to the power specified by a floating-point number of another array)
● APL64
■ APL+Win


Operation: Power
(Lower Time is Better)
Elapsed Time (milliseconds)
0.500
0.400
0.300
0.200
0.100
0.000
0
25000
50000
75000
100000
Array Size (raise a floating-point number of an array to the power specified by a Boolean number of another array)
● APL64
■ APL+Win


Operation: Power
(Lower Time is Better)
Elapsed Time (milliseconds)
0.800
0.700
0.600
0.500
0.400
0.300
0.200
0.100
0.000
0
25000
50000
75000
100000
Array Size (raise a floating-point number of an array to the power specified by an integer of another array)
● APL64
■ APL+Win

Monadic Iota (ι) Enhanced to Support Non-Scalar Arguments

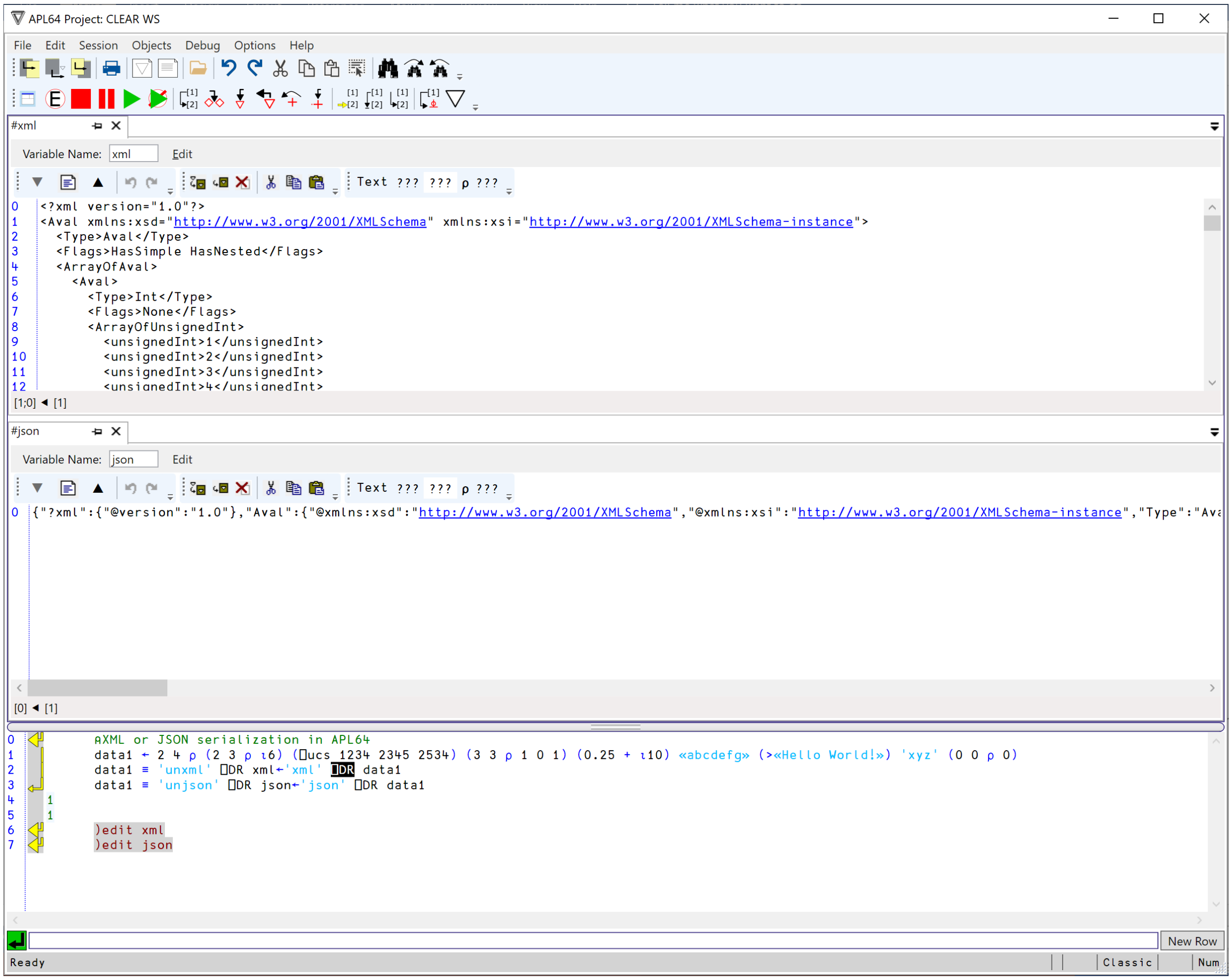
The monadic Iota (ι) APL64 Project primitive function now supports non-singleton arguments.



```
      ⍳2 5 3
 1 1 1   1 1 2   1 1 3
 1 2 1   1 2 2   1 2 3
 1 3 1   1 3 2   1 3 3
 1 4 1   1 4 2   1 4 3
 1 5 1   1 5 2   1 5 3

 2 1 1   2 1 2   2 1 3
 2 2 1   2 2 2   2 2 3
 2 3 1   2 3 2   2 3 3
 2 4 1   2 4 2   2 4 3
 2 5 1   2 5 2   2 5 3
      A←2 3⍴⍳6
      A≡A[⍳⍴A]
1
      ⍳⍴A
 1 1  1 2  1 3
 2 1  2 2  2 3
```

