Welcome to HP VEE

HEWLETT
PACKARD

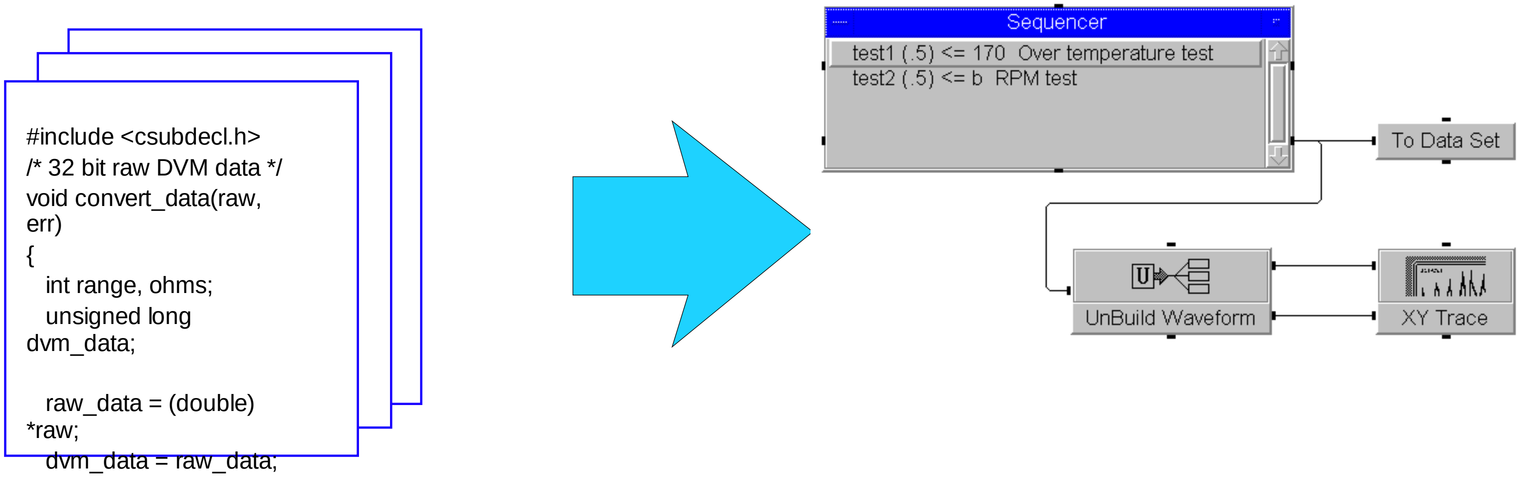
HP VEE
...Objective and Agenda

- Objective - Learn to use HP VEE and to meet your Test & Measurement Programming Challenges

- Agenda
  - Fundamentals
  - Objects
  - Functions
  - Operator Interface
  - Instruments
  - Records and DataSets
  - Sequencer

What is HP VEE?

HP VEE (Visual Engineering Environment) is a next generation "Graphical Programming Language" for developing and running test programs.



```
#include <csubdecl.h>
/* 32 bit raw DVM data */
void convert_data(raw,
err)
{
  int range, ohms;
  unsigned long
dvm_data;

  raw_data = (double)
*raw;
  dvm_data = raw_data;
```

Sequencer

test1 (.5) <= 170 Over temperature test

test2 (.5) <= b RPM test

Why HP VEE?

- Designed for Test
- Ease of Use
- Optimized for System Performance
- Open Systems

HP VEE is the only GPL designed for Test & Measurement solutions

Focus of HP’s Test System Strategy

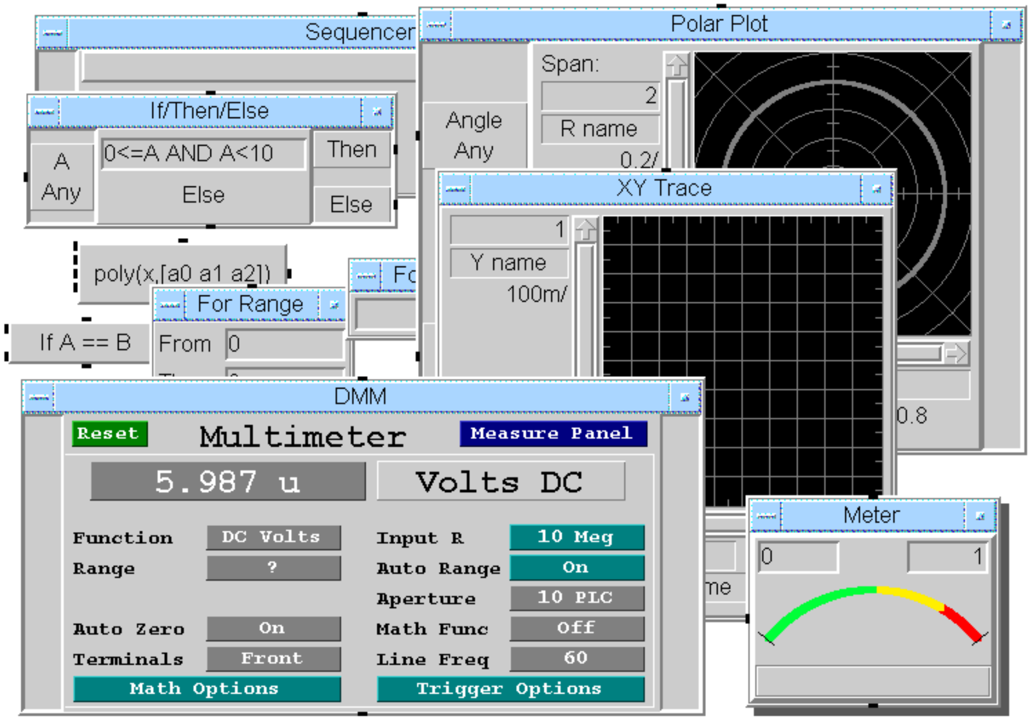
- Offer scaleable price/performance in products and services
- Make system integration easy
- Embrace open measurement, computer, and software standards
- Offer a broad range of products and services from system components to custom services
- Support multi-vendor system environments

HP VEE

How does HP VEE Work?

Objects

- It provides hundreds of high-level objects that perform most test system functions
  - I/O
  - Analysis
  - Display
  - Test Sequencing
  - Flow Control



5.987 u

Volts DC

```
Function     DC Volts      Input R      10 Meg
Range        ?             Auto Range   On
                           Aperture     10 PLC
Auto Zero    On            Math Func    Off
Terminals    Front         Line Freq    60
```

HEWLETT
PACKARD

The Approach for Learning HP VEE

- Hands-on is the best way to learn
- Don’t make it too hard
- Learn the objects

VEE Operation Fundamentals

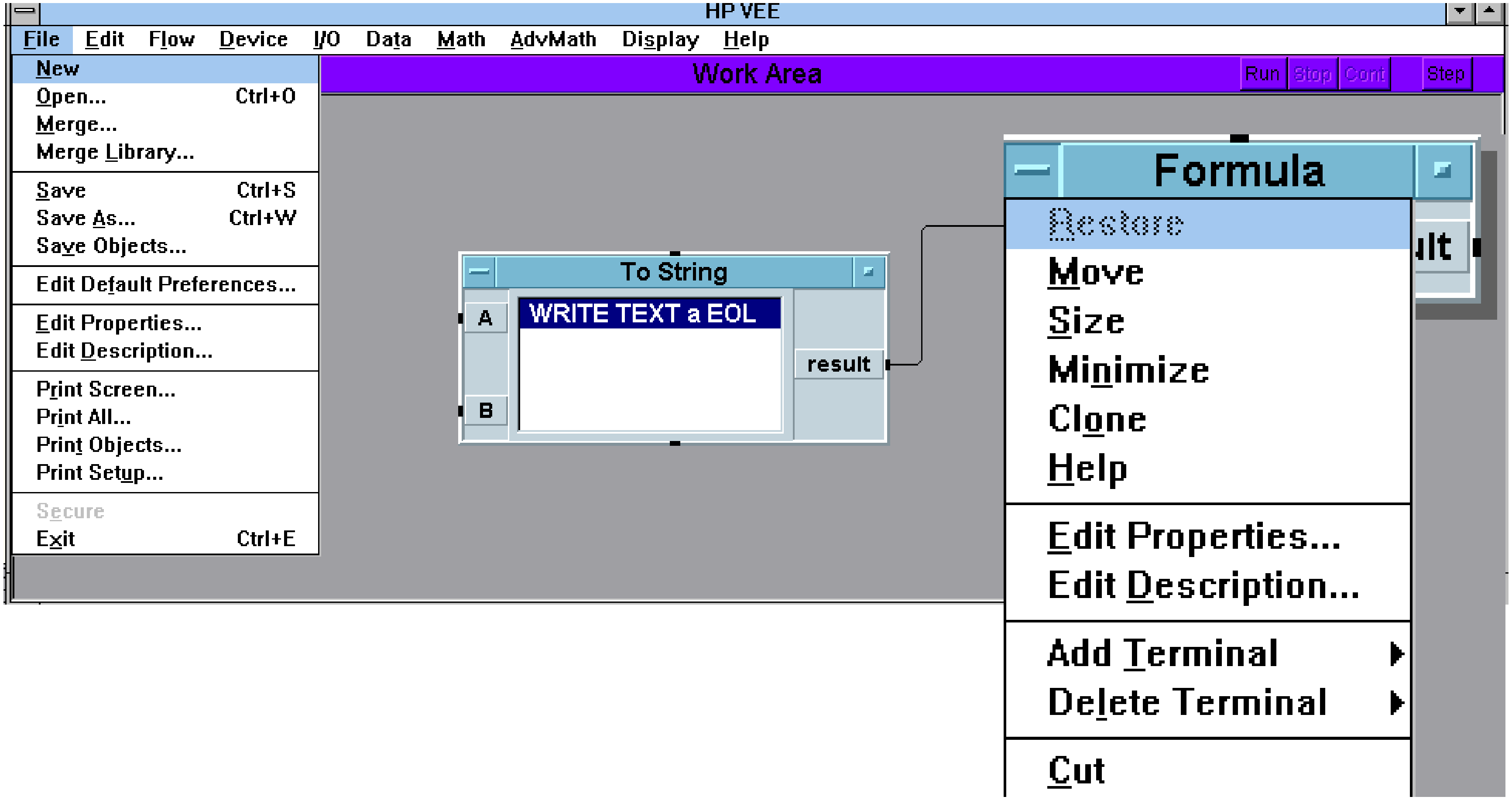
- Synchronous Operation
- Propagation Rules
- Multiple Threads

Useful Definitions For HP VEE

- Program the finished solutions with objects linked together
- Work Area "the executable block diagram" The area within the HP VEE window in which you build programs. The "Detail" View
- Object any item placed on the work area
- Icon View a small, graphical representation of an object
- Open View the maximized view of an object
- Panel View the operator interface

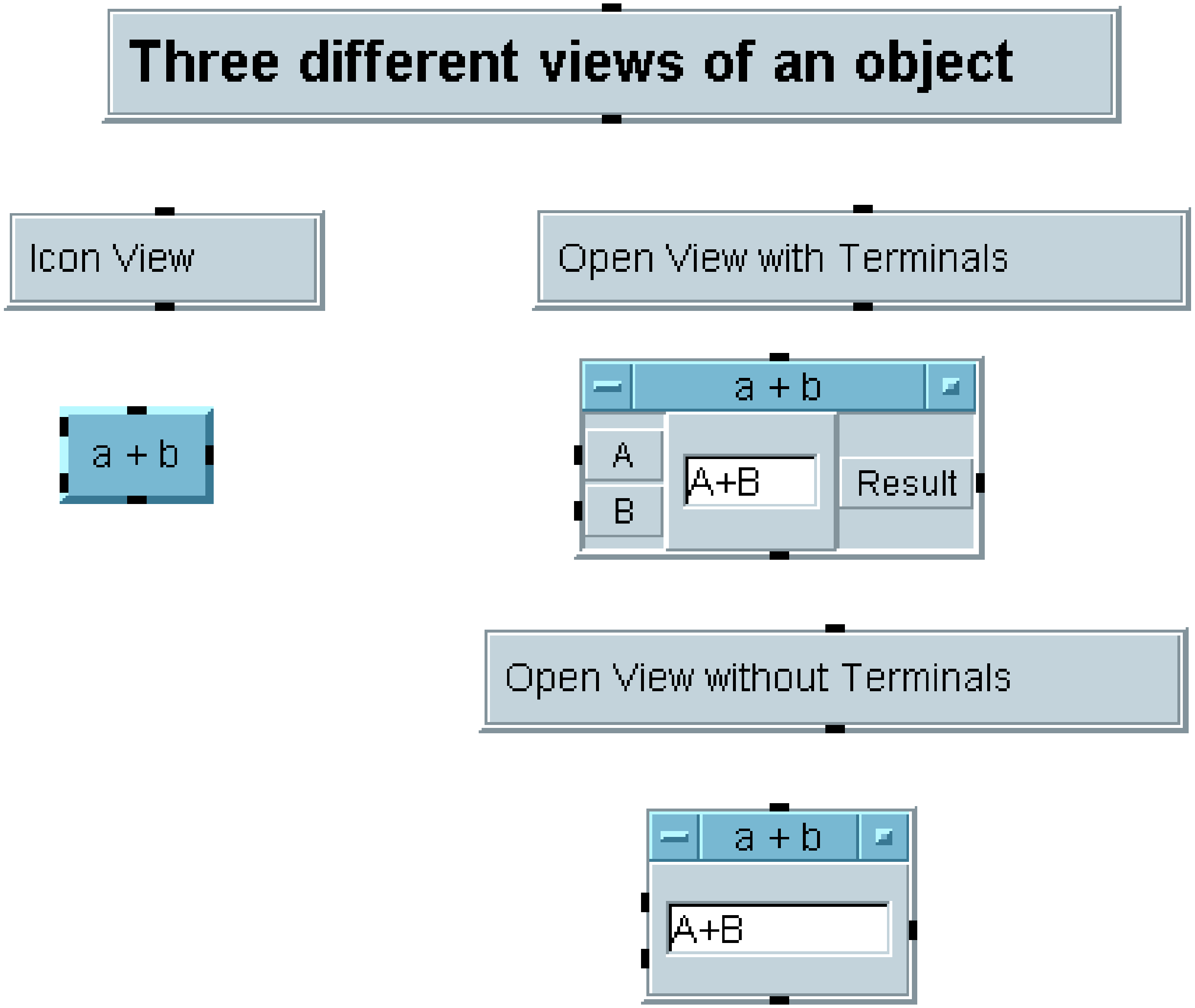
HP VEE Work Area

HP VEE



WRITE TEXT a EOL

Icon vs. Open View



Three different views of an object

Icon View

Open View with Terminals

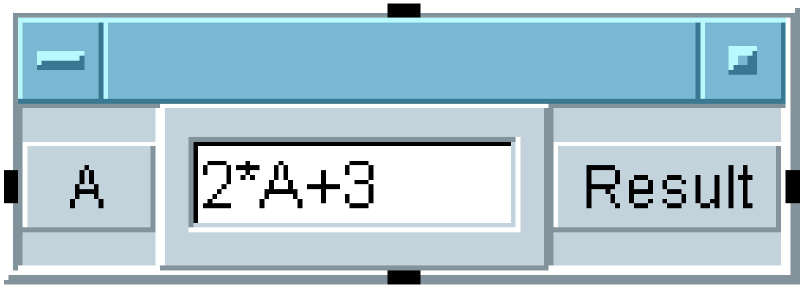
Open View without Terminals

More Useful Definitions For HP VEE

- Thread a set of objects connected by solid lines
- Operate to execute an object
- Ping send data or sequence instructions across a line to a terminal
- Container the package that is transmitted over lines and is processed by objects. The Container can be any of the HP VEE data types

Pins and Terminals

- Data Input/Output
- Sequence Input/Output
- Asynchronous Control Input
- Error Output



Pin

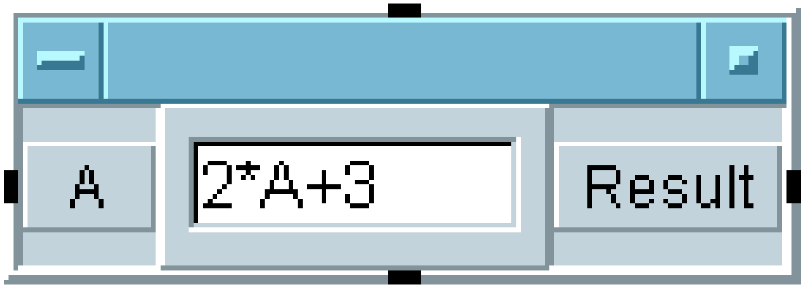
Terminal

HP VEE

Synchronous Operation

Sequence In

Data Input



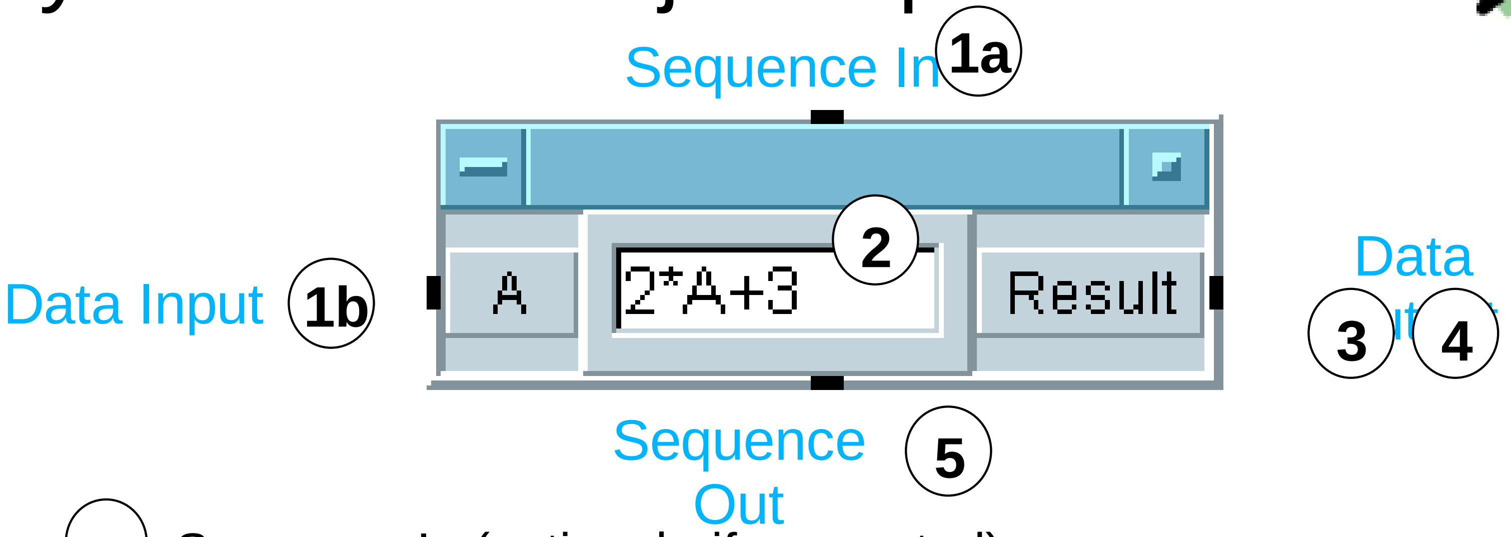
Data Output

Sequence Out

- ➤ Data flows left-to-right
- ➤ Sequence flows top-to-bottom

- All Data Input pins must be connected for an object to fire
- All Data Input pins must be pinged before the object will operate
- A single Data Input pin cannot accept more than one line

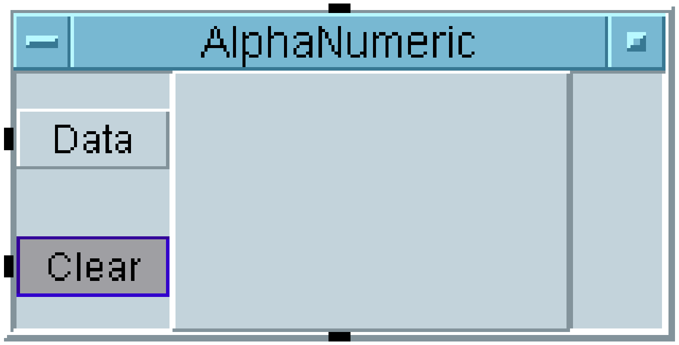
Synchronous Object Operation



- 1a Sequence In (optional - if connected)
- 1b Data in is accepted
- 2 Object operates
- 3 Data out is sent
- 4 Object waits for all data out to be sent and for "receipt acknowledged"
- 5 Sequence out fires
- 6 Object deactivates

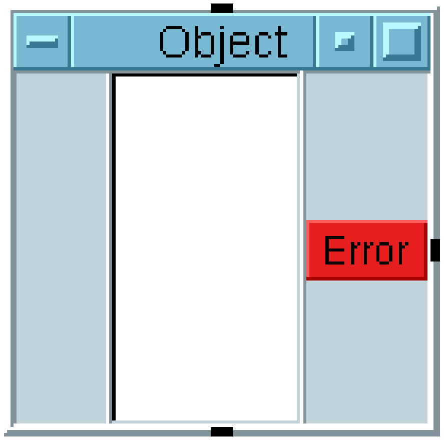
Optional Object Connections

- Data In, Data Out
  - Many objects allow additional data in, data out terminals



- Control Input
  - Ping causes immediate execution of object sub-function
  - Is not required for overall object execution
  - Examples: (Clear, Autoscale X, etc.)