⎕DR Enhanced to Support XML & JSON Serialization & Deserialization



```
<?xml version="1.0"?>
<Aval xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Type>Aval</Type>
  <Flags>HasSimple HasNested</Flags>
  <ArrayOfAval>
    <Aval>
      <Type>Int</Type>
      <Flags>None</Flags>
      <ArrayOfUnsignedInt>
        <unsignedInt>1</unsignedInt>
        <unsignedInt>2</unsignedInt>
        <unsignedInt>3</unsignedInt>
        <unsignedInt>4</unsignedInt>
```

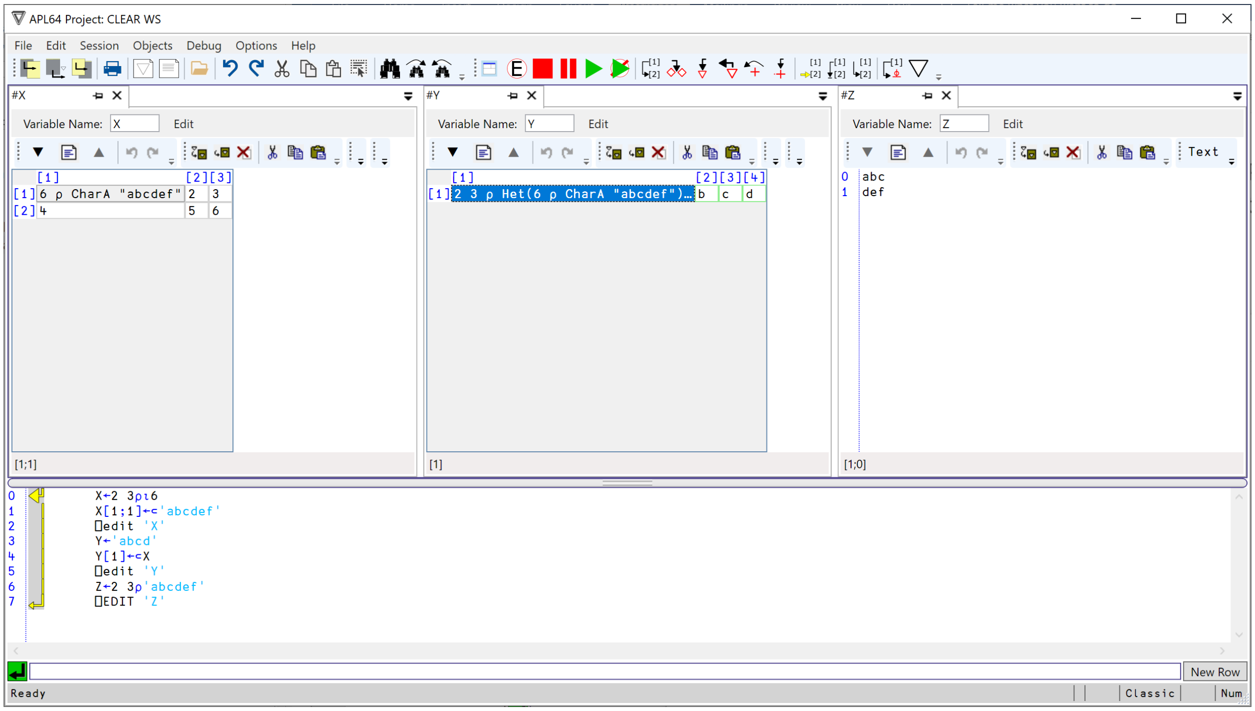
```
{"?xml":{"@version":"1.0"},"Aval":{"@xmlns:xsd":"http://www.w3.org/2001/XMLSchema","@xmlns:xsi":"http://www.w3.org/2001/XMLSchema-instance","Type":"Ava
```

```
      ⍝XML or JSON serialization in APL64
      data1 ← 2 4 ρ (2 3 ρ ι6) (⎕ucs 1234 2345 2534) (3 3 ρ 1 0 1) (0.25 + ι10) «abcdefg» (>«Hello World!») 'xyz' (0 0 ρ 0)
      data1 ≡ 'unxml' ⎕DR xml←'xml' ⎕DR data1
      data1 ≡ 'unjson' ⎕DR json←'json' ⎕DR data1
1
1

      )edit xml
      )edit json
```

Unified Variable Editor

The new unified variable editor in the APL64 Project can review and edit an APL variable of any type, including heterogeneous and nested variables, unlike APL+Win. The APL+Win graphical editor is limited to simple text or simple numeric variable editing. Manual, element-level editing of variables is supported in APL64 like in APL+Win.



	[1]	[2]	[3]
[1]	6 ρ CharA "abcdef"	2	3
[2]	4	5	6

[1;1]

	[1]	[2]	[3]	[4]
[1]	2 3 ρ Het(6 ρ CharA "abcdef")…	b	c	d

[1]

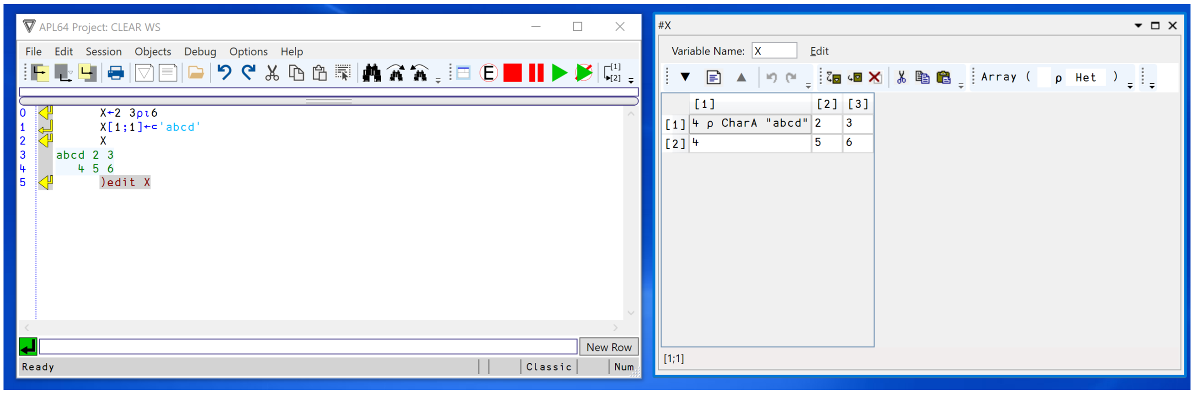
```
0 abc
1 def
```

[1;0]

```
      X←2 3ρι6
      X[1;1]←⊂'abcdef'
      ⎕edit 'X'
      Y←'abcd'
      Y[1]←⊂X
      ⎕edit 'Y'
      Z←2 3ρ'abcdef'
      ⎕EDIT 'Z'
```

Floating and Docked Editor Panes

In the APL64 Project, variable and user-defined function editor panes may be floated separately from the main session window:



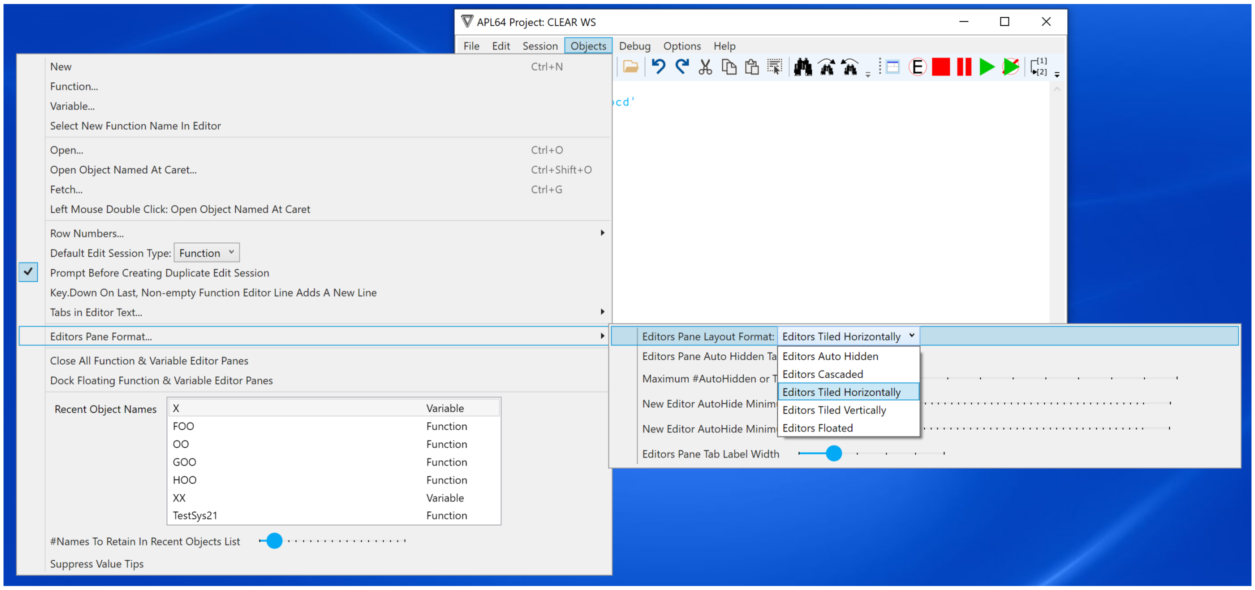
```
      X←2 3ρι6
      X[1;1]←⊂'abcd'
      X
abcd 2 3
   4 5 6
      )edit X
```