- Error Output
  - Overrides standard object behavior
  - Activates when error occurs during object execution
  - Activates INSTEAD OF data outputs
  - Allows HP VEE to continue execution after error



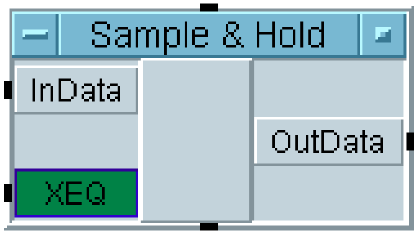
Adding Optional Inputs

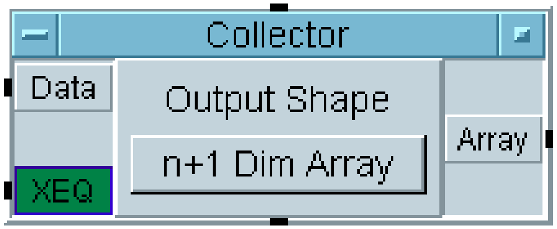
- Object menu provides ability to add terminals to objects
  - Data and Control
  - Inputs and Outputs

- Terminal can be opened to Edit Name (double-click)
  - Type and shape can be modified if required

XEQ Control Pin

- XEQ control causes immediate object operation
  - Available data used
- Required by some data building objects
- Useful for continuing after error
- Objects with XEQ pins
  - UserObject
  - Confirm (OK)
  - Set Values
  - Collector
  - Call
  - Sample/Hold





Propagation Rules

- Pre-Run & Activation, Auto Execute, Wait for Input
- Order of Execution
- Parallel threads

Propagation Definitions

- PreRun
  - Checks for "static" structure of program
  - Feedback loops, connected inputs
  - Occurs for entire model when RUN pressed
  - Occurs for single thread if START object pressed
  - Objects reset to initial conditions
  - Files rewound
  - Errors cleared

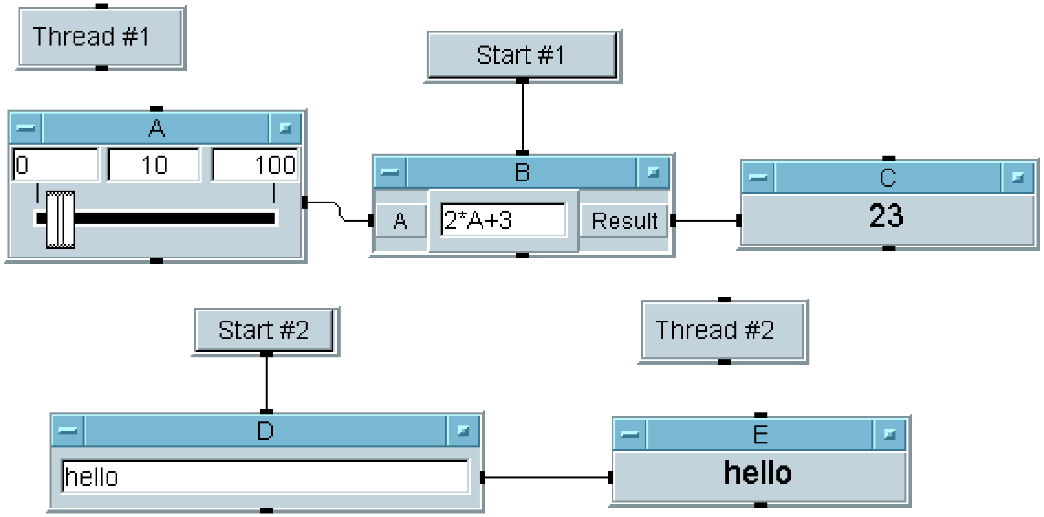
- Activate
  - Analogous to PreRun, but for individual UserObject

- Auto Execute
  - Propagation initiates at Data object, after user input

- Wait for Input
  - Running program pauses until user input

Start Objects

- Useful in Debugging
- Allow execution sequence to begin
- Affect only their own thread
- At Run time, all START objects on every thread operate prior to any other objects



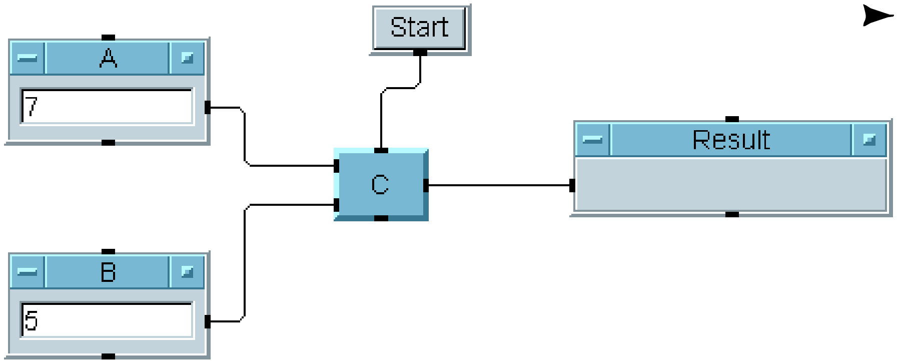
➤ Initiates thread propagation

➤ Pressing Start #1 does not affect Thread #2

HP VEE

Propagation

- **Unconstrained Objects** (A & B)
  - –No input constraints (Data In or Sequence In)
  - –Control inputs do not constrain an object
- **Constrained Objects** (C)
  - –Have either Data Input or Sequence Input pins connected



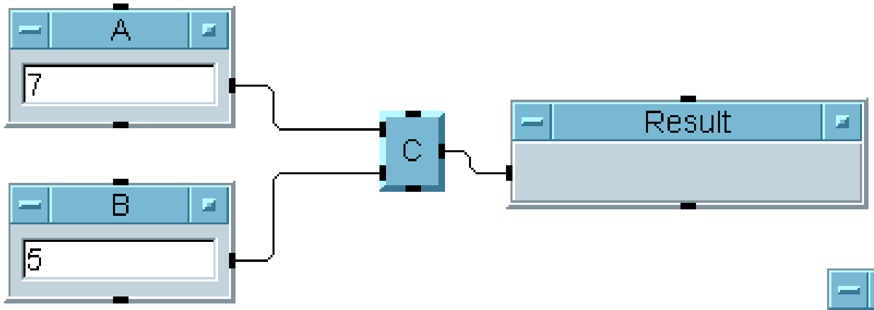
➢A or B may operate anytime after Start

HEWLETT PACKARD

HP VEE

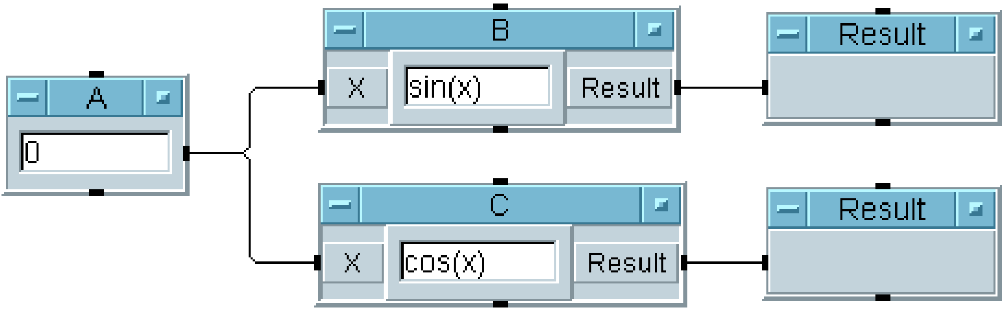
Propagation Rules

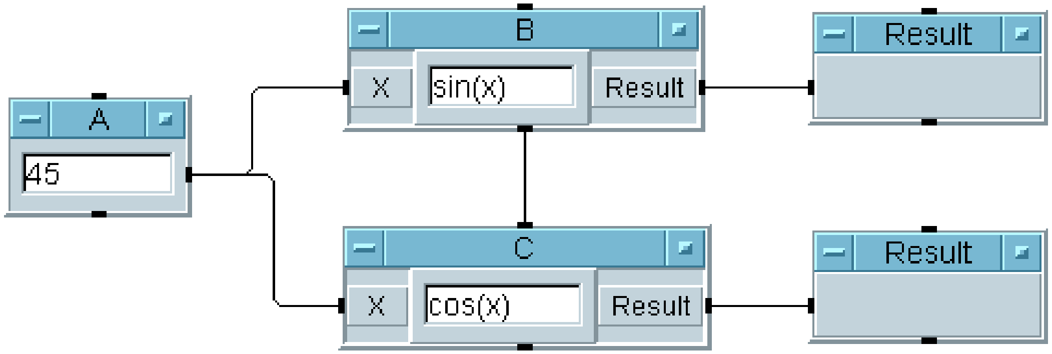
- Unconstrained objects may operate in any order



- ➤C must wait for both A & B
- ➤A or B may operate first

- ➤Both B & C must wait for A, after which
- ➤B and C will operate in an unknown order

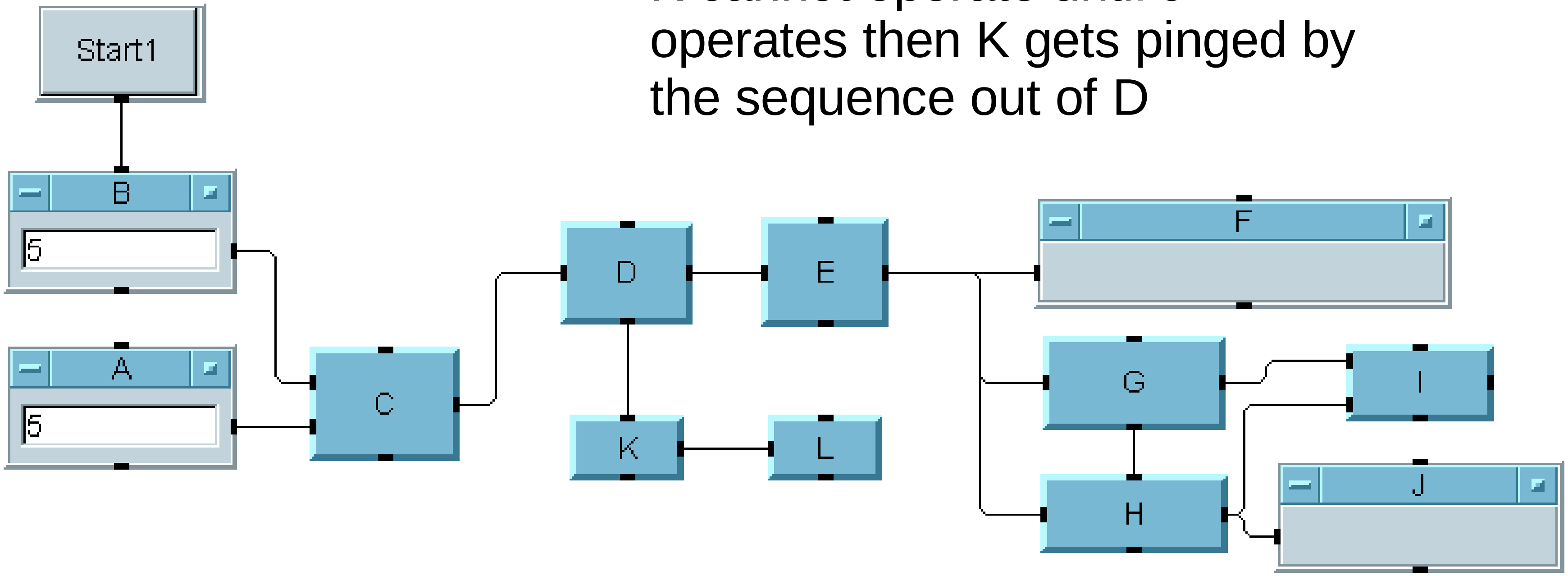




- ➤C must wait for B to finish. Not vice versa

Propagation Example

- ➤A operates first after Start
- ➤K cannot operate until J operates then K gets pinged by the sequence out of D

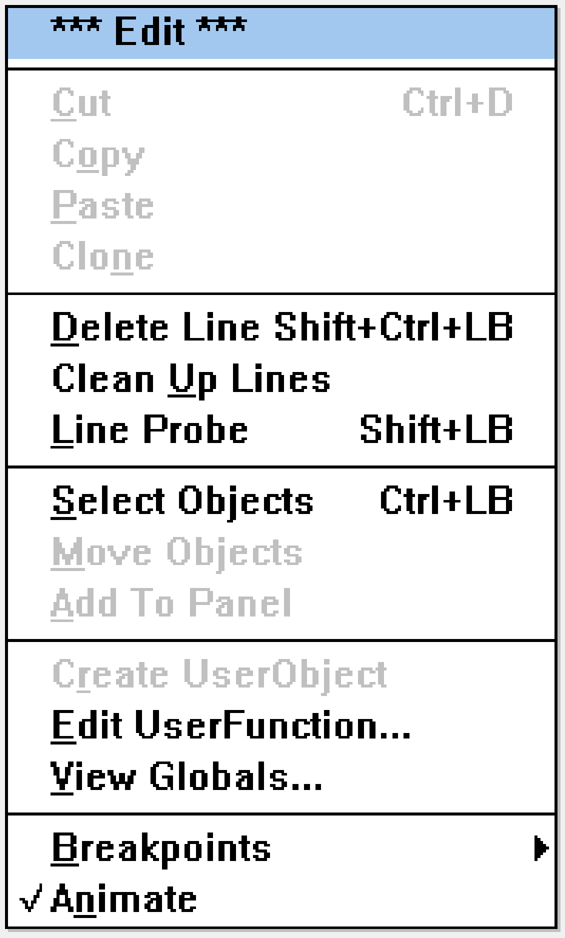


Multiple Thread Propagation

- Parallel threads are time-sliced by "propagation engine"
- Timeslice = 1 primitive object
- Note:
  - Each object on an iterating subthread of a repeat device (iterator) counts as one timeslice
  - UserObjects are MULTIPLE objects. Each object in a UserObject is a primitive object
  - UserFunctions propagate to completion. Not TimeSliced

Debugging

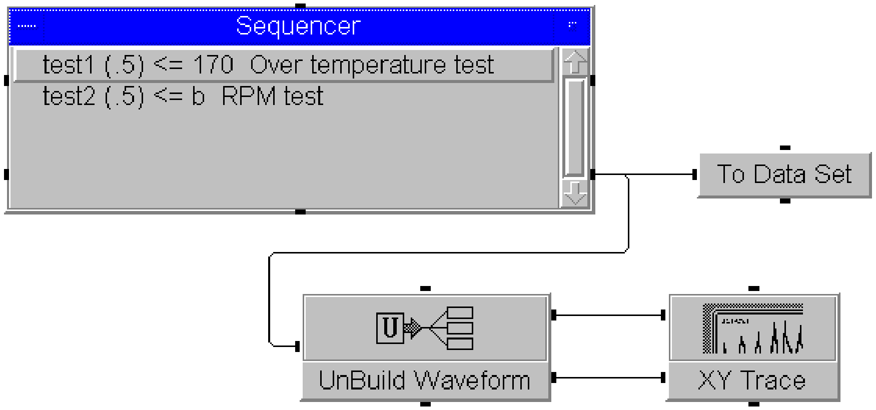
- Show Execution Flow
  - Highlights each object during operation
- Show Data Flow
  - Shows data container moving along threa
- Set Breakpoints
  - Pauses execution at this point
- Line Probe <SHIFT LB on Line>
  - Shows data container on thread



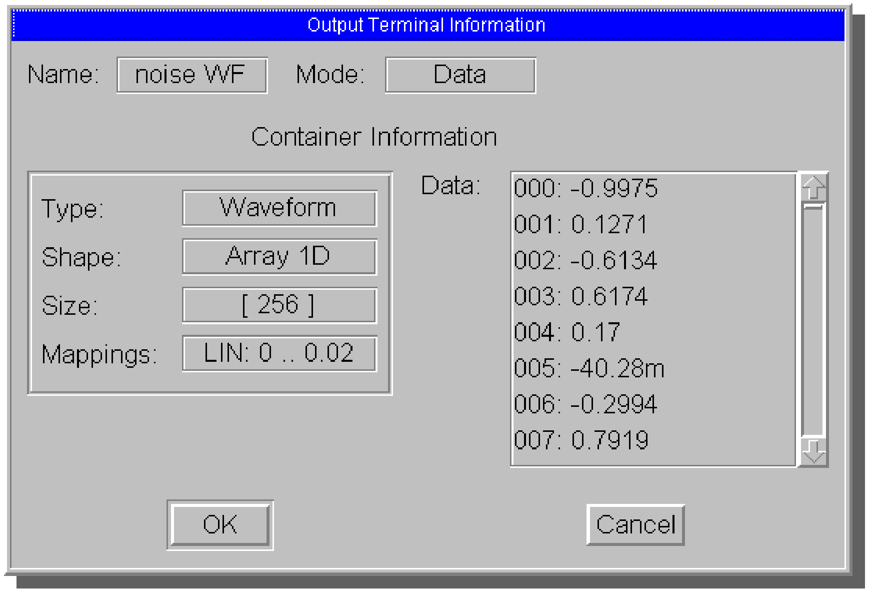
Debugging

HP VEE

- Debugging features are provided to quickly get systems up and running
  - Step
  - Breakpoints
  - Animate
    - Show Execution Flow
    - Show Data Flow
  - Line Probe
  - Bus Monitor



test1 (.5) <= 170 Over temperature test
test2 (.5) <= b RPM test



Output Terminal Information

Name: noise WF   Mode: Data

Container Information

Type: Waveform
Shape: Array 1D
Size: [256]
Mappings: LIN: 0 .. 0.02

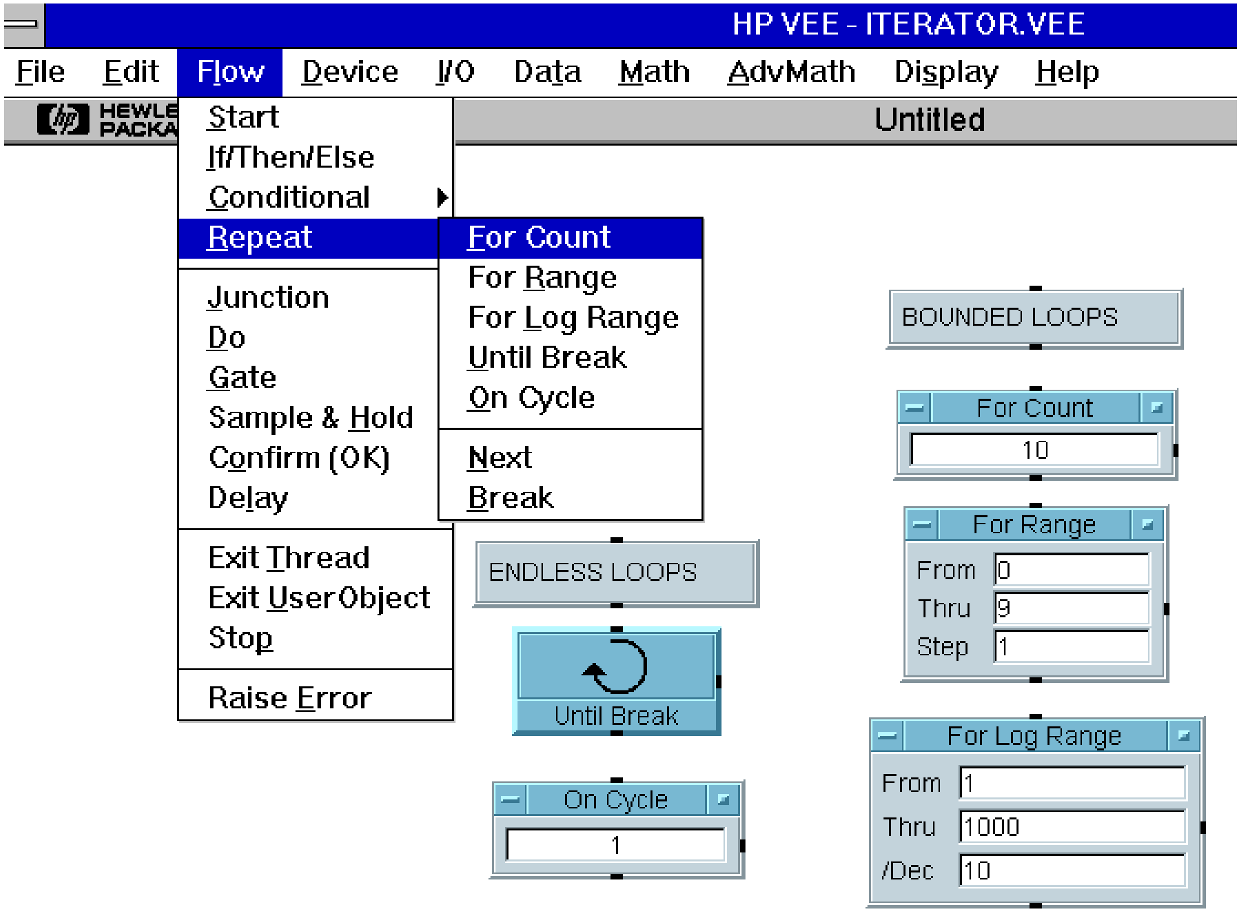
Data:
000: -0.9975
001: 0.1271
002: -0.6134
003: 0.6174
004: 0.17
005: -40.28m
006: -0.2994
007: 0.7919

Repeat (Iterators)

- Repeatedly propagate data onto a subthread

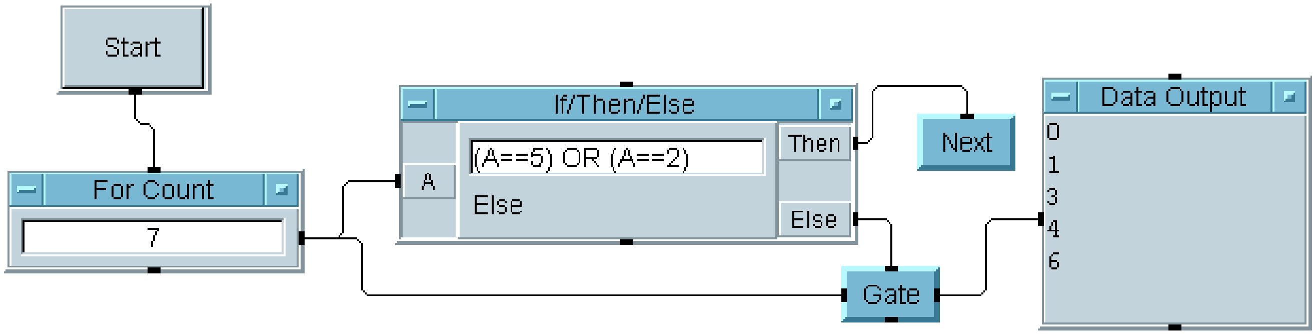
- Bounded Loop
  - For Count
  - For Range
  - For Log Range

- Endless Loop
  - Until Break
  - On Cycle



Early Loop Termination

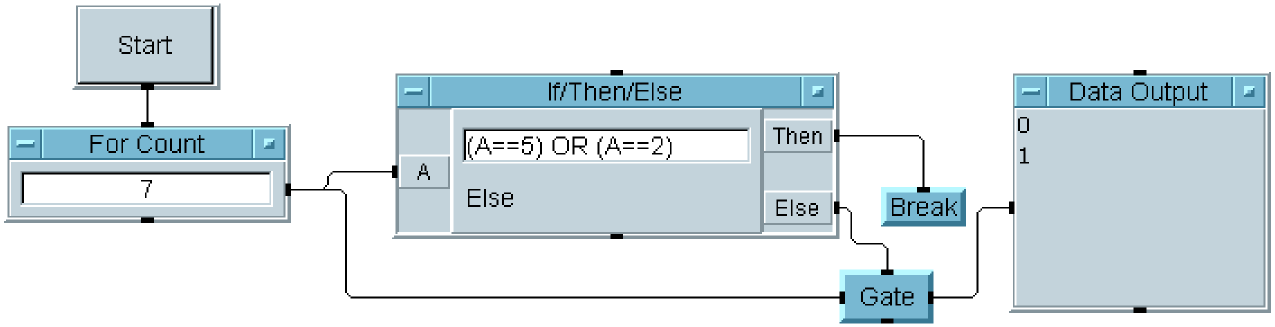
- Next - terminates propagation of current iteration



```
0
1
3
4
6
```

- ▸ A executes repeatedly

- Break - terminates current and future iterations



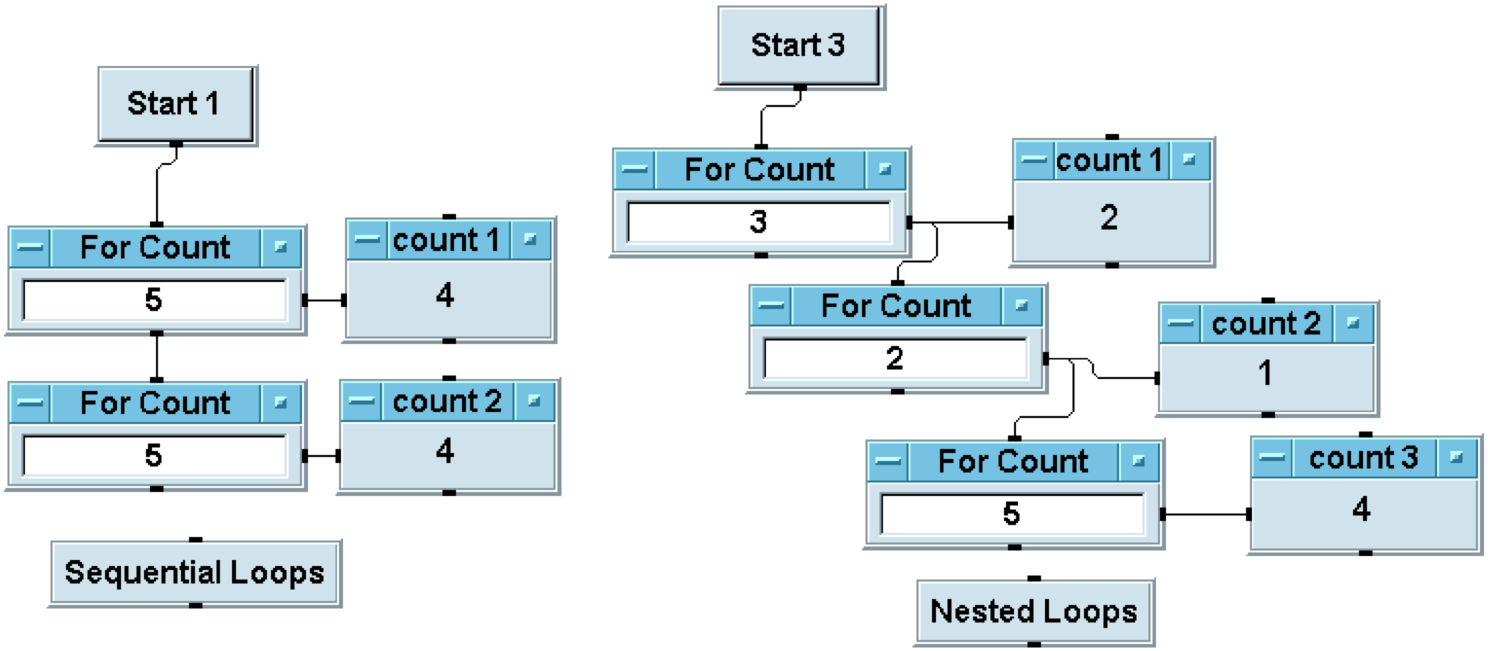
```
0
1
```

- ▸ A executes repeatedly

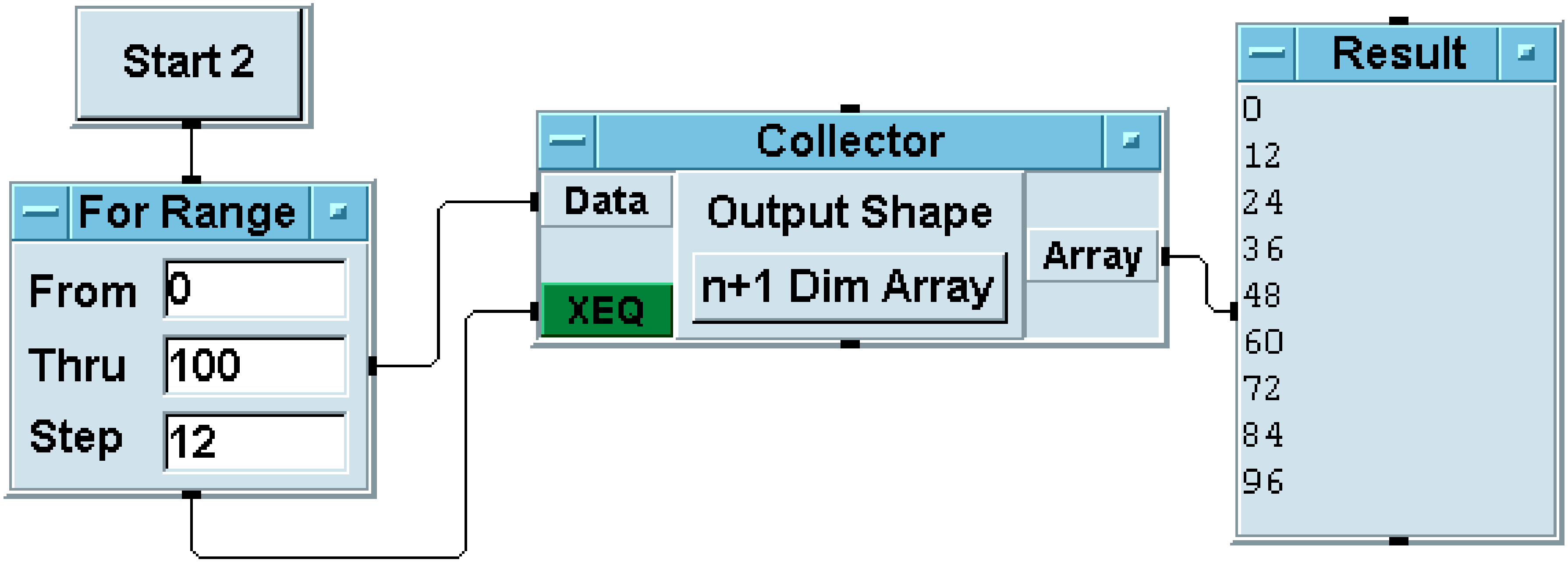
Thread Structures

- Subthread basic unit of work
  - Applicable to functionality of UserObject

- Structures of subthreads



Collecting Interative Data

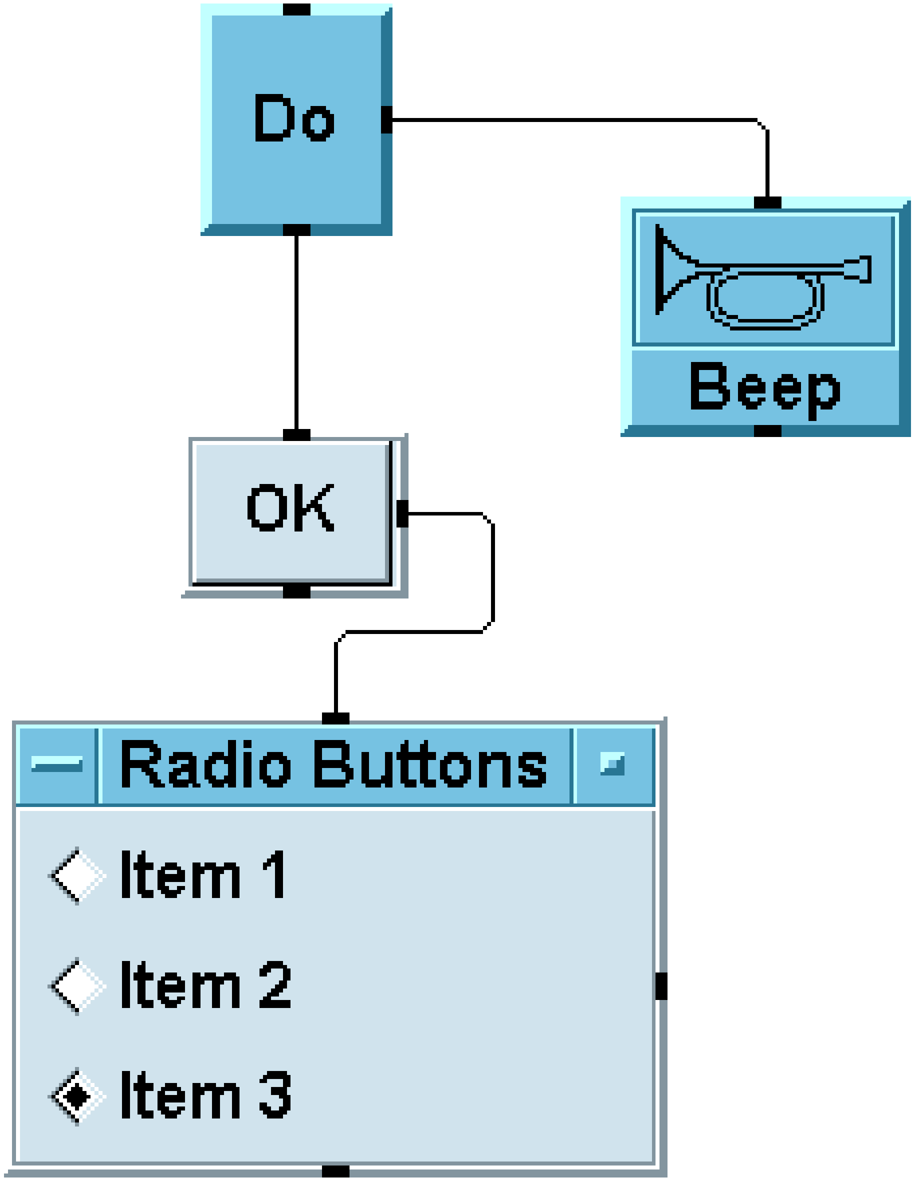


From 0
Thru 100
Step 12

Output Shape
n+1 Dim Array

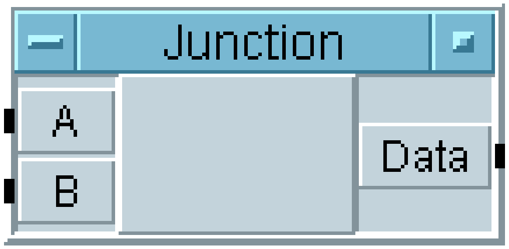
```
0
12
24
36
48
60
72
84
96
```

Confirm (OK) and Do

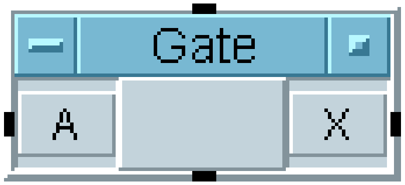


Flow (Data)

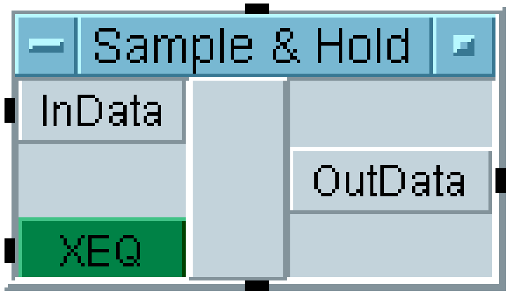
- Junction
  - Wired-OR which sends its most recent input data
  - Often used to send 2 or more data lines to the same input pin
  - Extra inputs are added as DATA inputs
  - Only data object with asynchronous inputs



- Gate
  - Similar to a "latch"
  - Holds input data until Sequence In is pinged (No sequence in connection on Gate --- data passes through)



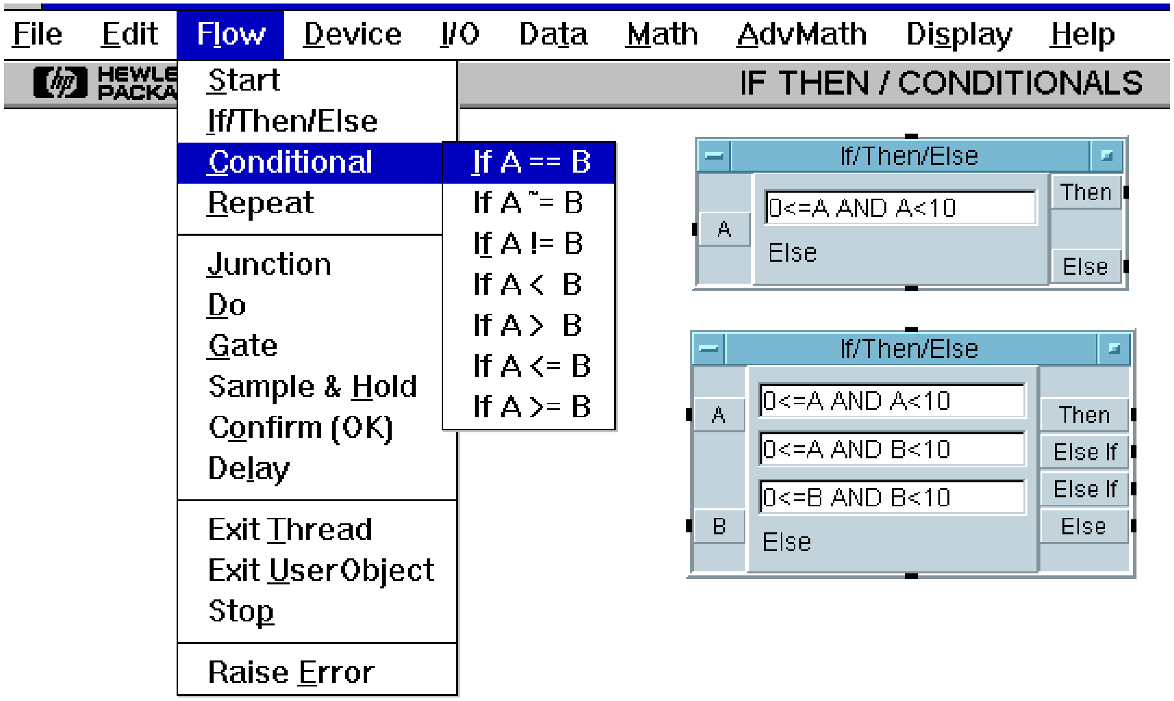
- Sample & Hold
  - Sends data every time XEQ pin is pinged



Conditional Branching

- If/Then/Else
  - Allows testing according to user formula
  - Allows many inputs
  - Allows Else/If and Else outputs to give the capability for multi-conditionals (case arguments)

- Conditionals
  - Pre-formulated two-way comparisons



0<=A AND A<10

0<=A AND A<10

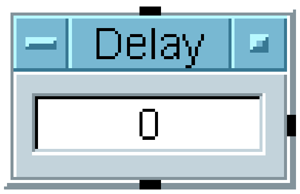
0<=A AND B<10

0<=B AND B<10

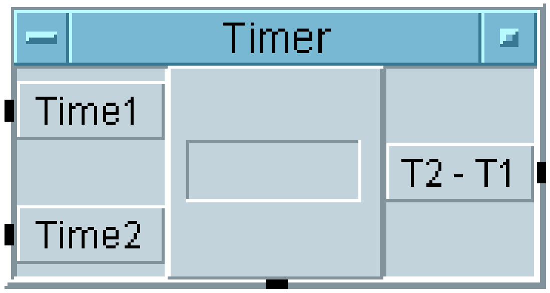
HP VEE

Time Related Objects

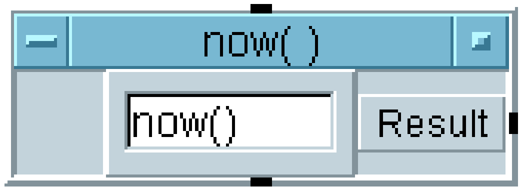
- Delay
  - Delays propagation for n seconds



- Timer
  - Measures execution time between two objects

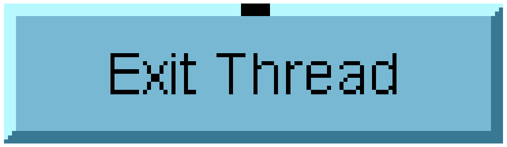


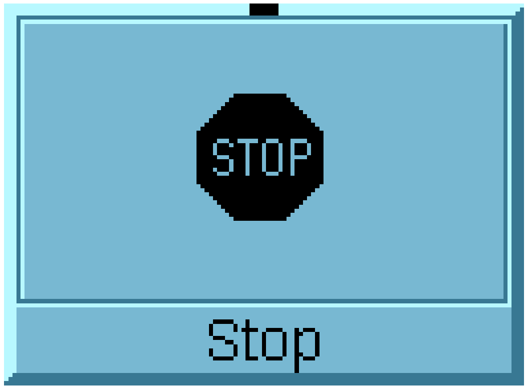
- Now
  - Indicates time of execution



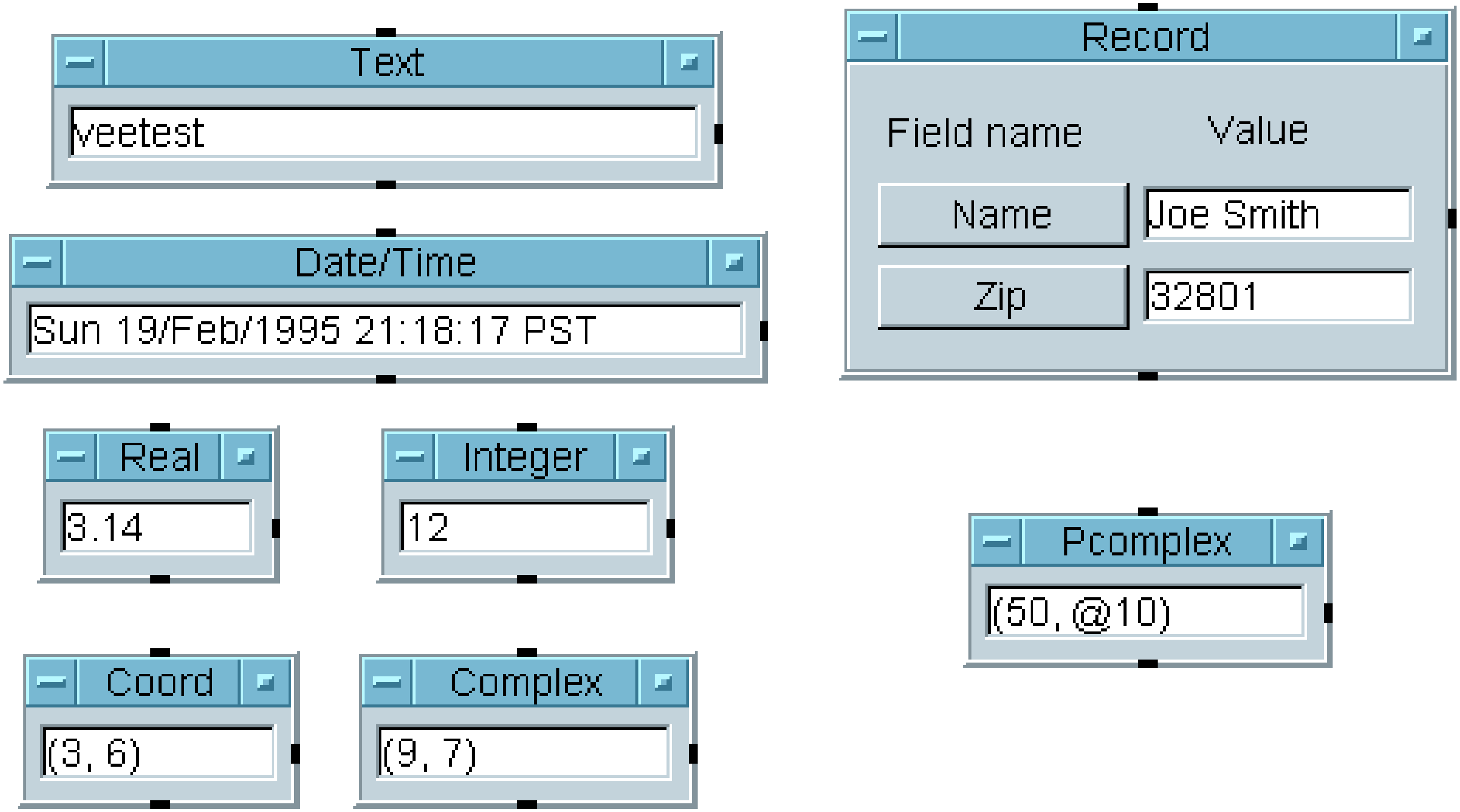
Termination (Exits)

- Exit Thread
  - Terminates propagation of an individual thread
- Stop
  - Terminates program
  - Equivalent to pressing stop button





HP VEE Data Types



Text

veetest

Date/Time

Sun 19/Feb/1995 21:18:17 PST

Real

3.14

Integer

12

Coord

(3, 6)

Complex

(9, 7)