	[1]	[2]	[3]
[1]	4 ρ CharA "abcd"	2	3
[2]	4	5	6

[1;1]

Editor Layout Options in the Session

Variable and function editor pane layouts can be selected and modified in the APL64 Project session:


APL64 Project: CLEAR WS
File
Edit
Session
Objects
Debug
Options
Help
New
Ctrl+N
Function...
Variable...
Select New Function Name In Editor
Open...
Ctrl+O
Open Object Named At Caret...
Ctrl+Shift+O
Fetch...
Ctrl+G
Left Mouse Double Click: Open Object Named At Caret
Row Numbers...
Prompt Before Creating Duplicate Edit Session
Key.Down On Last, Non-empty Function Editor Line Adds A New Line
Tabs in Editor Text...
Editors Pane Format...
Close All Function & Variable Editor Panes
Dock Floating Function & Variable Editor Panes
#Names To Retain In Recent Objects List
Suppress Value Tips
Editors Pane Auto Hidden Ta
Maximum #AutoHidden or T
New Editor AutoHide Minim
New Editor AutoHide Minim
Editors Pane Tab Label Width
Editors Auto Hidden
Editors Cascaded
Editors Tiled Horizontally
Editors Tiled Vertically
Editors Floated

Default Edit Session Type: Function

☑ Prompt Before Creating Duplicate Edit Session

Editors Pane Layout Format: Editors Tiled Horizontally

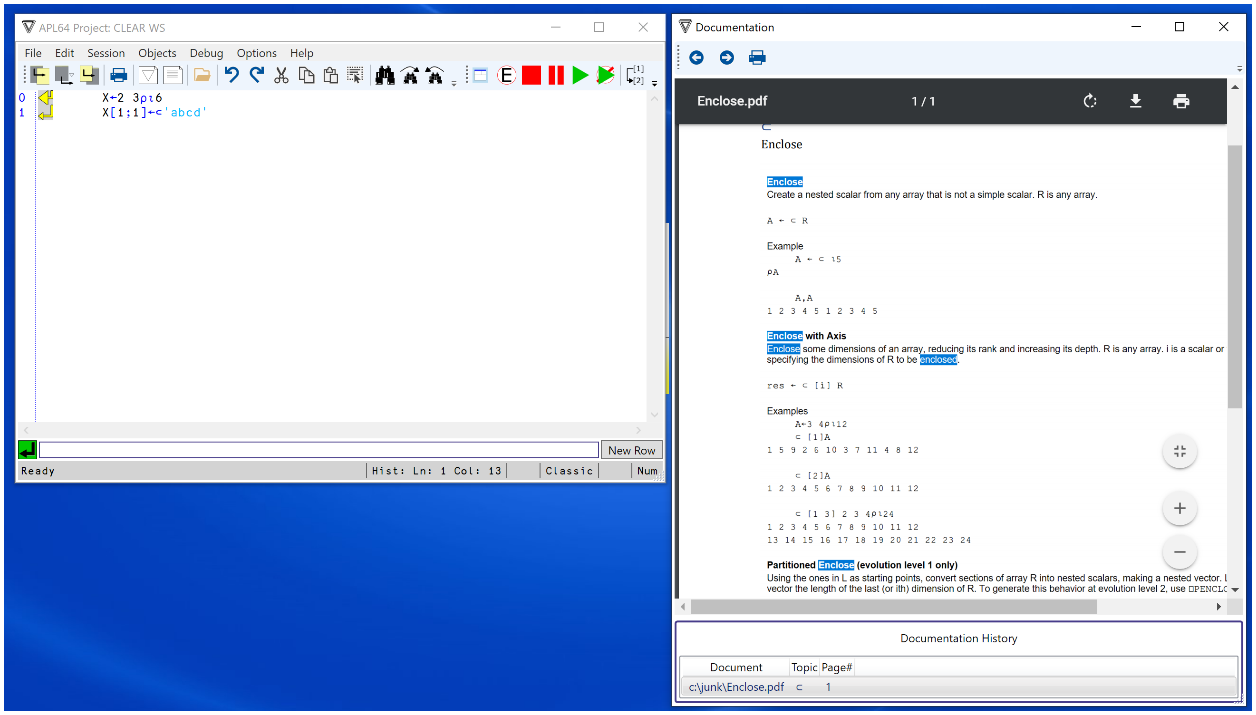
Recent Object Names	
X	Variable
FOO	Function
OO	Function
GOO	Function
HOO	Function
XX	Variable
TestSys21	Function

Context-Sensitive APL Documentation in the Session

Context-sensitive documentation for APL primitive functions, system variables, functions and commands has been implemented in the APL64 Project.

When the cursor is placed on an element in the session and the F1 key is pressed, the applicable documentation will be displayed.

The documentation window contains a ‘documentation history’ list so the user may return to documentation elements previously-viewed during the session. The documentation window is floated independently of the main session window. The documentation pdf-format supports bookmarks and table of contents containing APL glyphs and APL glyph names. This feature is new to the APL64 Project and is not available in APL+Win.



```
0        X←2 3⍴⍳6
1        X[1;1]←⊂'abcd'
```

Ready | Hist: Ln: 1 Col: 13 | Classic | Num

⊂

Enclose

Enclose
Create a nested scalar from any array that is not a simple scalar. R is any array.

```
A ← ⊂ R
```

Example

```
      A ← ⊂ ⍳5
⍴A

      A,A
1 2 3 4 5 1 2 3 4 5
```

Enclose with Axis
Enclose some dimensions of an array, reducing its rank and increasing its depth. R is any array. i is a scalar or specifying the dimensions of R to be enclosed.

```
res ← ⊂ [i] R
```

Examples

```
      A←3 4⍴⍳12
      ⊂ [1]A
1 5 9 2 6 10 3 7 11 4 8 12

      ⊂ [2]A
1 2 3 4 5 6 7 8 9 10 11 12

      ⊂ [1 3] 2 3 4⍴⍳24
1 2 3 4 5 6 7 8 9 10 11 12
13 14 15 16 17 18 19 20 21 22 23 24
```

Partitioned Enclose (evolution level 1 only)
Using the ones in L as starting points, convert sections of array R into nested scalars, making a nested vector. L vector the length of the last (or ith) dimension of R. To generate this behavior at evolution level 2, use ⎕PENCLC

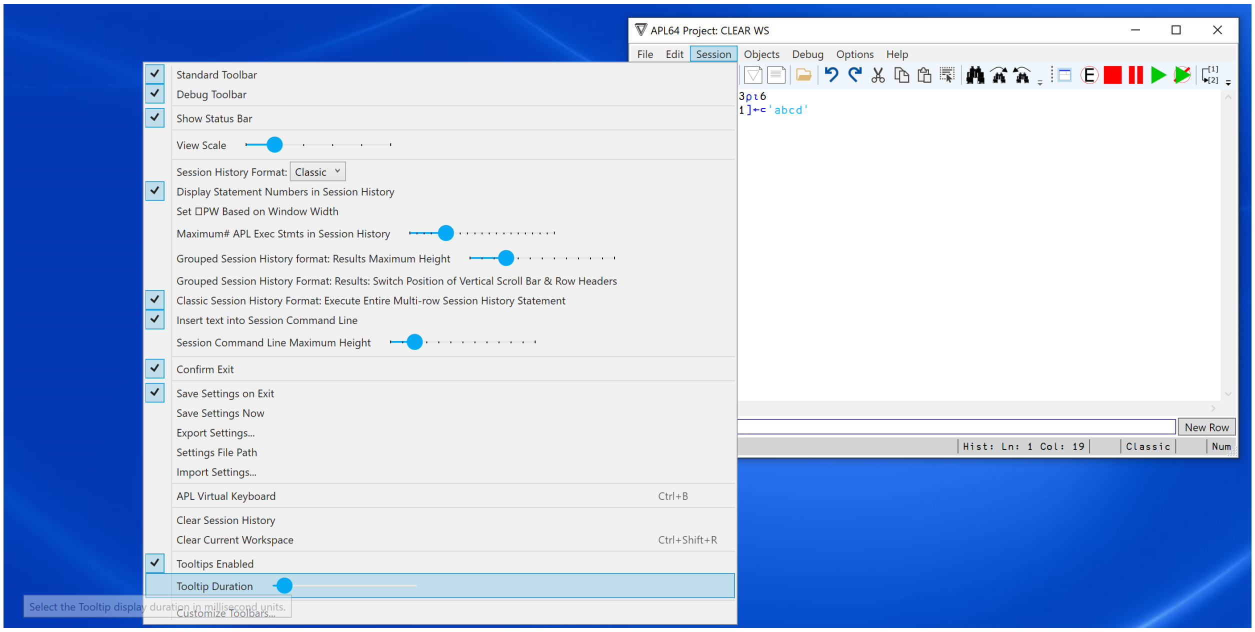
Documentation History

Document	Topic	Page#
c:\junk\Enclose.pdf	⊂	1

Tooltips in the Session

Tooltips are provided for most APL64 Project Session menu and toolbar items.