Record

Field name	Value
Name	Joe Smith
Zip	32801

Pcomplex

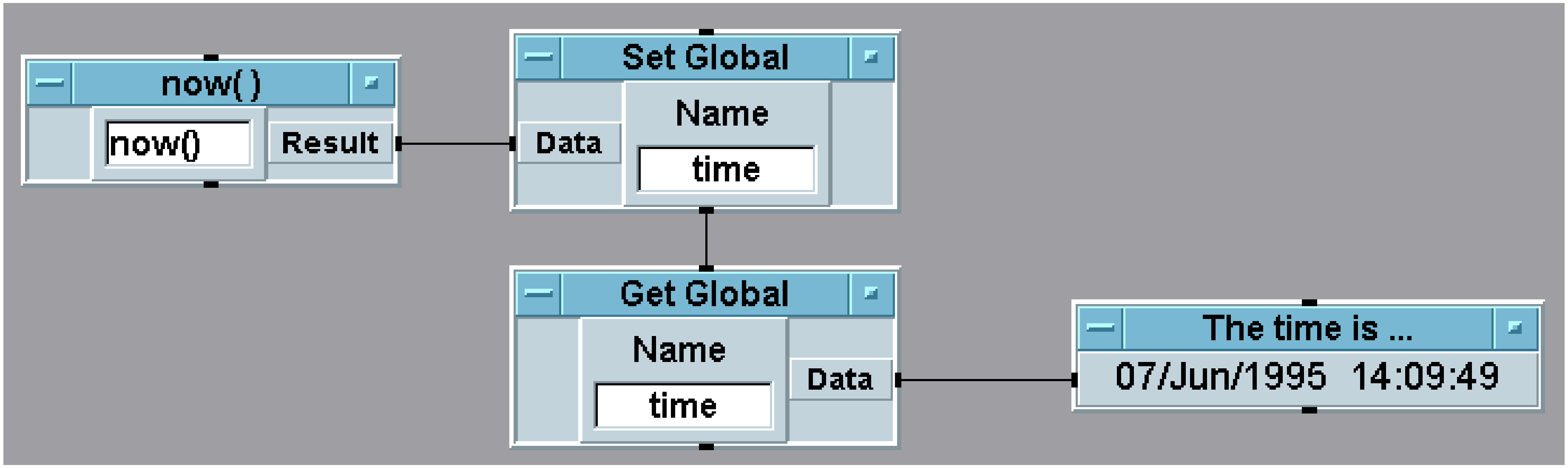
(50, @10)

HEWLETT PACKARD

Global Variables

- Allows you to reference data that is maintained in a single place

- Can accept any VEE container



Setting/Getting Globals

- You can only SET a global with the
  Data ==> Globals ==> SET GLOBAL object

- You can Get a global 2 ways
  1. Data ==> Globals ==> GET GLOBAL object
  2. Any valid expression (field)

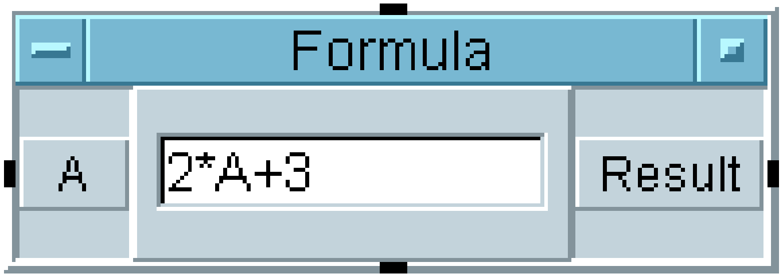
- Hierarchy:
  1. Local variables (terminal name)
  2. Global variables

Global - Debugging

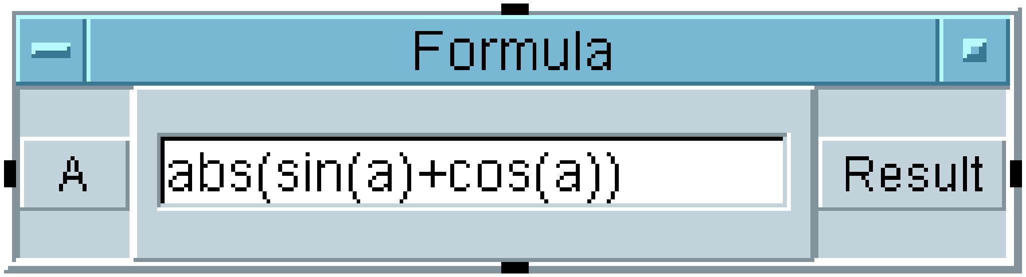
Common Error

- Globals are optionally erased when you press Run or Start
- So, you must SET before you GET
- Edit ==> View Globals is available after you SET the value

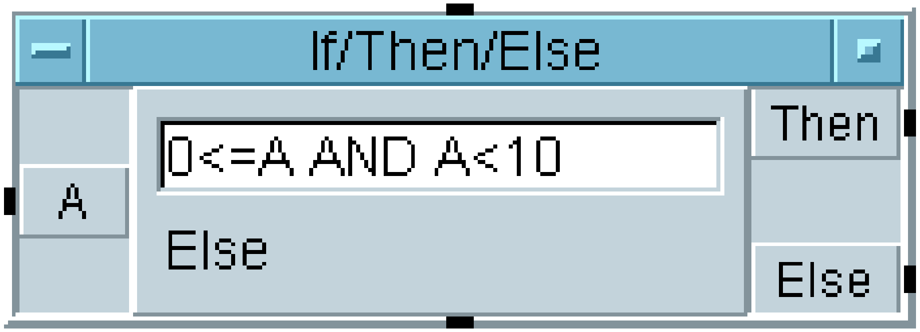
Formula and Expressions



2*A+3

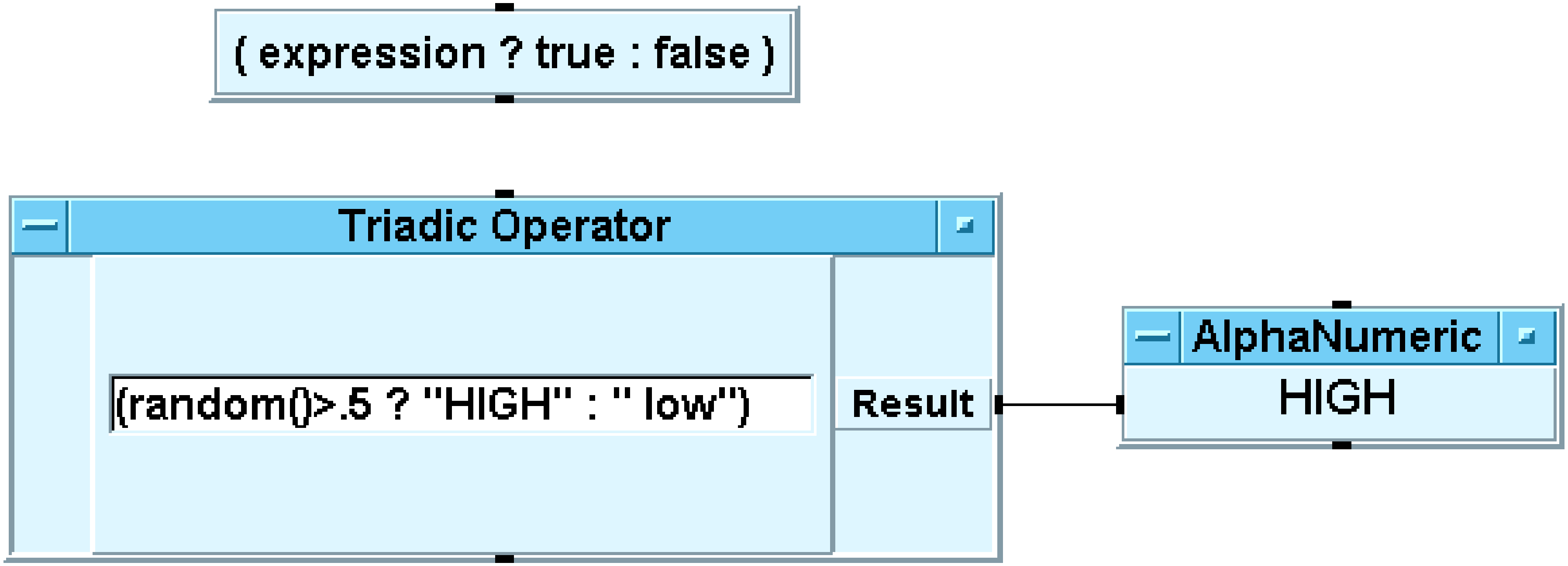


abs(sin(a)+cos(a))



0<=A AND A<10

Triadic Operator



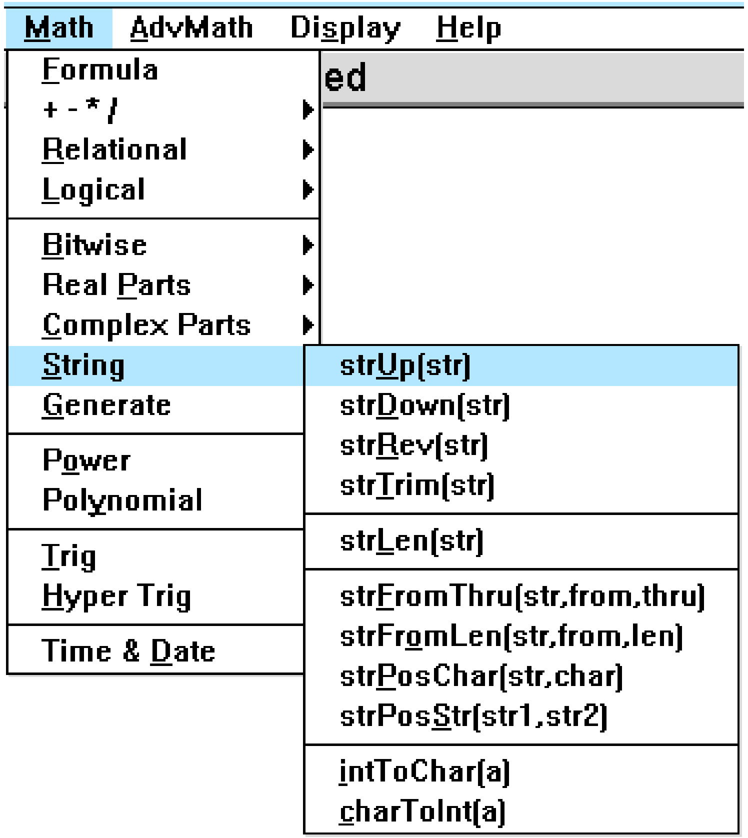
(expression ? true : false)

```
(random()>.5 ? "HIGH" : " low")
```

Math ==> String

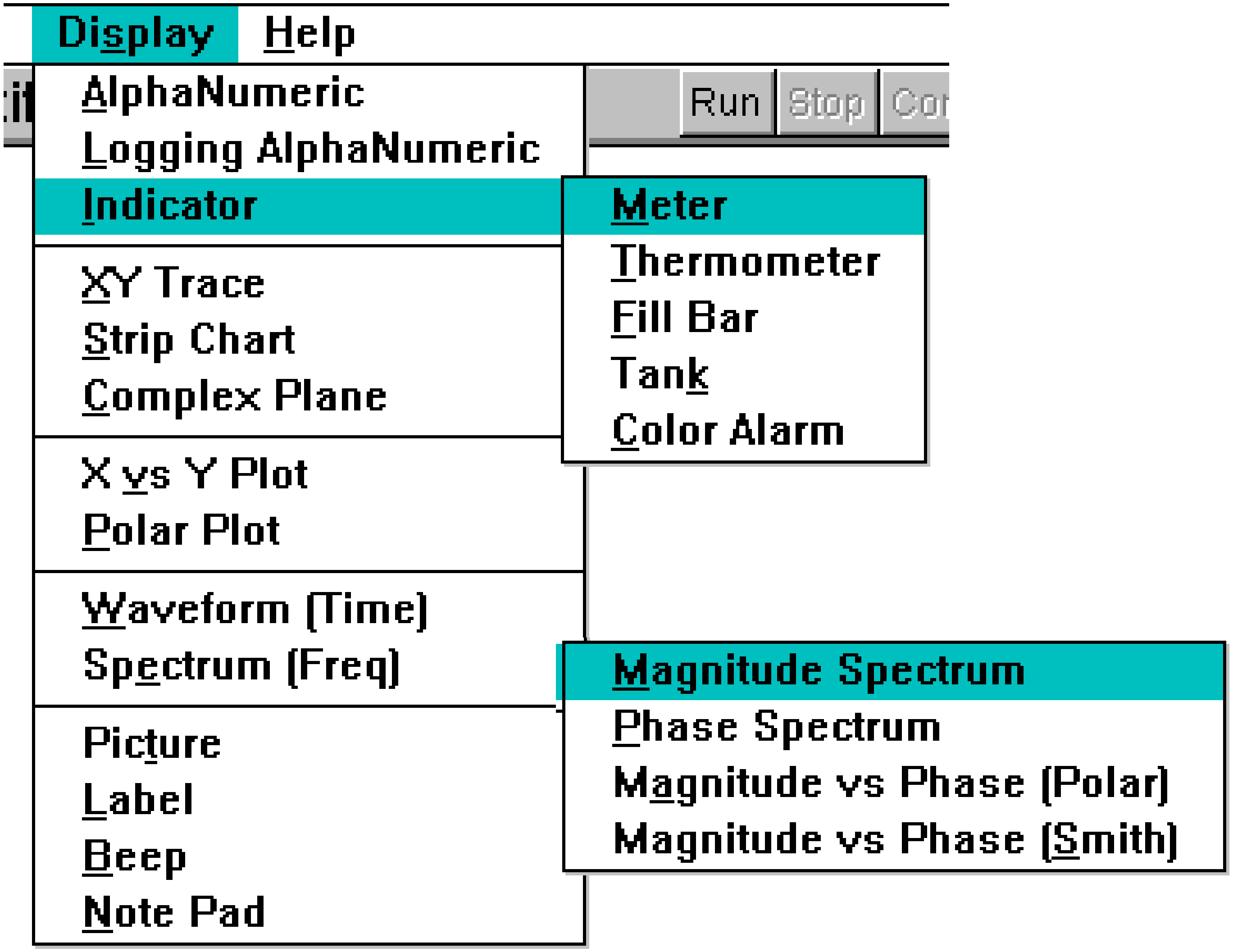
- Strings
  strUp ("footBar") = "FOOTBAR"
  strDown ("FootBAr") = "footbar"
  strRev ("footbar") = "raboof"
  strTrim ("   foot bar   ") = "foot bar"
  strLen ("footbar") = 6
  strFromThru ("footbar",0,2) = "foot"
  strFromLen ("footbar",0,2) = "fo"
  strPosChar ("footbar","bao") = 1
  strPosStr ("footbar","oba") = 2

  strPosChar ("footbar","m") = -1
  strTrim ("abfootbarabab","ba") =
  "footbar"



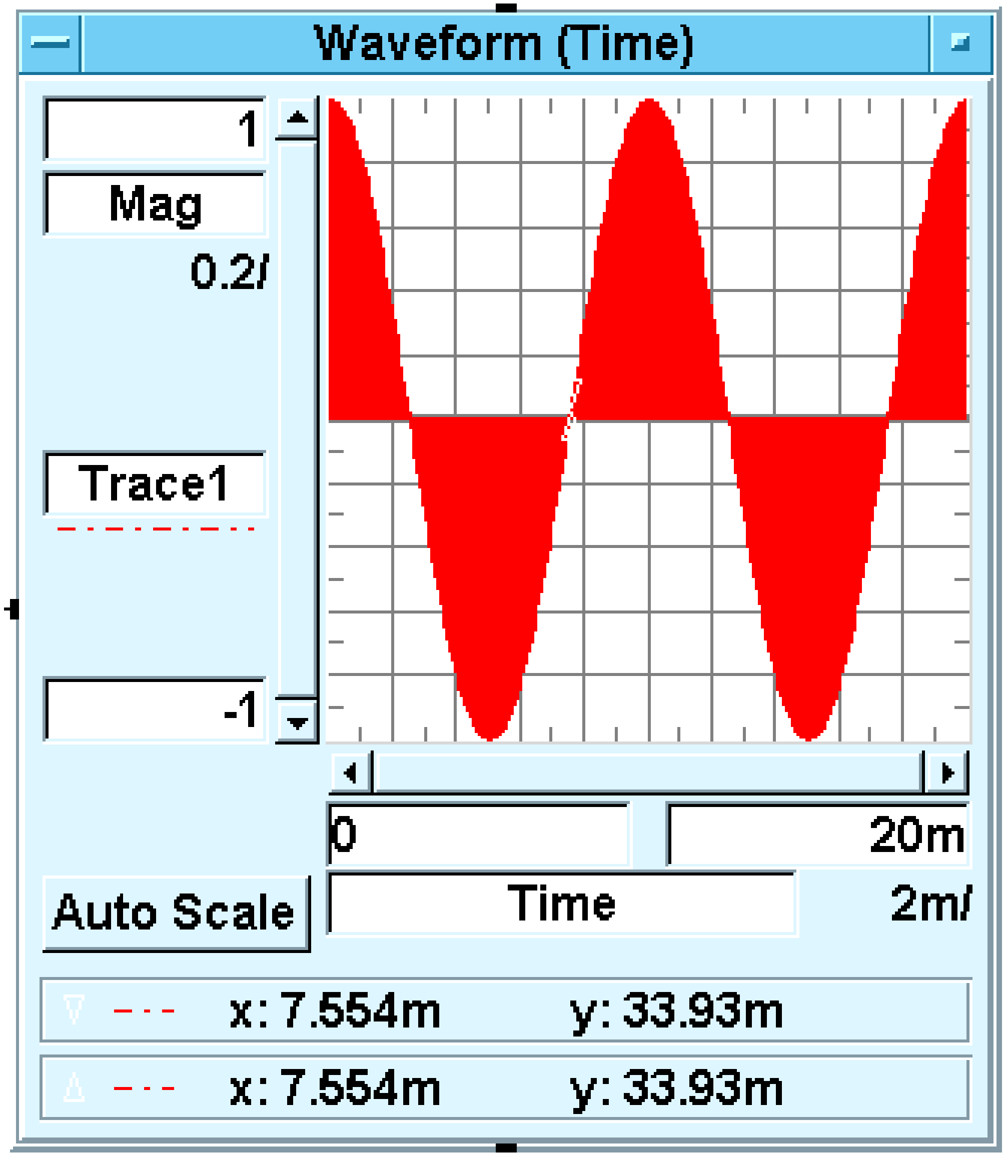
HP VEE

HP VEE Display Objects



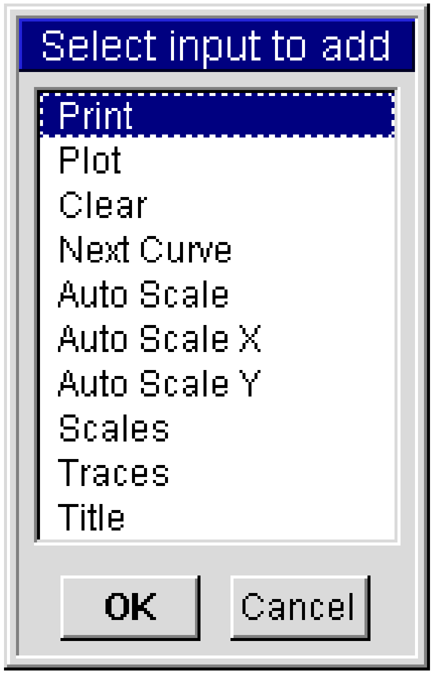
HEWLETT
PACKARD

Display Control Pins



x: 7.554m y: 33.93m

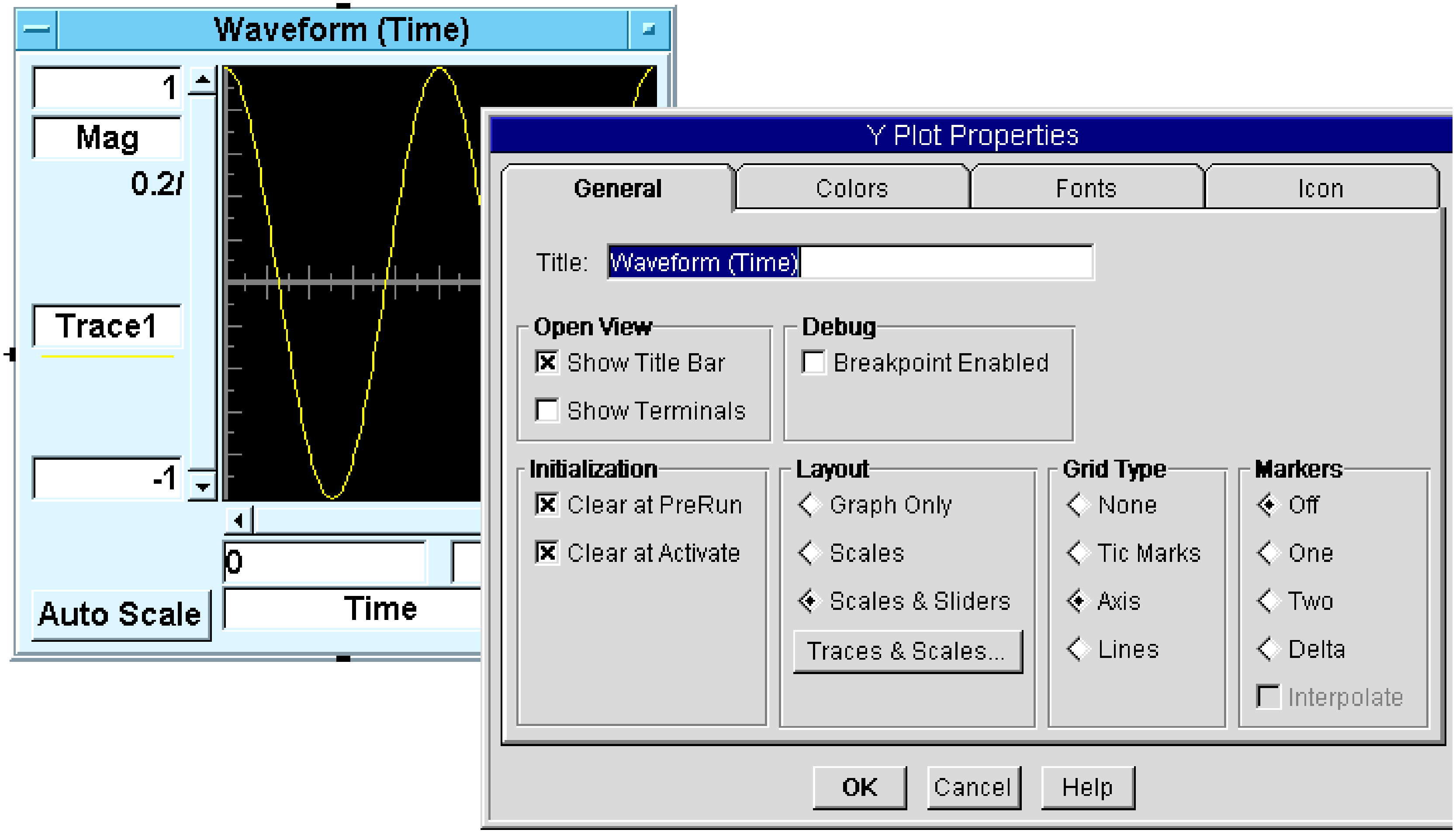
x: 7.554m y: 33.93m


Select input to add

- Print
- Plot
- Clear
- Next Curve
- Auto Scale
- Auto Scale X
- Auto Scale Y
- Scales
- Traces
- Title

OK Cancel

Display Properties



Y Plot Properties

General | Colors | Fonts | Icon

Title: Waveform (Time)

Open View
- ☒ Show Title Bar
- □ Show Terminals

Debug
- □ Breakpoint Enabled

Initialization
- ☒ Clear at PreRun
- ☒ Clear at Activate

Layout
- ◇ Graph Only
- ◇ Scales
- ◆ Scales & Sliders
- Traces & Scales...

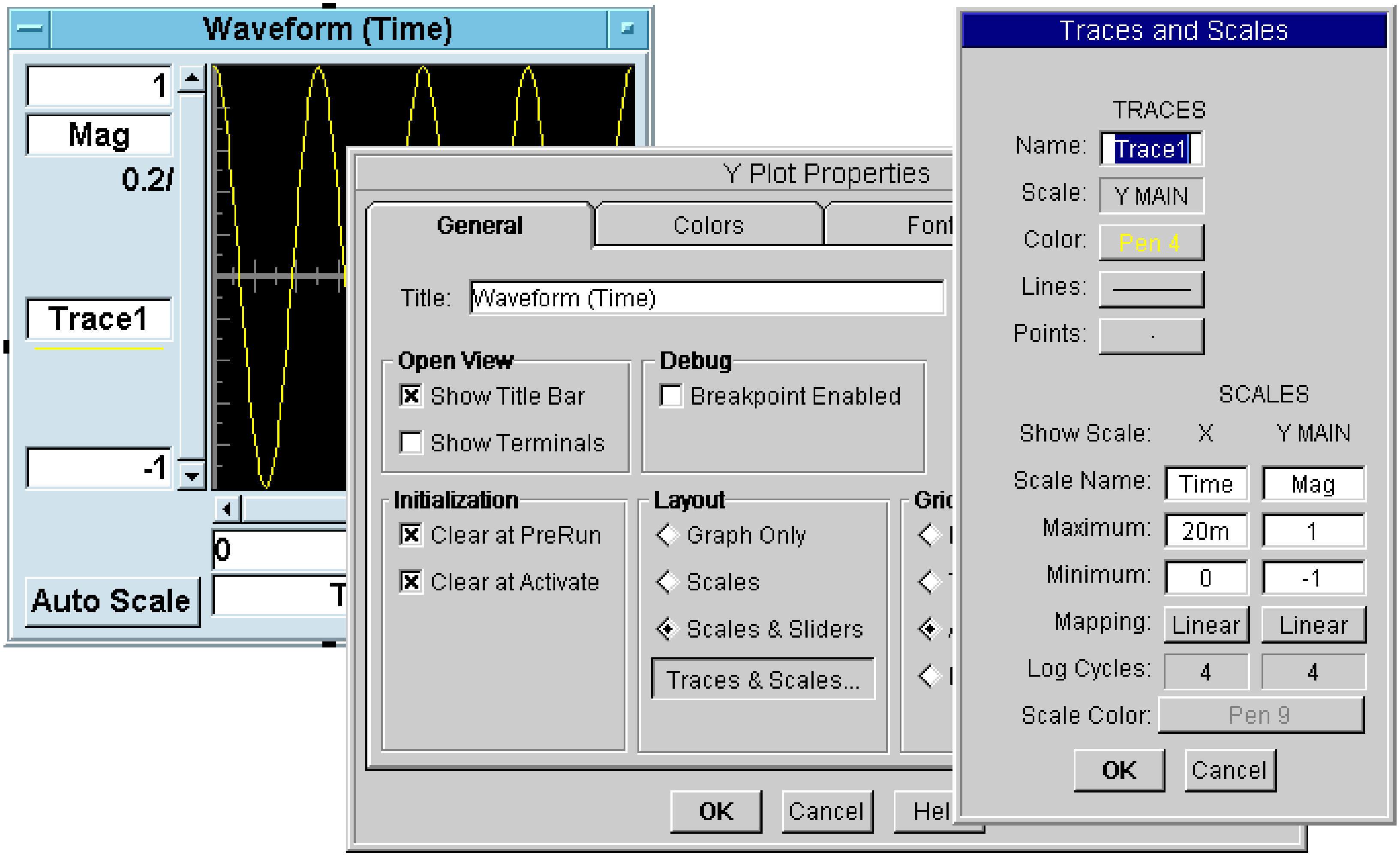
Grid Type
- ◇ None
- ◇ Tic Marks
- ◆ Axis
- ◇ Lines

Markers
- ◆ Off
- ◇ One
- ◇ Two
- ◇ Delta
- □ Interpolate

OK | Cancel | Help

HP VEE

Display Traces and Scales



Waveform (Time)

1

Mag

0.2/

Trace1

-1

0

Auto Scale

Y Plot Properties

General | Colors | Font

Title: Waveform (Time)

Open View
☒ Show Title Bar
☐ Show Terminals

Debug
☐ Breakpoint Enabled

Initialization
☒ Clear at PreRun
☒ Clear at Activate

Layout
◇ Graph Only
◇ Scales
◆ Scales & Sliders

Traces & Scales...

Grid

OK | Cancel | Hel

Traces and Scales

TRACES

Name: Trace1

Scale: Y MAIN

Color: Pen 4

Lines: ———

Points: ·

SCALES

Show Scale:	X	Y MAIN
Scale Name:	Time	Mag
Maximum:	20m	1
Minimum:	0	-1
Mapping:	Linear	Linear
Log Cycles:	4	4
Scale Color:	Pen 9	

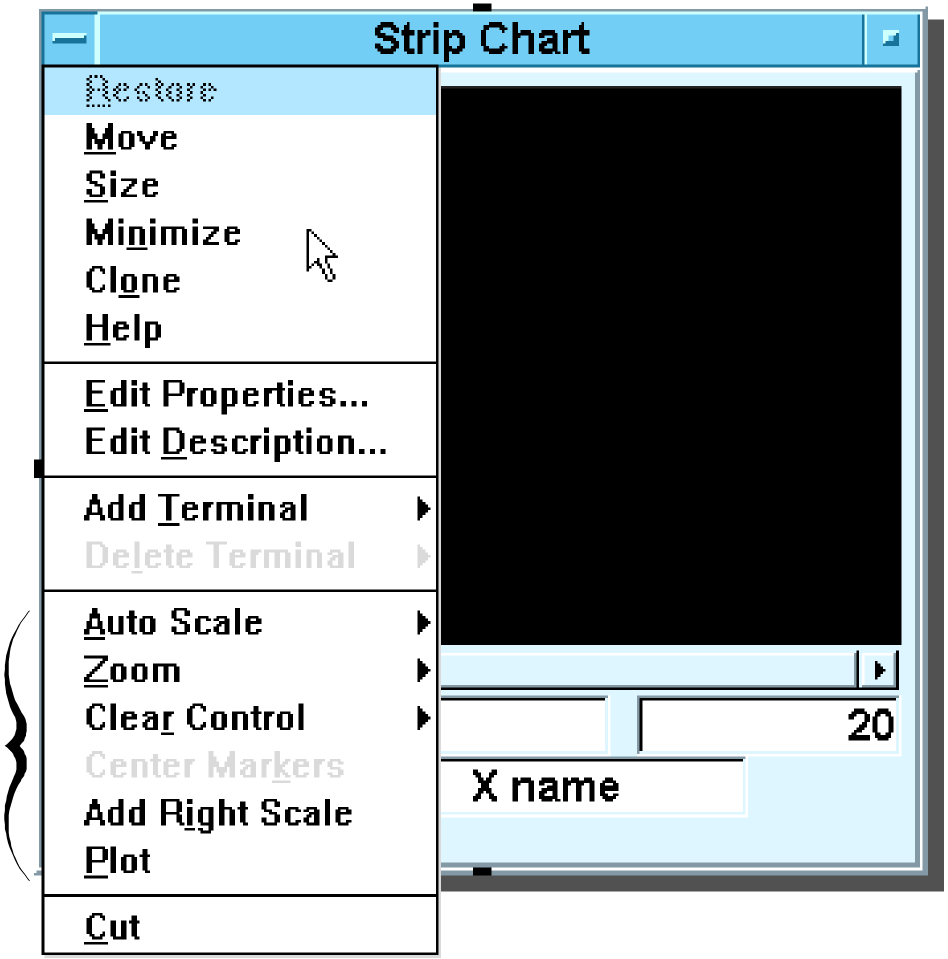
OK | Cancel

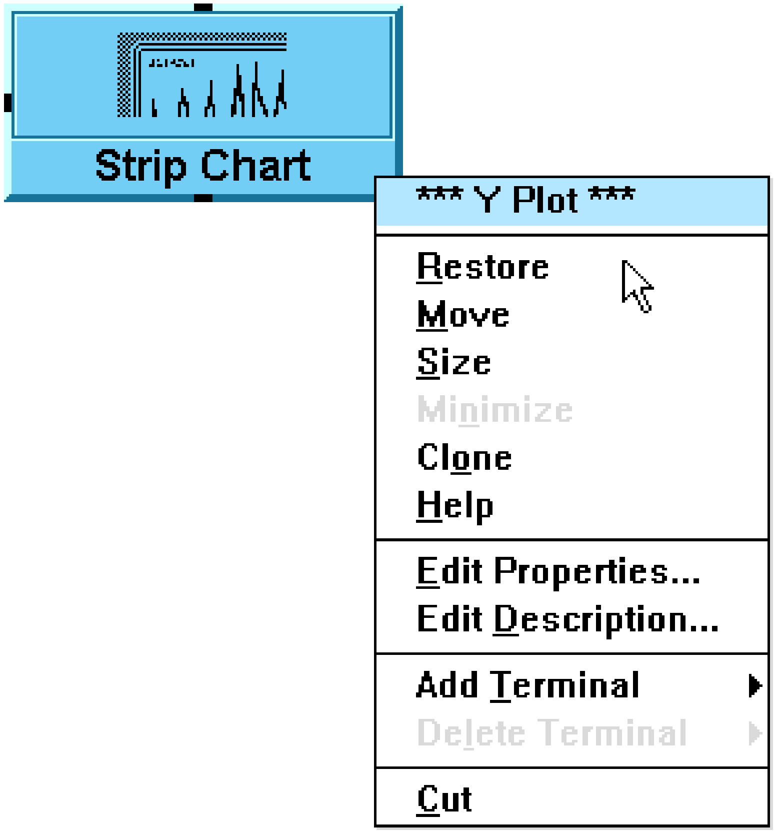
HEWLETT PACKARD

Object Menu

OPEN VIEW

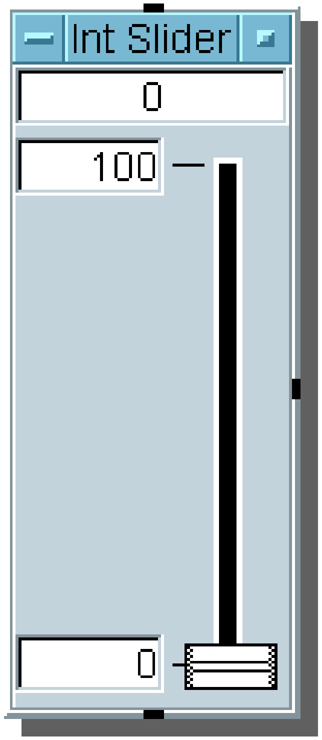
ICON VIEW

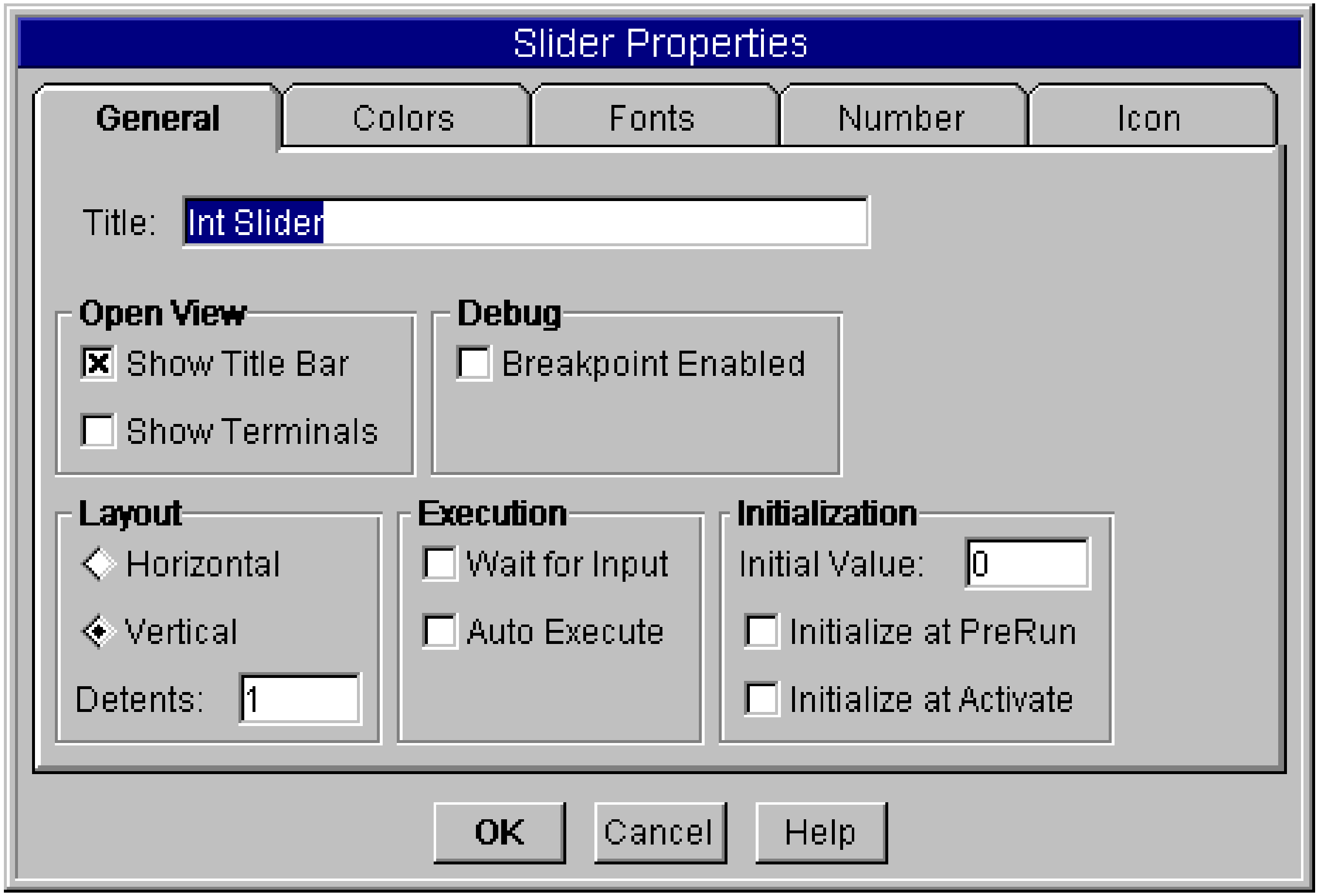




HP VEE

Object Properties





Slider Properties

Title: Int Slider

Open View

☒ Show Title Bar

□ Show Terminals

Debug

□ Breakpoint Enabled

Layout

○ Horizontal

● Vertical

Detents: 1

Execution

□ Wait for Input

□ Auto Execute

Initialization

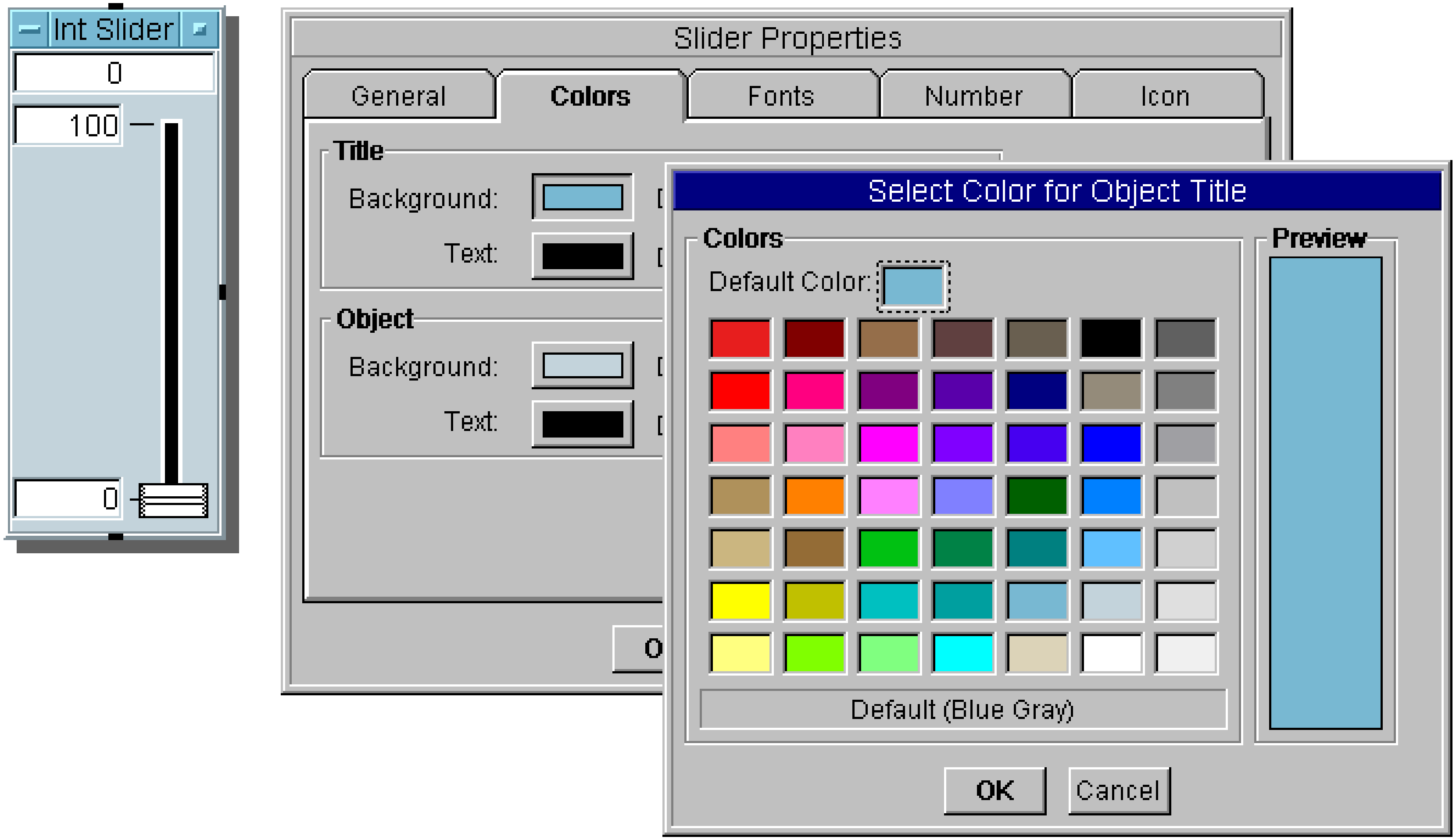
Initial Value: 0

□ Initialize at PreRun

□ Initialize at Activate

HP VEE

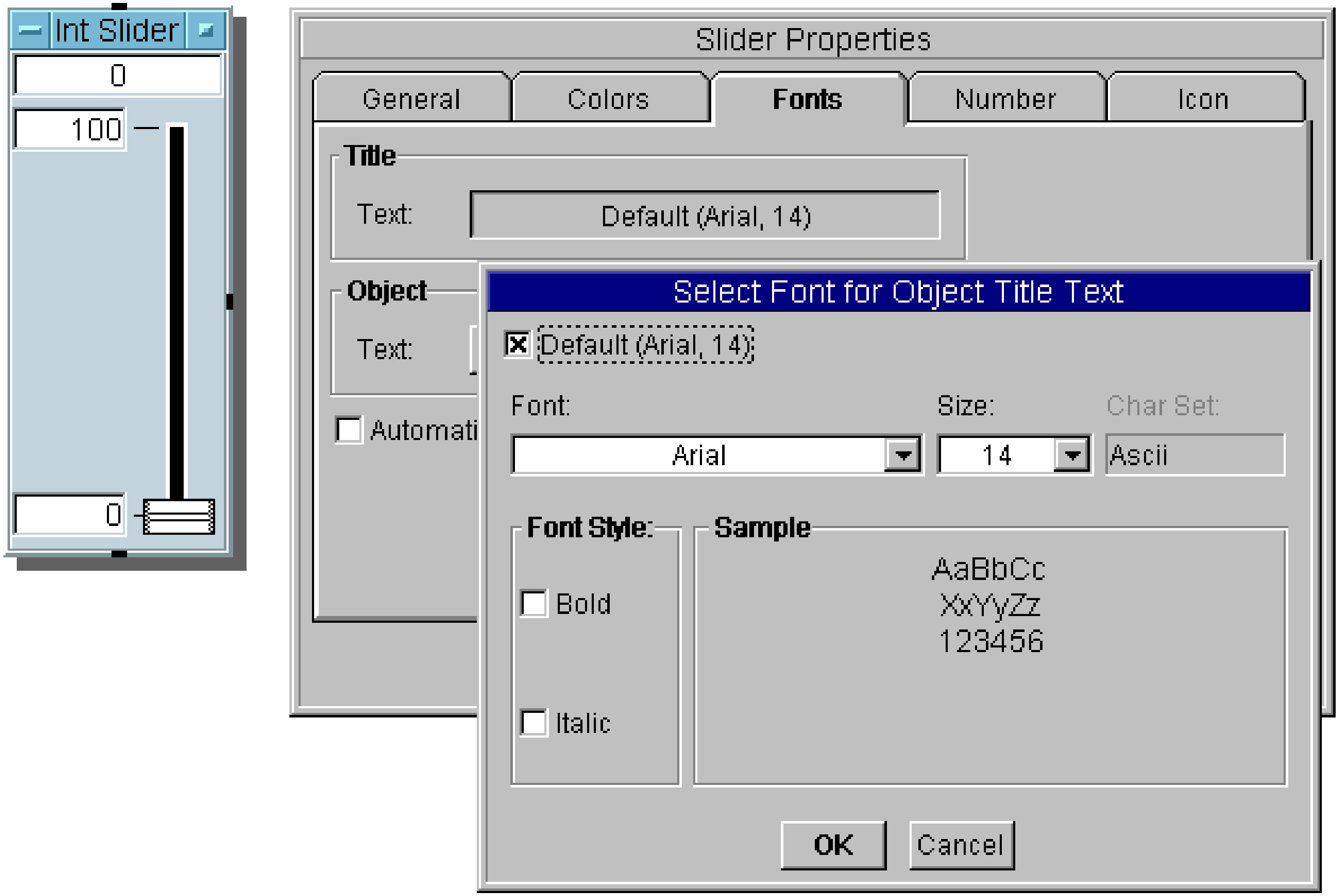
Object Color



HEWLETT PACKARD

HP VEE

Object Font



Int Slider

0

100

0

Slider Properties

General | Colors | **Fonts** | Number | Icon

Title

Text: Default (Arial, 14)