- Option to display tooltips for the session
- Option to set the tooltip duration for the session



```
3⍴⍳6
1]←⊂'abcd'
```

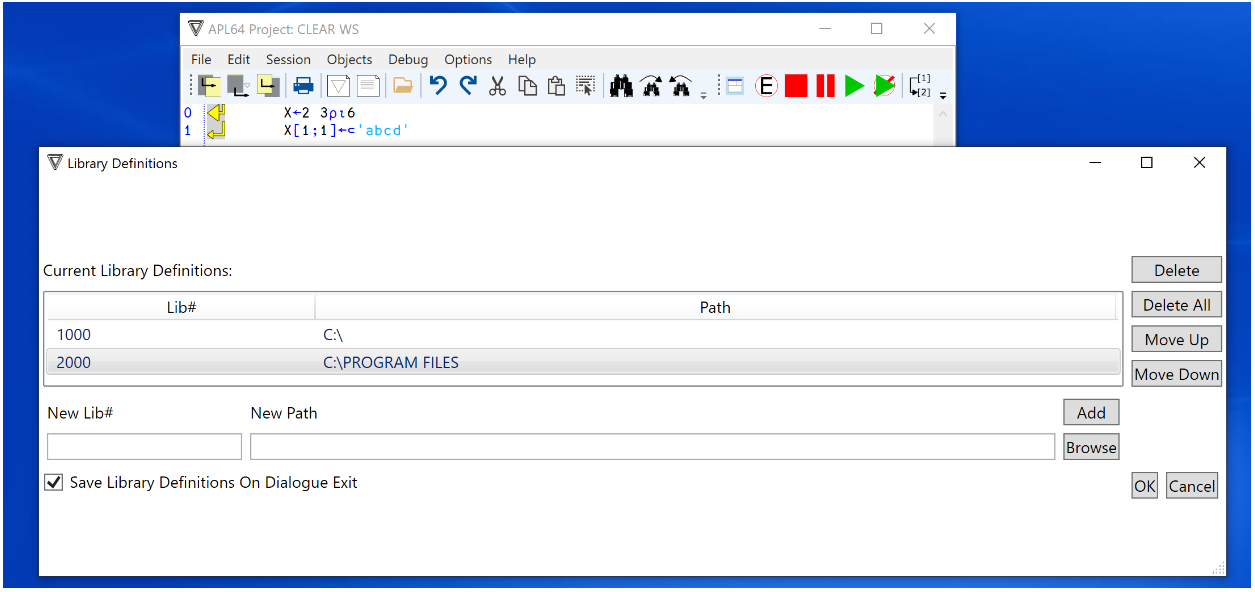
Hist: Ln: 1 Col: 19 | Classic | Num

☑ Standard Toolbar
☑ Debug Toolbar
☑ Show Status Bar
View Scale
Session History Format: Classic
☑ Display Statement Numbers in Session History
Set ⎕PW Based on Window Width
Maximum# APL Exec Stmts in Session History
Grouped Session History format: Results Maximum Height
Grouped Session History Format: Results: Switch Position of Vertical Scroll Bar & Row Headers
☑ Classic Session History Format: Execute Entire Multi-row Session History Statement
☑ Insert text into Session Command Line
Session Command Line Maximum Height
☑ Confirm Exit
☑ Save Settings on Exit
Save Settings Now
Export Settings...
Settings File Path
Import Settings...
APL Virtual Keyboard Ctrl+B
Clear Session History
Clear Current Workspace Ctrl+Shift+R
☑ Tooltips Enabled
Tooltip Duration
Customize Toolbars...

Select the Tooltip display duration in millisecond units.

Library Definitions Dialog

The APL64 Project library definitions dialog has been implemented and enhanced.



```
      X←2 3⍴⍳6
      X[1;1]←⊂'abcd'
```