Object

Text:

□ Automati

Select Font for Object Title Text

☒ Default (Arial, 14)

Font: Arial

Size: 14

Char Set: Ascii

Font Style:

□ Bold

□ Italic

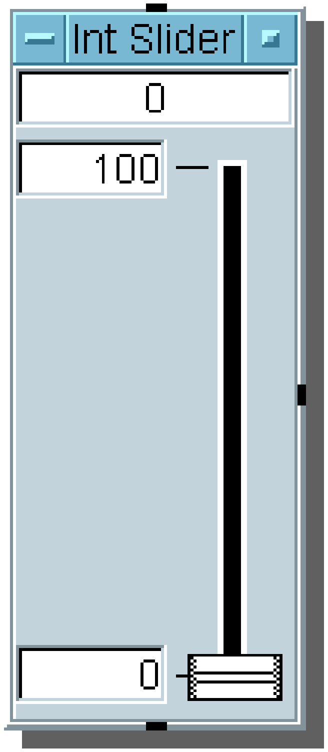
Sample

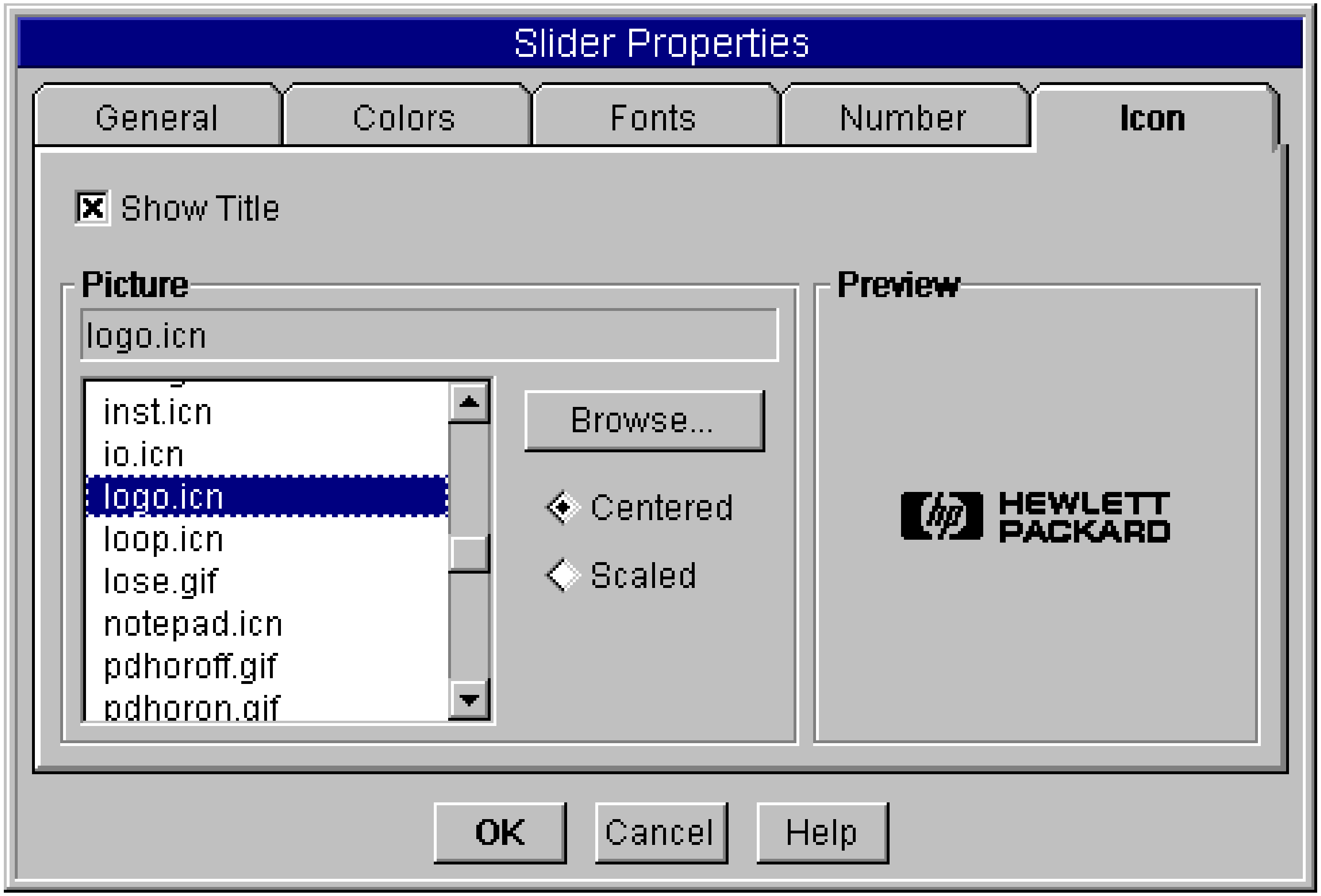
AaBbCc
XxYyZz
123456

OK | Cancel

HEWLETT
PACKARD

Object Icon





Slider Properties

☒ Show Title

Picture

logo.icn

- inst.icn
- io.icn
- logo.icn
- loop.icn
- lose.gif
- notepad.icn
- pdhoroff.gif
- pdhoron.gif

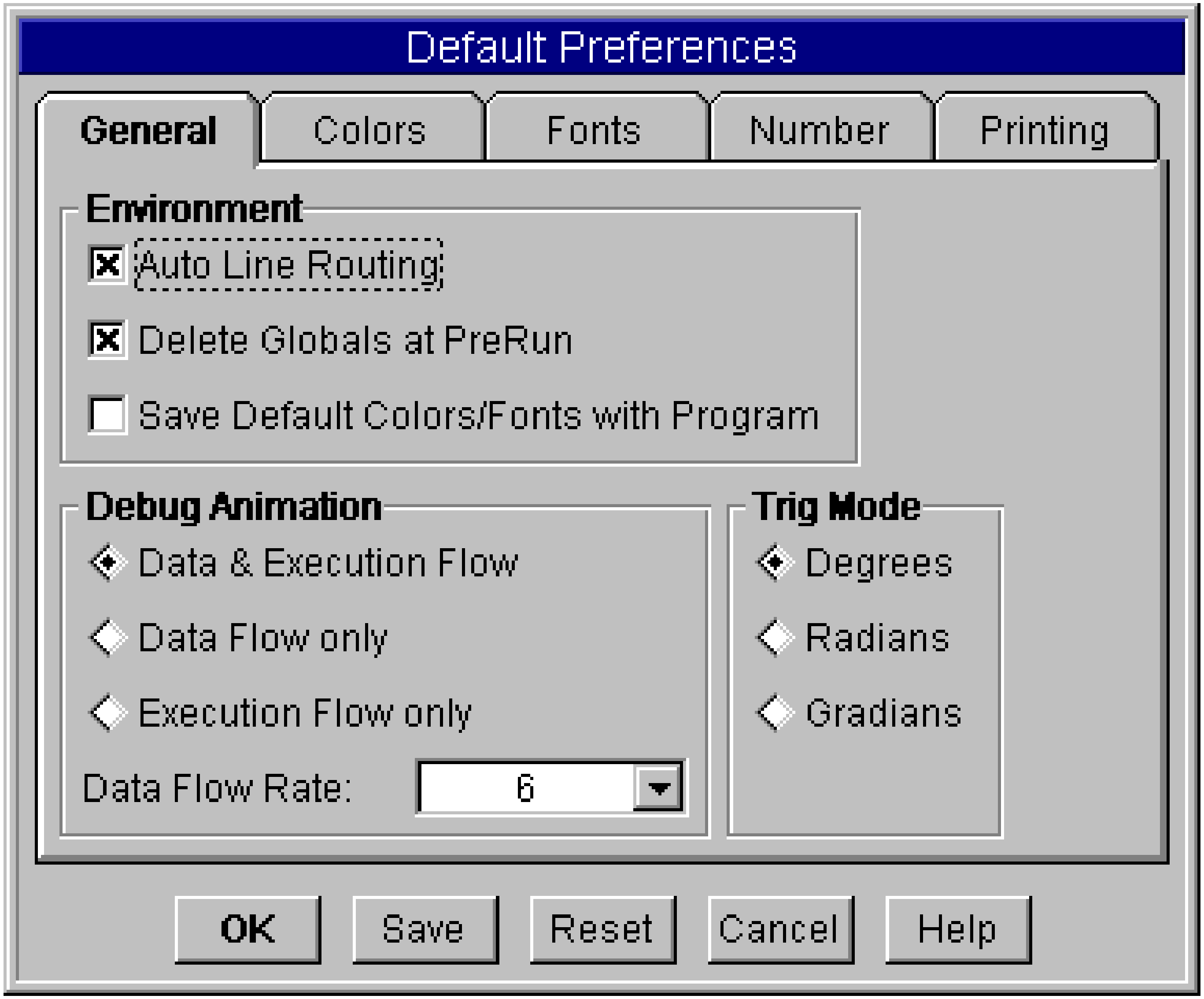
● Centered

○ Scaled

Preview

HP VEE

Default Preferences



Environment

☒ Auto Line Routing

☒ Delete Globals at PreRun

□ Save Default Colors/Fonts with Program

Debug Animation

● Data & Execution Flow

○ Data Flow only

○ Execution Flow only

Data Flow Rate: 6

Trig Mode

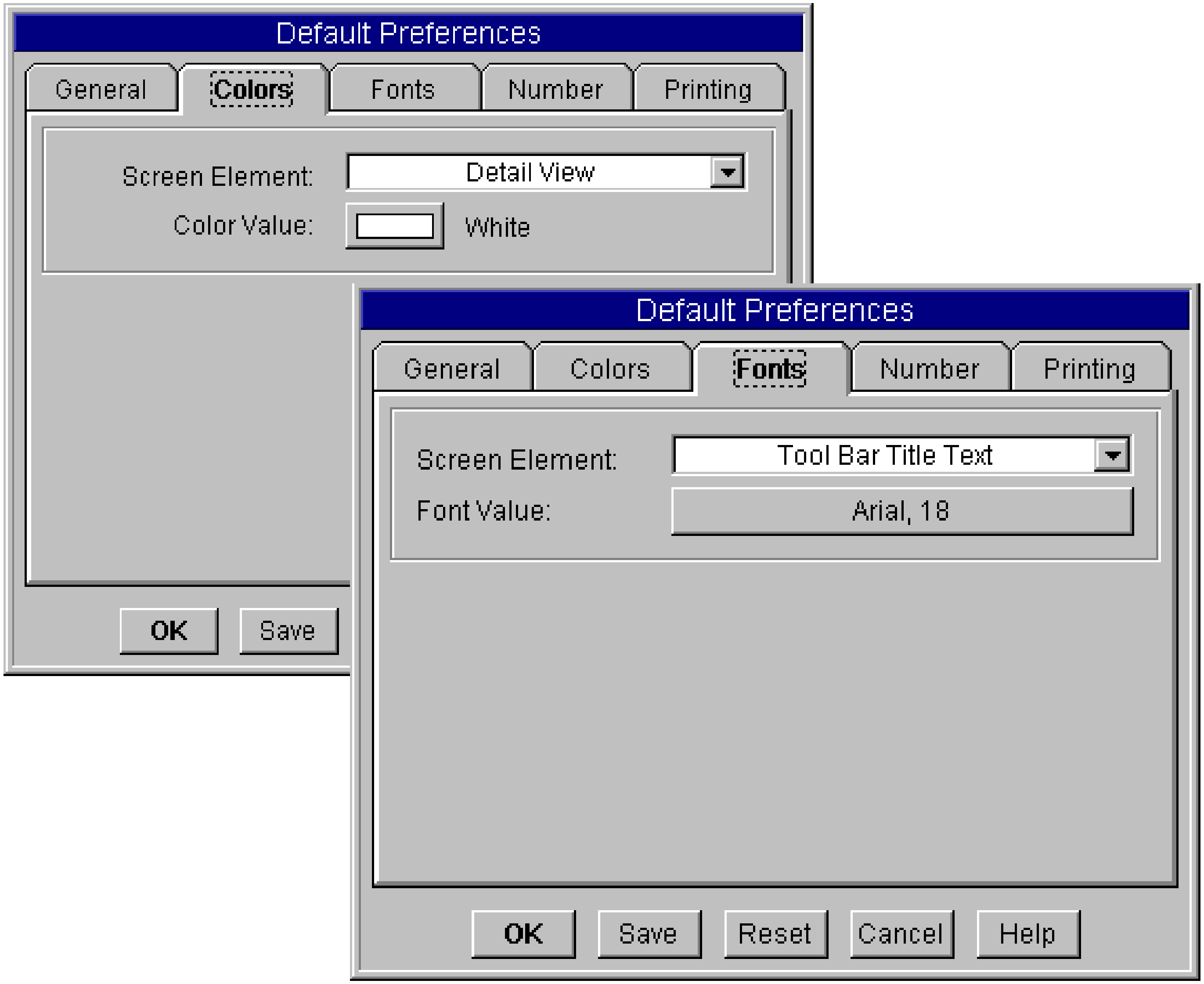
● Degrees

○ Radians

○ Gradians

HEWLETT PACKARD

Colors and Fonts



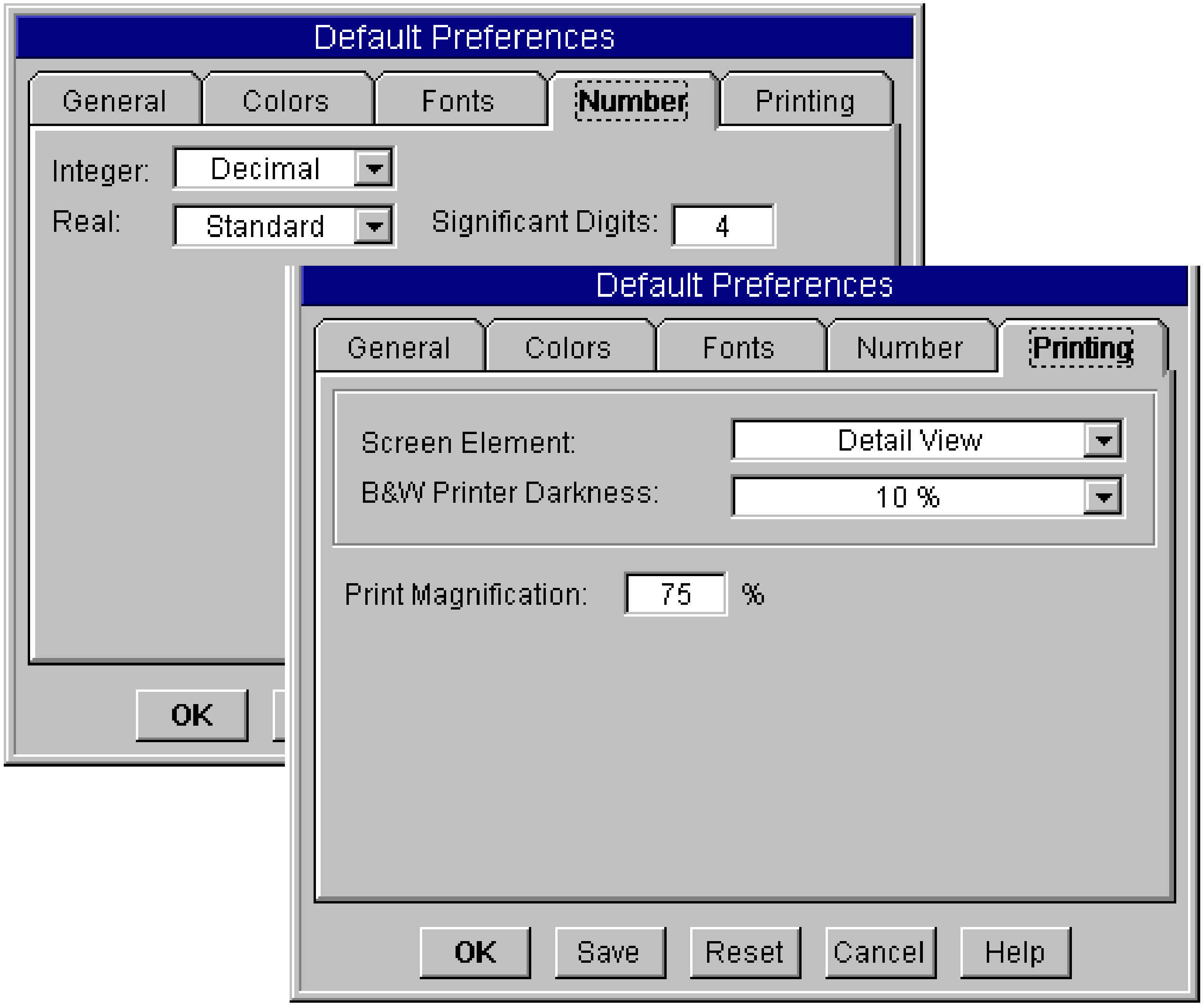
Screen Element: Detail View

Color Value: □ White

Screen Element: Tool Bar Title Text

Font Value: Arial, 18

Numbers and Printing



Integer: Decimal

Real: Standard

Significant Digits: 4

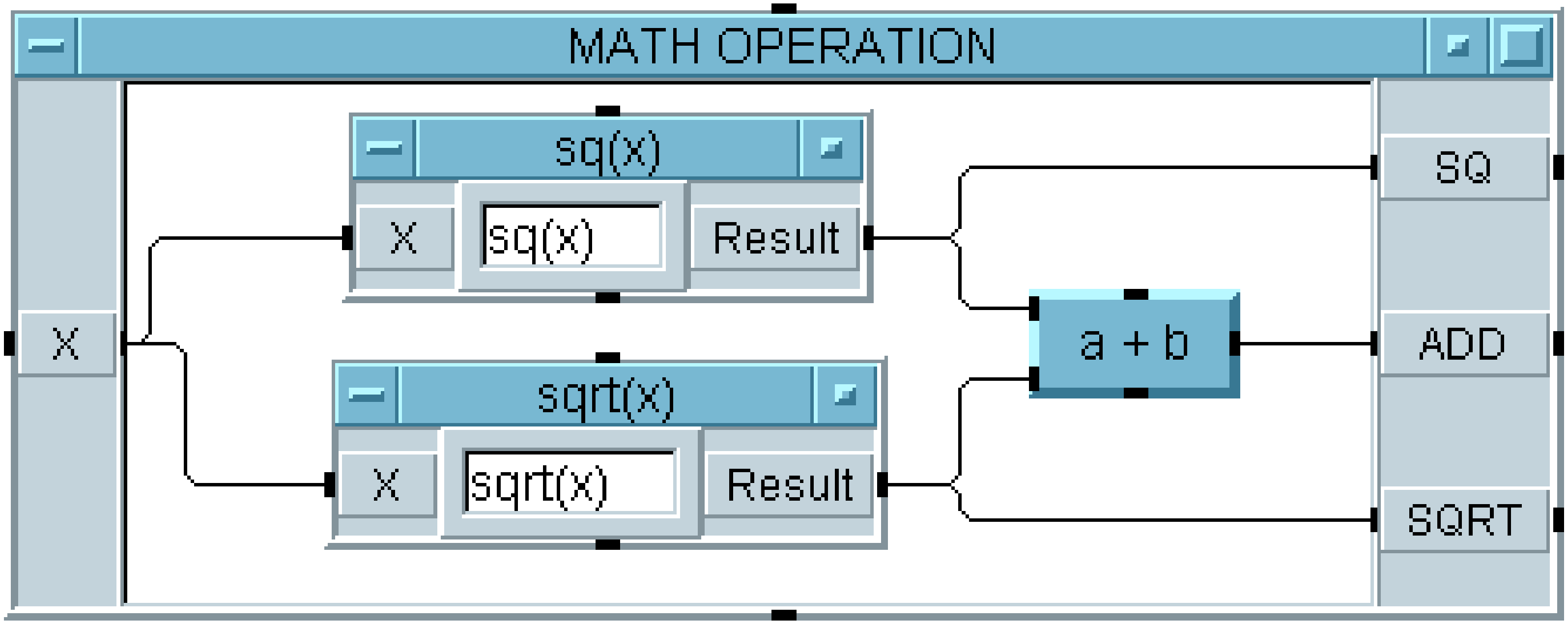
Screen Element: Detail View

B&W Printer Darkness: 10 %

Print Magnification: 75 %

HP VEE UserObjects

- Provides a work area within an object



- A context

UserObjects

Behavior

- Obey all the rules of objects
  - Need all Data and Sequence inputs satisfied

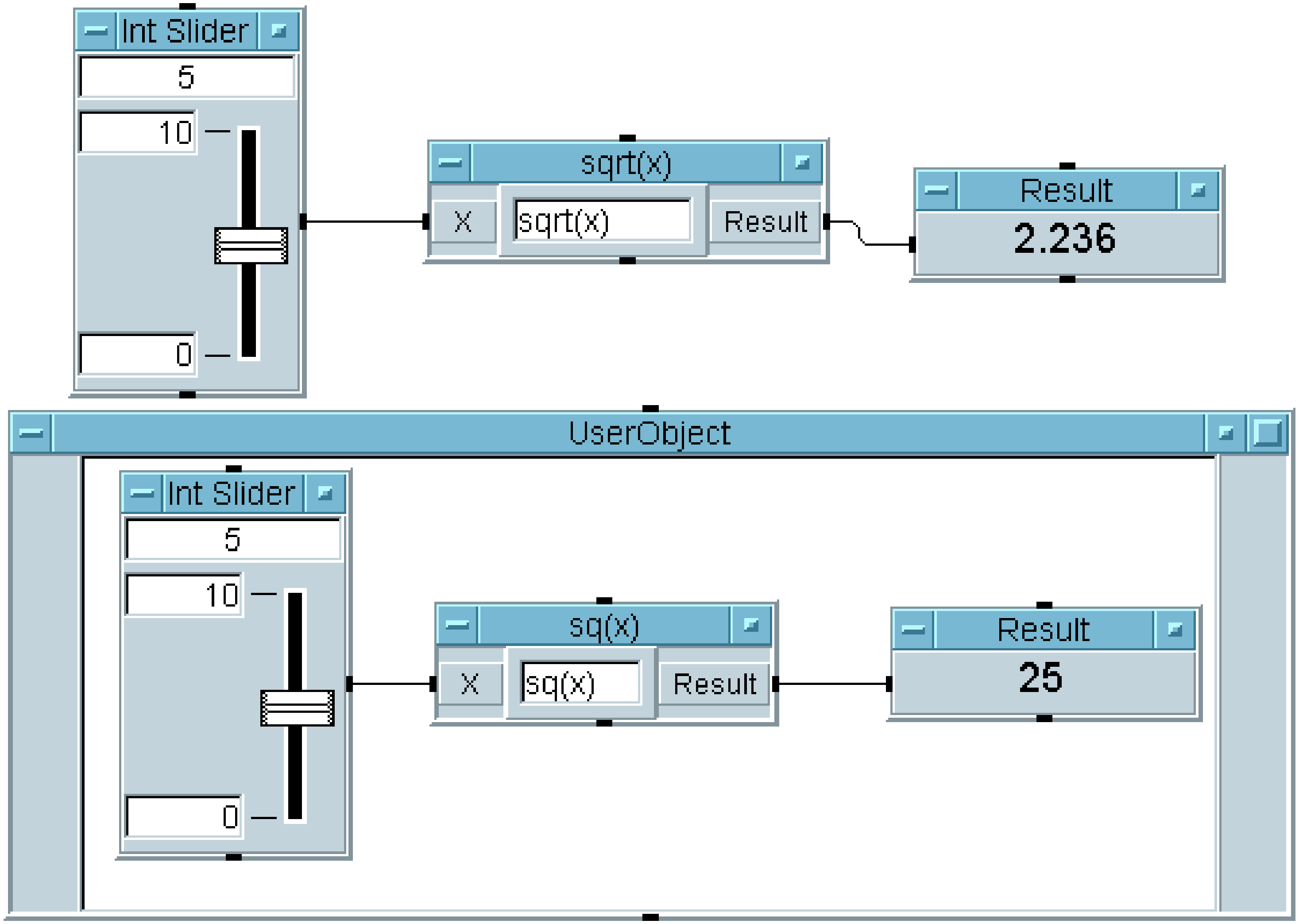
- Behave like work area
  - Supports all objects like main work area
    - Supports multiple threads
    - May be embedded within other UserObjects or UserFunctions

Purpose

- Encapsulate groups of objects that provide a function into a single object
  - Unclutters work area
  - Facilitates easy understanding of programs behavior
- Allows modular ("top down") design
  - Unlimited nesting
- Can be stored in central object directory
  - Easy sharing and re-use

    File ==> Merge

Parallel Thread Example

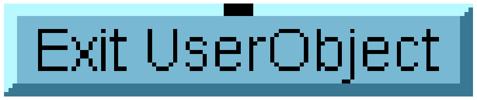


Termination of UserObjects

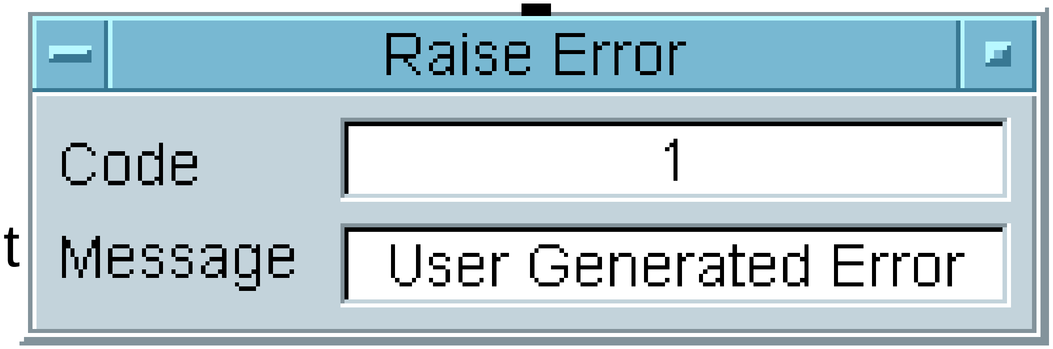
- Causes of deactivation
  - All threads operate to completion
  - Exit UserObject
  - Untrapped error
- Results
  - Data output pins activate ONLY those pinged within context
  - Sequence out activates

Early Termination

- Exit UserObject
  - All threads in context halt
  - Outputs which received data activate
  - Sequence out activates
  - Allows you to create a custom dialog box with a "cancel" button



- Raise Error
  - User-generated error
  - All threads in context halt
  - NO data pins activate
  - Error pin propagates escape code, otherwise an Error dialog box



Raise Error

Code 1

Message User Generated Error

Encapsulation with a UserObject

Bottom Up Design

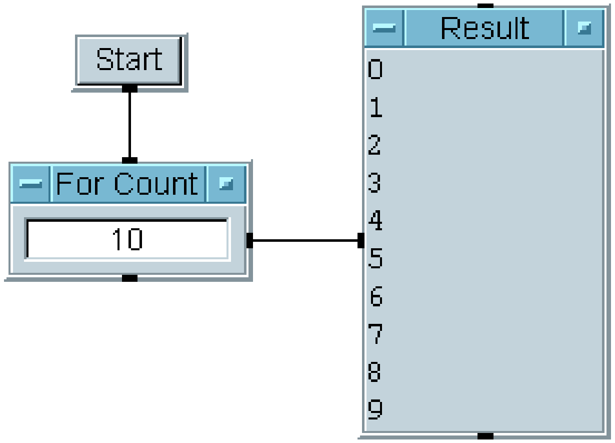
- Encapsulate Existing Object(s)
  - Select desired object(s)
  - Edit ==> Create UserObject

- Advantages
  - All connections become data pins
  - Allows prototyping in main work area

- Disadvantages
  - Redundant connections must be edited
  - Ill-conceived object selection yields nonfunctional UserObject

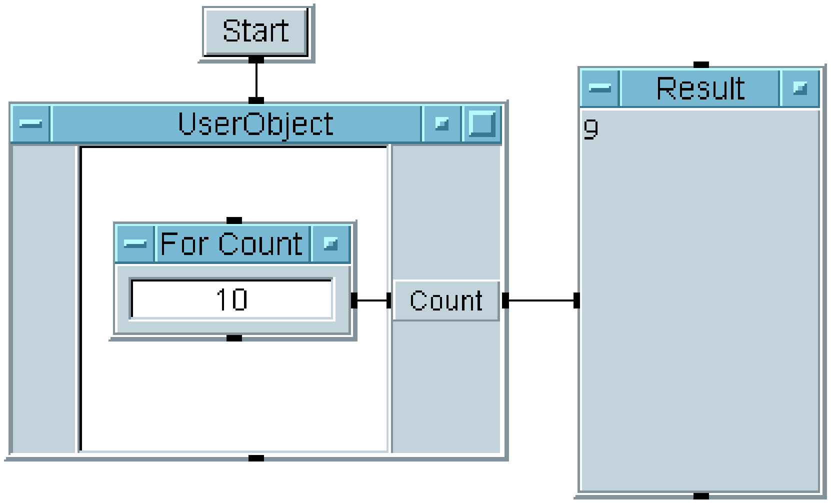
Common Problem in Create UserObject

- The working program ...



```
0
1
2
3
4
5
6
7
8
9
```

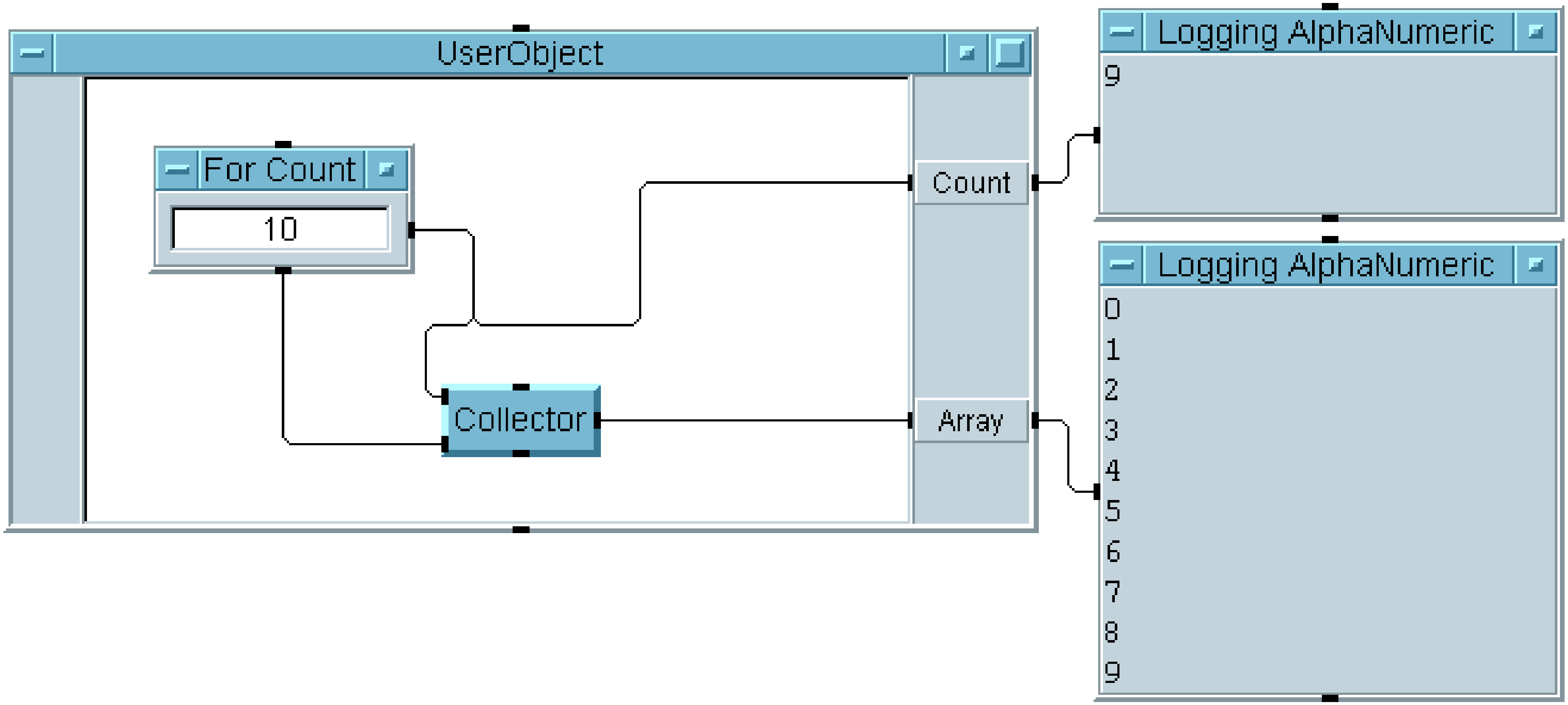
- ... partially encapsulated



```
9
```

- GIVES DIFFERENT RESULTS!

UserObject With Collector



UserObject - Top Down Design

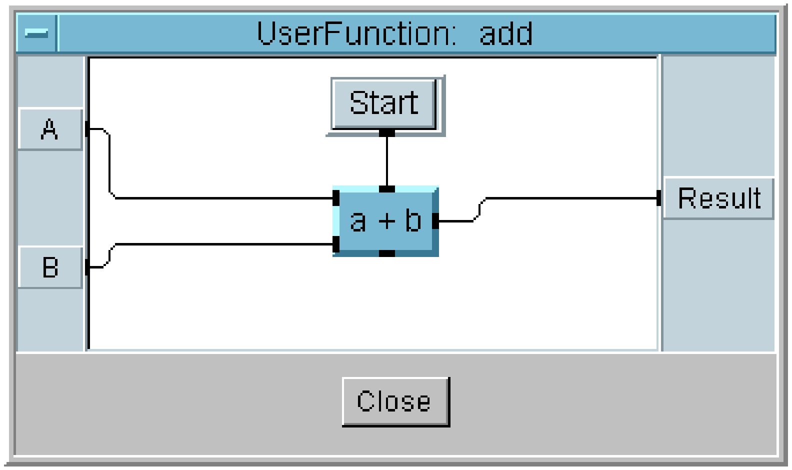
- Define the problem and its constraints
- Identify and define logical order and sequence
- Define subtasks
- Implement units
  - Device ==> UserObjects
- Structured programming
  - Exactly same principles apply as in textual languages

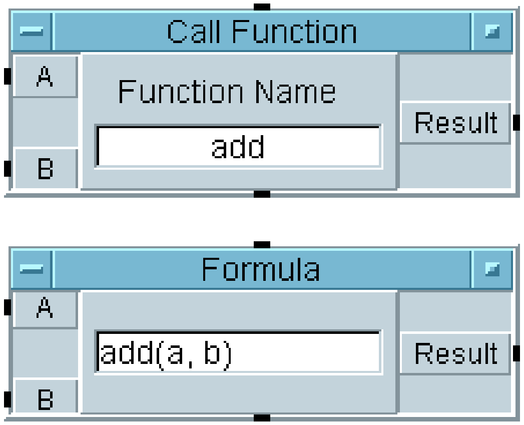
Building With a UserObject
Top Down Design

- Start with an empty UserObject
  - Device ==> UserObject
- Build the model that will provide the basic unit of functionality
- Add data inputs and outputs
- Test individually
- No symbolic procedure calls with UserObjects
  - No recursion
  - Multiple occurrences = multiple copies

HP VEE UserFunctions

- "Next Generation" of UserObjects
- This allows users to change the source and have the changes reflected everywhere
- Large applications with duplicate code will require less memory and take less disk space





Function Name: add

add(a, b)

Functions

- Three types
  - UserFunctions (made from UserObjects)
  - Compiled Functions (made from C programs)
  - Remote Functions (UserFunctions that run on another system)
- Allow you to build and call custom VEE functions
- Represent a uniform method to scale ease-of-use vs. performance

UserFunctions

- Created from UserObjects
- Have the same synchronous operation as other objects
- Can be local to a program or imported from a library

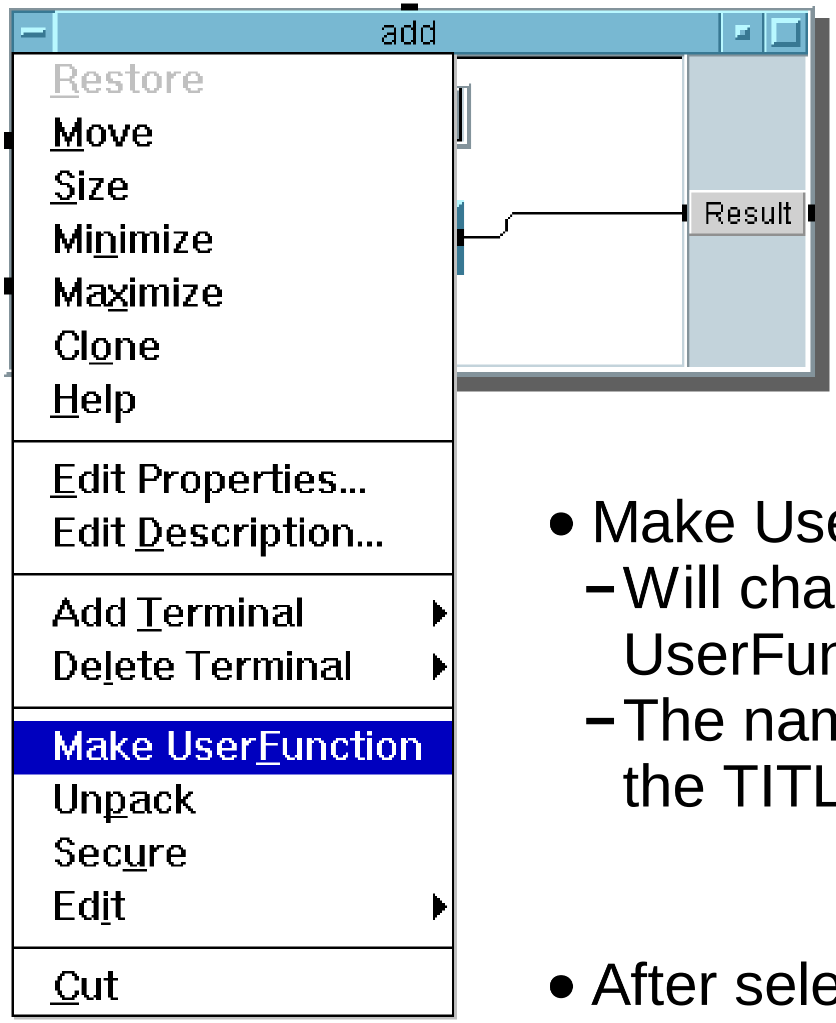
UserFunctions

- Edit local UserFunctions in one place - have the changes effected everywhere
  - Edit ==> Edit UserFunction

- Don’t clone - smaller programs mean faster loading, faster editing

- UserFunctions imported from a library may be viewed

Making UserFunctions

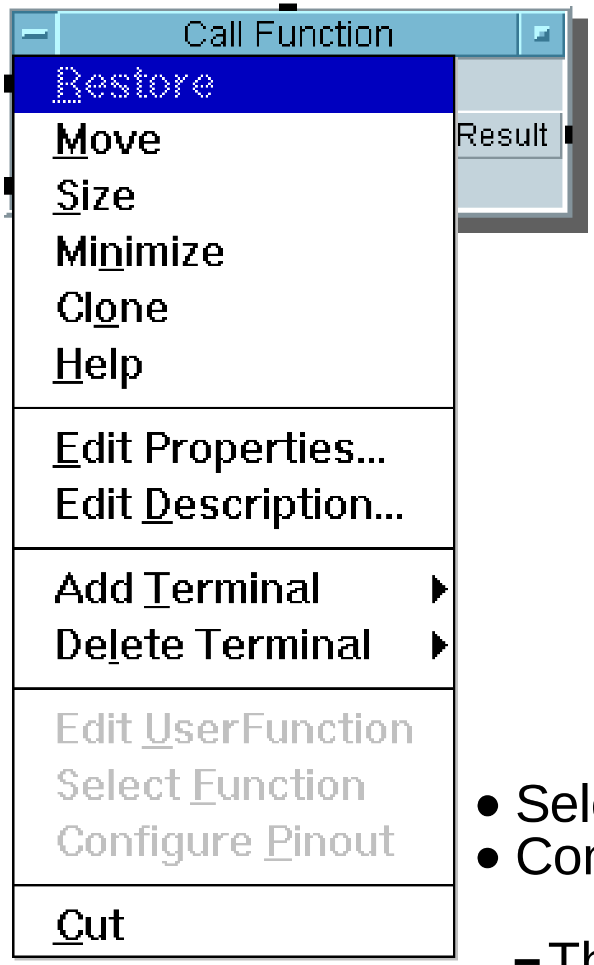


- Make UserFunction
  - Will change the UserObject into a UserFunction
  - The name of the Function is taken from the TITLE of the UserObject
- After selecting "make userfunction", it is now available with this program to call, edit, etc.

HP VEE

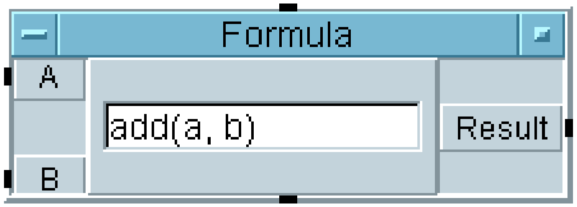
Calling UserFunctions

Device ==> Function ==> Call
e



- Select Function...
- Configure pin-outs
  - The function name can be specified as a control pin

or from ANY



In an expression -

- Input terminals are function parameters to pass in
- You can only retrieve data from the top data output terminal

Note: Don’t forget the null parentheses () if no data is sent

Merge Library

File ==> Merge Library

- Allows you to merge UserFunctions from a saved program file into your active program
- This will merge ALL UserFunctions from the file

Edit UserFunctions

Edit ==> Edit UserFunctions

- Only available for local functions that you created or merged
- Multiple edit windows can STAY open while a program is executing
- If you add or delete terminals after creating a UserFunction - you must go the CALL objects and select "Configure Pin-out"

Import/Delete/Call UserFunctions

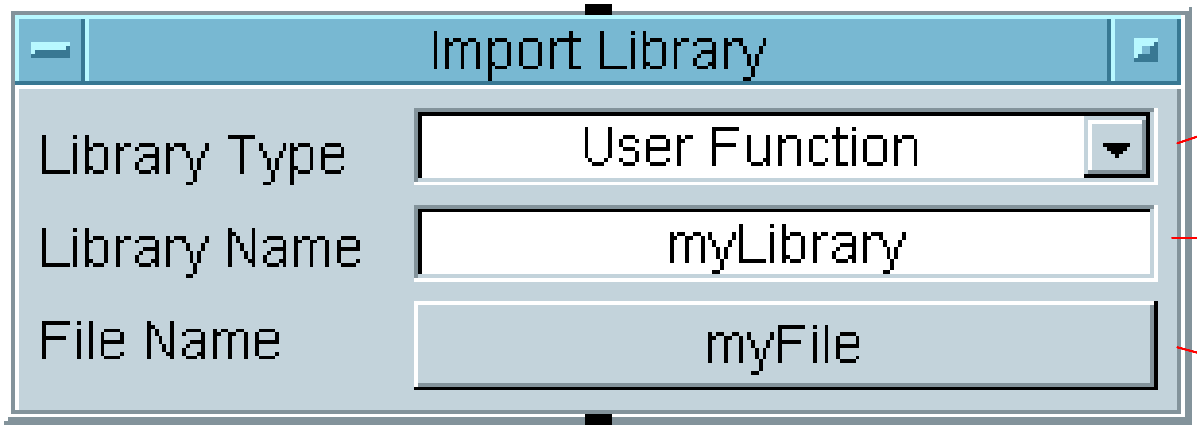
- To allow a UserFunction to be shared among multiple VEE programs

- To spread out load times

- To only load the parts of a program that are required

Import/Delete UserFunctions

- Allows you to dynamically load functions from a file to use in your program



Import Library

Library Type: User Function

Library Name: myLibrary

File Name: myFile

UserFunction

"handle" used for deleting library later

file that is the library of functions

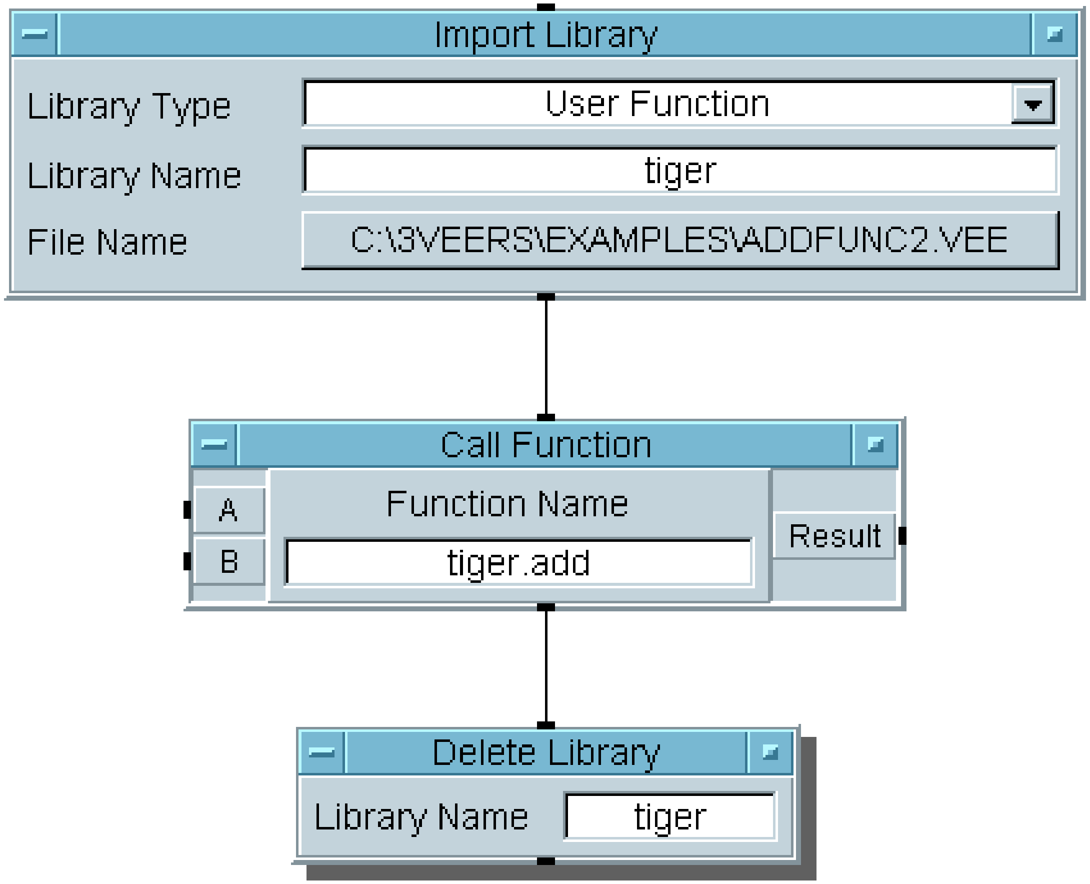
- Also - the object menu includes a "load lib" or "delete lib" choice for interactively loading or unloading of library.