Lib#	Path
1000	C:\
2000	C:\PROGRAM FILES

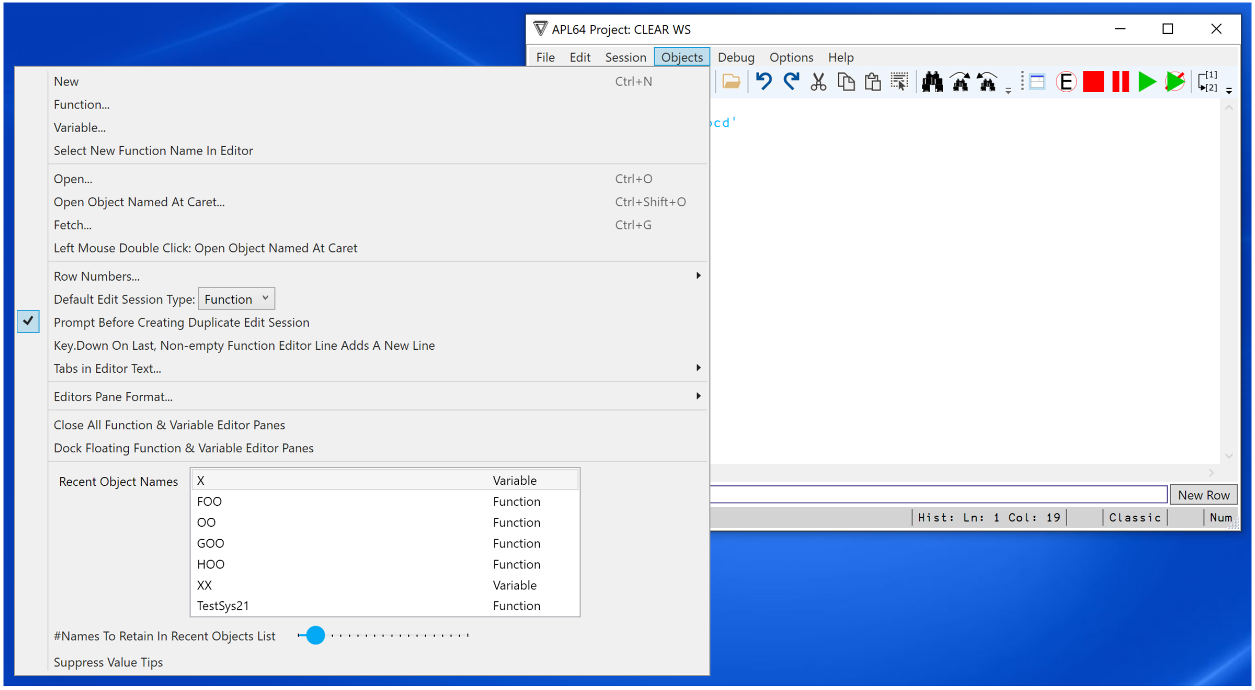
☑ Save Library Definitions On Dialogue Exit

Support for Multiple Monitors

The APL64 Project session includes improved support for multiple monitors with varying resolution.

Recent Objects List

The enhanced recent objects list in the APL64 Project session illustrates the object name and object type.


APL64 Project: CLEAR WS
File
Edit
Session
Objects
Debug
Options
Help
New Ctrl+N
Function...
Variable...
Select New Function Name In Editor
Open... Ctrl+O
Open Object Named At Caret... Ctrl+Shift+O
Fetch... Ctrl+G
Left Mouse Double Click: Open Object Named At Caret
Row Numbers...
Tabs in Editor Text...
Editors Pane Format...
Close All Function & Variable Editor Panes
Dock Floating Function & Variable Editor Panes
#Names To Retain In Recent Objects List
Suppress Value Tips
New Row
Hist: Ln: 1 Col: 19
Classic
Num

Default Edit Session Type: Function

☑ Prompt Before Creating Duplicate Edit Session

Key.Down On Last, Non-empty Function Editor Line Adds A New Line

Recent Object Names

X	Variable
FOO	Function
OO	Function
GOO	Function
HOO	Function
XX	Variable
TestSys21	Function

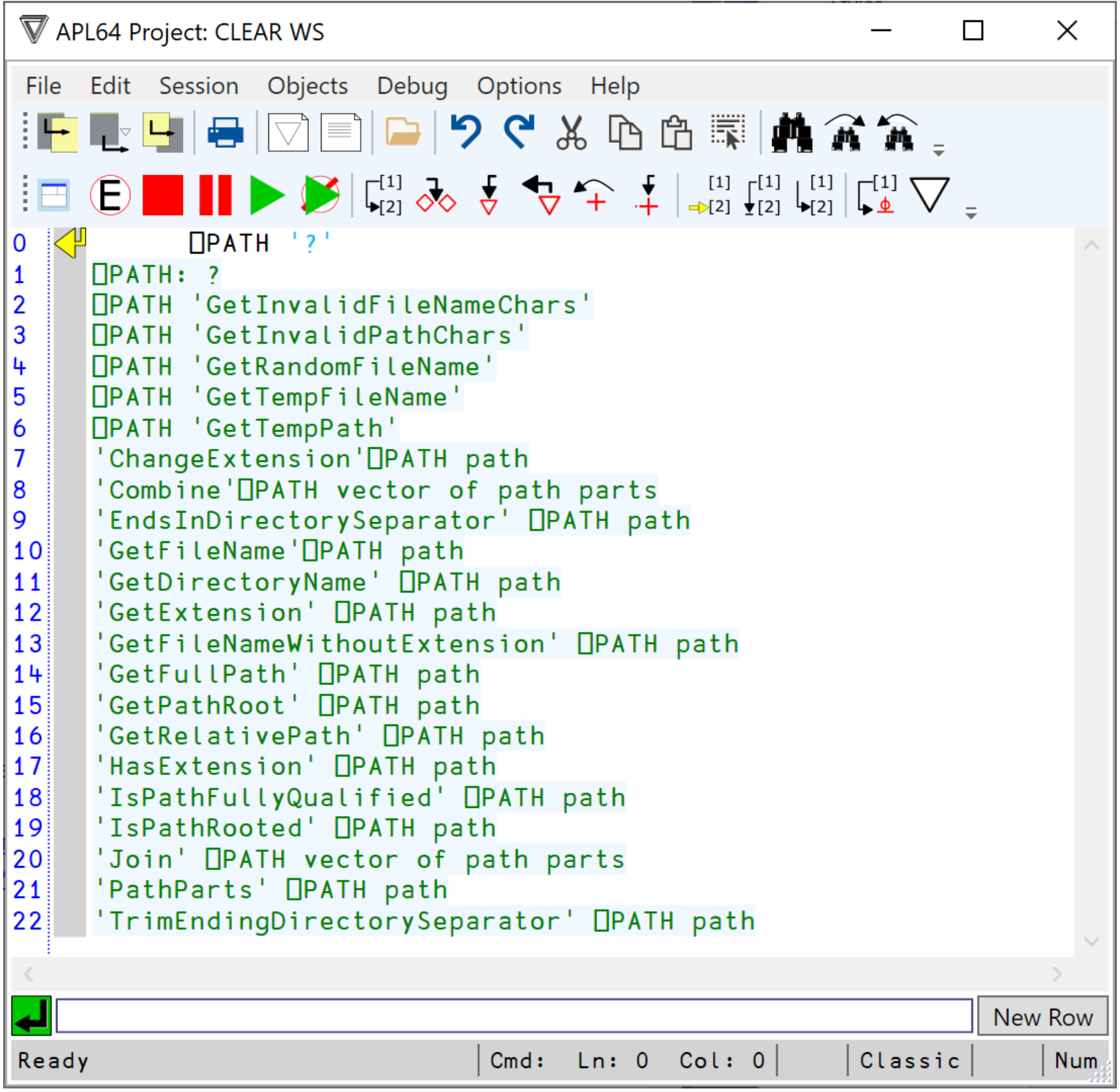
Color Printing Supported

Printing is supported with or without syntax colors in the APL 64 Project. This feature is new to the APL64 Project and is not available in APL+Win.

New System Functions Exclusive to the APL64 Project

- ⎕ACBD (GET System.AppContext.BaseDirectory)
- ⎕DTB, ⎕DLB, ⎕DLTB, ⎕DEB
- ⎕OVER
- ⎕UCASE, ⎕LCASE
- ⎕TEXTREPL
- ⎕LJUST, ⎕RJUST
- ⎕WHERE
- ⎕NEXTO
- ⎕MATRIFY
- ⎕SSTOMAT, ⎕MATTOSS, ⎕SSASSIGN, ⎕SSDROP, ⎕SSCOMPRESS
- ⎕SSCAT, ⎕SSDEB, ⎕SSDLB, ⎕SSDLTB, ⎕SSDTB
- ⎕SSFIND, ⎕SSINDEX, ⎕SSLEN, ⎕SSSHAPE, ⎕SSTAKE, ⎕SSUNIQUE
- ⎕DIV
- ⎕TRANSLATE
- ⎕WORDREPL
- ⎕NBLENGTH
- ⎕DTBR

- ⎕ROWFIND
- ⎕A (char vector 'ABCDEFGHIJKLMNOPQRSTUVWZYZ')
- ⎕D (Char vector '0123456789')
- ⎕EDIT (APL64 analogue of ⎕wcall 'W_Edit')
- ⎕PATH (.Net System.IO.Path features)



```
      ⎕PATH '?'
⎕PATH: ?
⎕PATH 'GetInvalidFileNameChars'
⎕PATH 'GetInvalidPathChars'
⎕PATH 'GetRandomFileName'
⎕PATH 'GetTempFileName'
⎕PATH 'GetTempPath'
'ChangeExtension'⎕PATH path
'Combine'⎕PATH vector of path parts
'EndsInDirectorySeparator' ⎕PATH path
'GetFileName'⎕PATH path
'GetDirectoryName' ⎕PATH path
'GetExtension' ⎕PATH path
'GetFileNameWithoutExtension' ⎕PATH path
'GetFullPath' ⎕PATH path
'GetPathRoot' ⎕PATH path
'GetRelativePath' ⎕PATH path
'HasExtension' ⎕PATH path
'IsPathFullyQualified' ⎕PATH path
'IsPathRooted' ⎕PATH path
'Join' ⎕PATH vector of path parts
'PathParts' ⎕PATH path
'TrimEndingDirectorySeparator' ⎕PATH path
```

APL2000 will continue to provide updates on the APL64 Project progress.

Contact sales@apl2000.com or call 301-208-7150 with questions or comments.