Import/Delete/Call UserFunctions

HP VEE

Remember:
Load Lib lets you manually load in the functions and test to be certain VEE can find the file

Remember:
Select Function will present a list of UserFunctions that are available.



Import Library

Library Type: User Function

Library Name: tiger

File Name: C:\3VEERS\EXAMPLES\ADDFUNC2.VEE

Call Function

Function Name: tiger.add

Delete Library

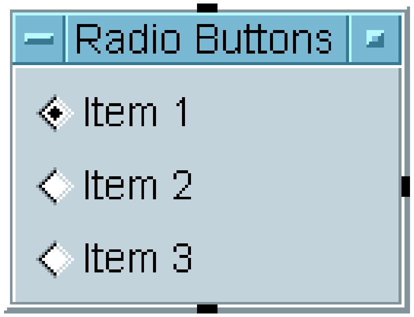
Library Name: tiger

User Interaction

- Definition - a user is someone who runs a program developed by someone else
  - User Inputs
  - Customization
  - Panel Views
  - Secure Programs
  - Combining Panels and UserObjects/UserFunctions

User Inputs

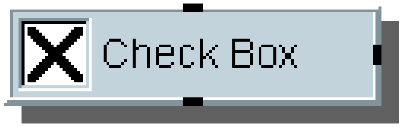
- Selection Control, Toggle Control, Sliders, Constants, Dialog Boxes
  - Allow developer to prompt user for a variety of inputs
  - Each input object allow Auto Execute and "Wait For Input" except for Dialog Boxes or Array Constants
  - Users input values without having to RE-START the

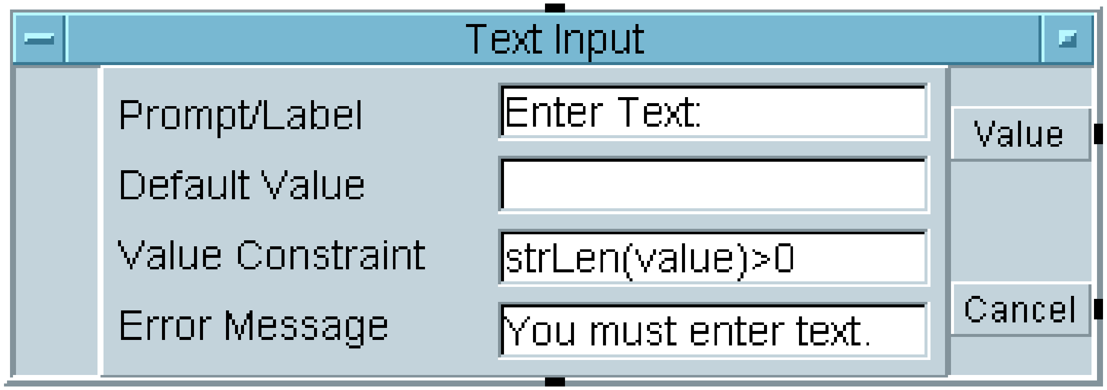

Radio Buttons

◆ Item 1

◇ Item 2

◇ Item 3


☒ Check Box


Text Input

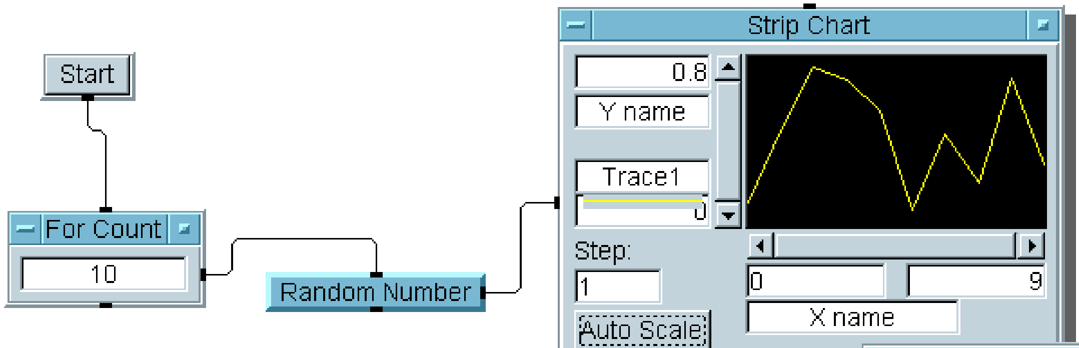
Prompt/Label	Enter Text:
Default Value	
Value Constraint	strLen(value)>0
Error Message	You must enter text.

User Customization Features

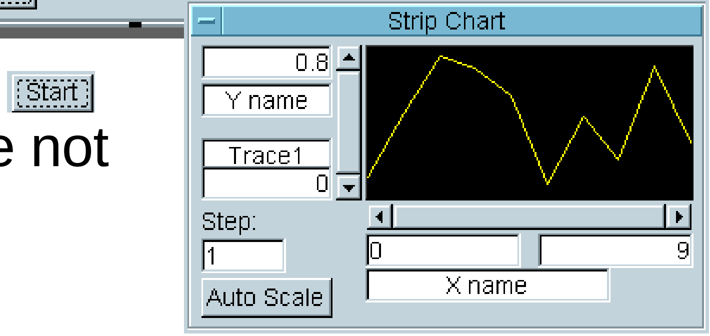
- Ability to size objects
- Ability to customize display features and colors
- Ability to annotate a program
  - Notepads
  - Custom object or program titles
  - Object descriptions
  - Labels and Pictures
- Properties for each object and for the work area

Panel View
A simpler view of the program

- Developer chooses objects from the Detail View



- Data and sequence lines are not on the Panel View



Panel Views

- Show only the necessary objects
- Secure the program from user intervention
- Provide an easy to read interface to a potentially complex program
- Improve performance by decreasing screen interaction

Creating a Panel View

- Build the program and verify that it runs properly
- Select the one or more objects you want to show on the Panel View
- Select Edit ==> Add to Panel
- Move and size objects on the Panel to maximize its effectiveness
- Press Panel and Detail to move between views

Panel View Characteristics

- Fewer choices appear on the main menu in Panel View
- If you can cut an object on the Detail View, its corresponding object on the Panel is gone
- The appearance (size, location, etc.) is not shared between views
- Shared values include:
  - Initialize Values
  - Clear Values
  - Data
  - Etc.

Securing a Panel View

- Creates a panel that does not allow a user to access the Detail View

- Three Step Process:
  1. Create the program with a Panel View;
  2. Select Secure, and save the source file

     The Panel View is available yet can no longer be edited
  3. Save the secured program to another file

Be certain to select a unique name to save the secured program
so you don’t overwrite the source file

UserObjects With Panel Views

- A UserObject is an independent work area
- Developers can create a Panel View for the UserObject
- Select objects within the UserObject and then use the UserObjects work area Edit menu to "Add a Panel"
  object menu Edit ==> Add to Panel
- Same applies to UserFunctions

Using "Show Panel on Execute"

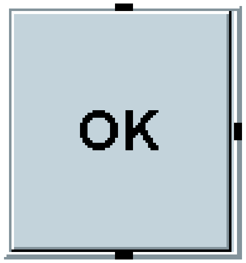
- Create a UserObject with a Panel View
- Edit Properties... of the UserObject
- General tab "Show Panel on Execute"
- When the UserObject operates, the panel "Pops Up" on the work area
- Again, this works the same for UserFunctions

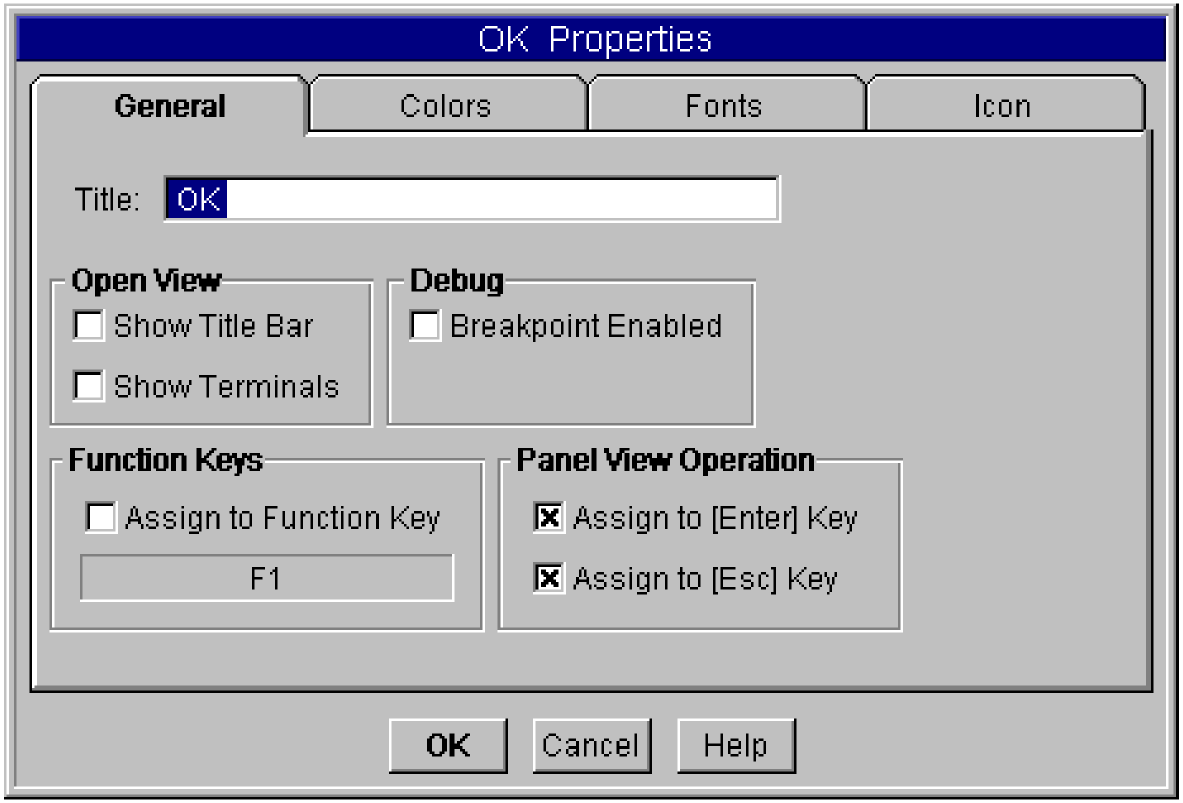
"Show Panel on Execute" Operation

- When the UserObject operates, the panel opens up in the center of the work area
- The Panel View of the UserObject disappears when the UserObject finishes - so -
- To use this feature effectively - developers should use the Confirm (OK) object to pause execution until the user responds

OK Button - Keyboard Actions

- OK Button can be assigned to Function, Enter and ESC keys





OK Properties

General | Colors | Fonts | Icon

Title: OK

Open View

☐ Show Title Bar

☐ Show Terminals

Debug

☐ Breakpoint Enabled

Function Keys

☐ Assign to Function Key

F1

Panel View Operation

☒ Assign to [Enter] Key

☒ Assign to [Esc] Key

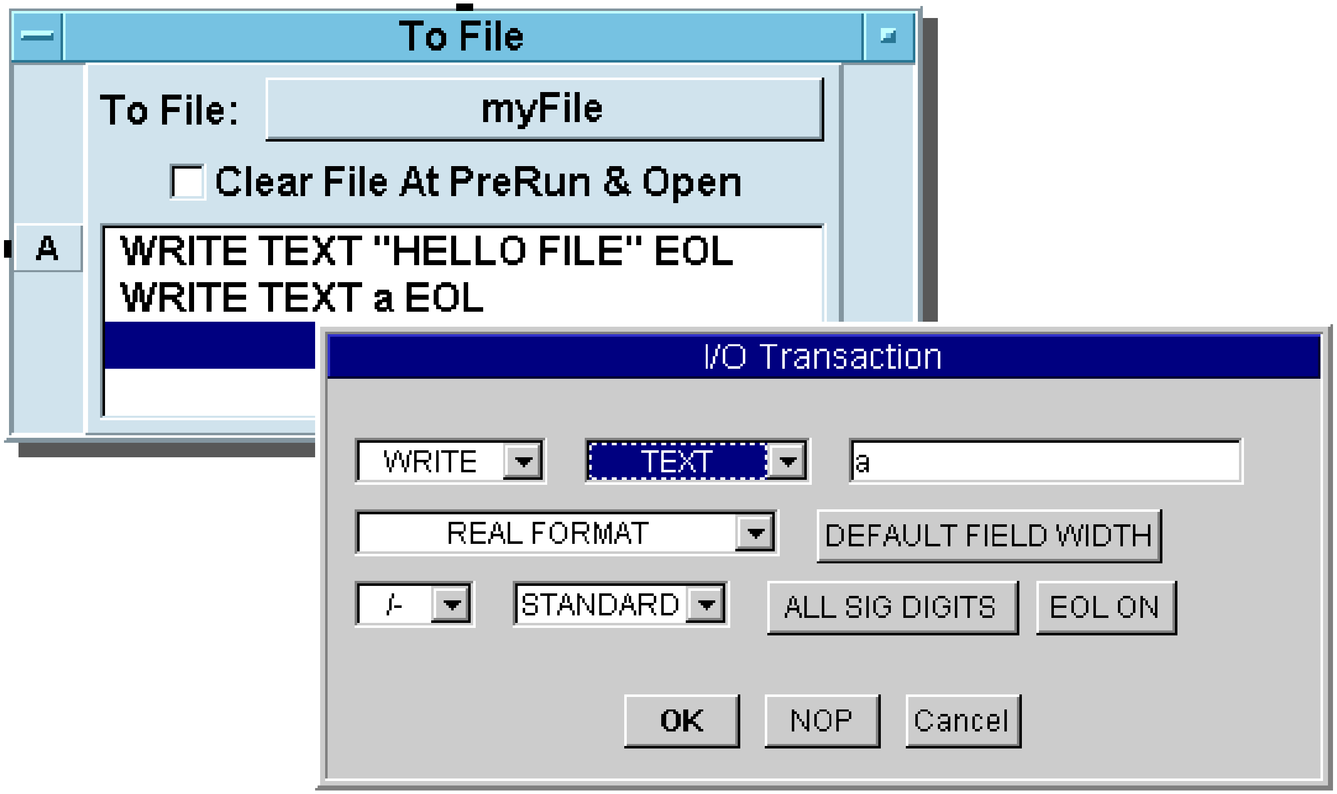
OK | Cancel | Help

HEWLETT PACKARD

Transactions

- Communication paths are implemented as Transaction-based objects

- Individual transactions handle multiple data items



To File

To File: myFile

□ Clear File At PreRun & Open

A

```
WRITE TEXT "HELLO FILE" EOL
WRITE TEXT a EOL
```

I/O Transaction

WRITE ▾ TEXT ▾ a

REAL FORMAT ▾ DEFAULT FIELD WIDTH

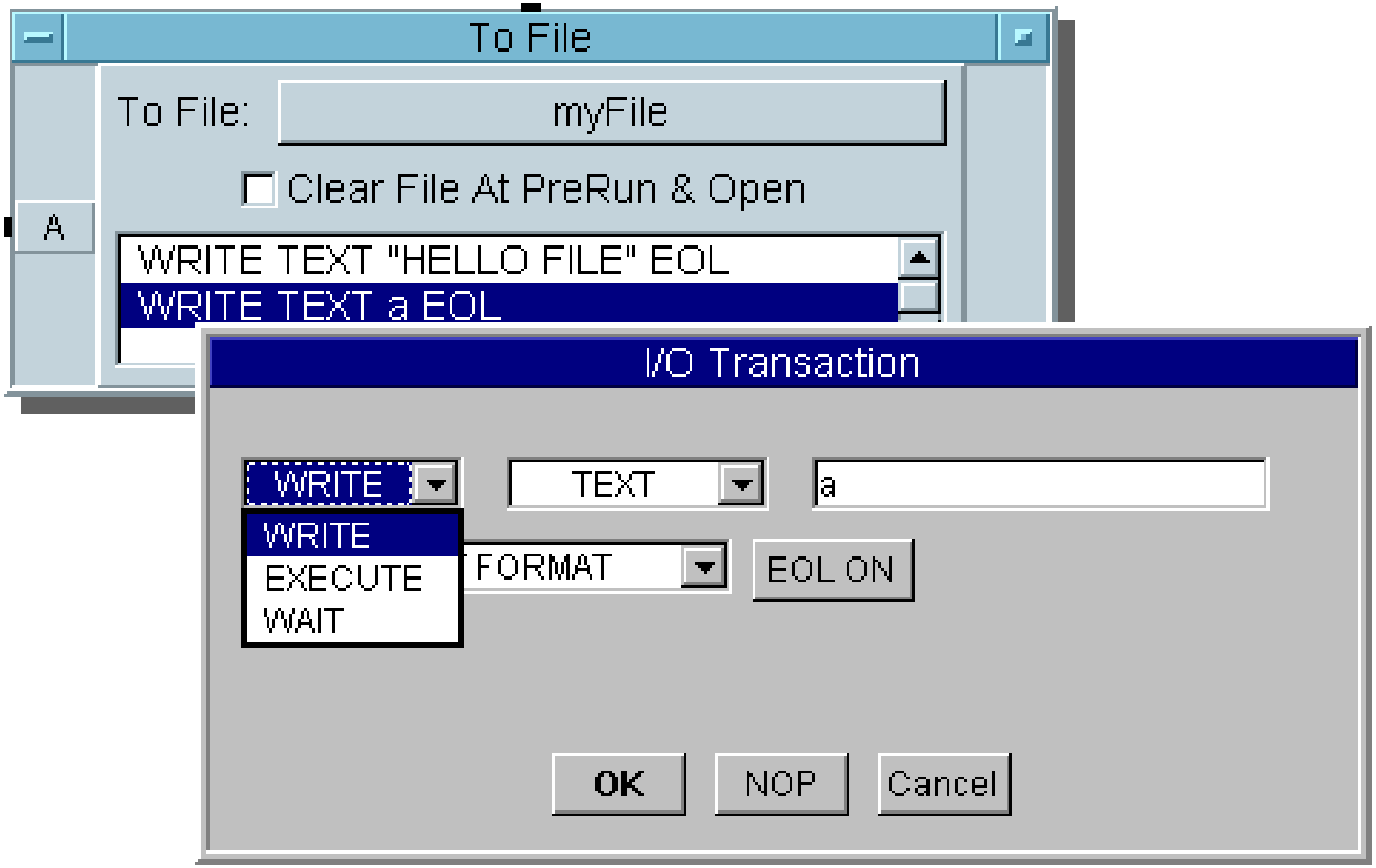
/- ▾ STANDARD ▾ ALL SIG DIGITS EOL ON

Transactions

- Specify action
  - READ, WRITE, EXECUTE, WAIT
- Specify data encoding (interpretation)
  - TEXT, BYTE, CASE for data being written
  - TEXT, BINARY, BINBLOCK, CONTAINER for data being read
- Specify formatting of data
  - Numerics represented at REAL, INTEGER, HEX, OCTAL
  - Full control of field width, justification

Actions

HP VEE



To File

To File: myFile

□ Clear File At PreRun & Open

```
WRITE TEXT "HELLO FILE" EOL
WRITE TEXT a EOL
```

I/O Transaction

WRITE | TEXT | a

HEWLETT PACKARD

TEXT Formats for READ - Text

- Match input data stream to required value and types
- Data conversion enforced
- Output pins take on type and shape required
- CHAR - Reads specified number of characters
  - Stored in string
- TOKEN - Allows multiple strings to be entered from data stream
  - SPACE DELIM - strings are separated by spaces
  - INCLUDE CHARACTERS - strings delimited by any non-member of set
  - EXCLUDE CHARACTERS - strings delimited by any member of set
- STRING -Reads all characters up to a specified limit

Text Formats for READ - Numeric

- OCTAL - Attempt to build INT32 value from
  numeric
  HEXADECIMAL data received
  INTEGER - OCTAL accepts 0..7
  HEX accepts 0..9, a-f, A-F
  INTEGER accepts 0..9

- REAL - Builds REAL64 value
  - Accepts 0..9, +, -, e, E, . (decimal point)

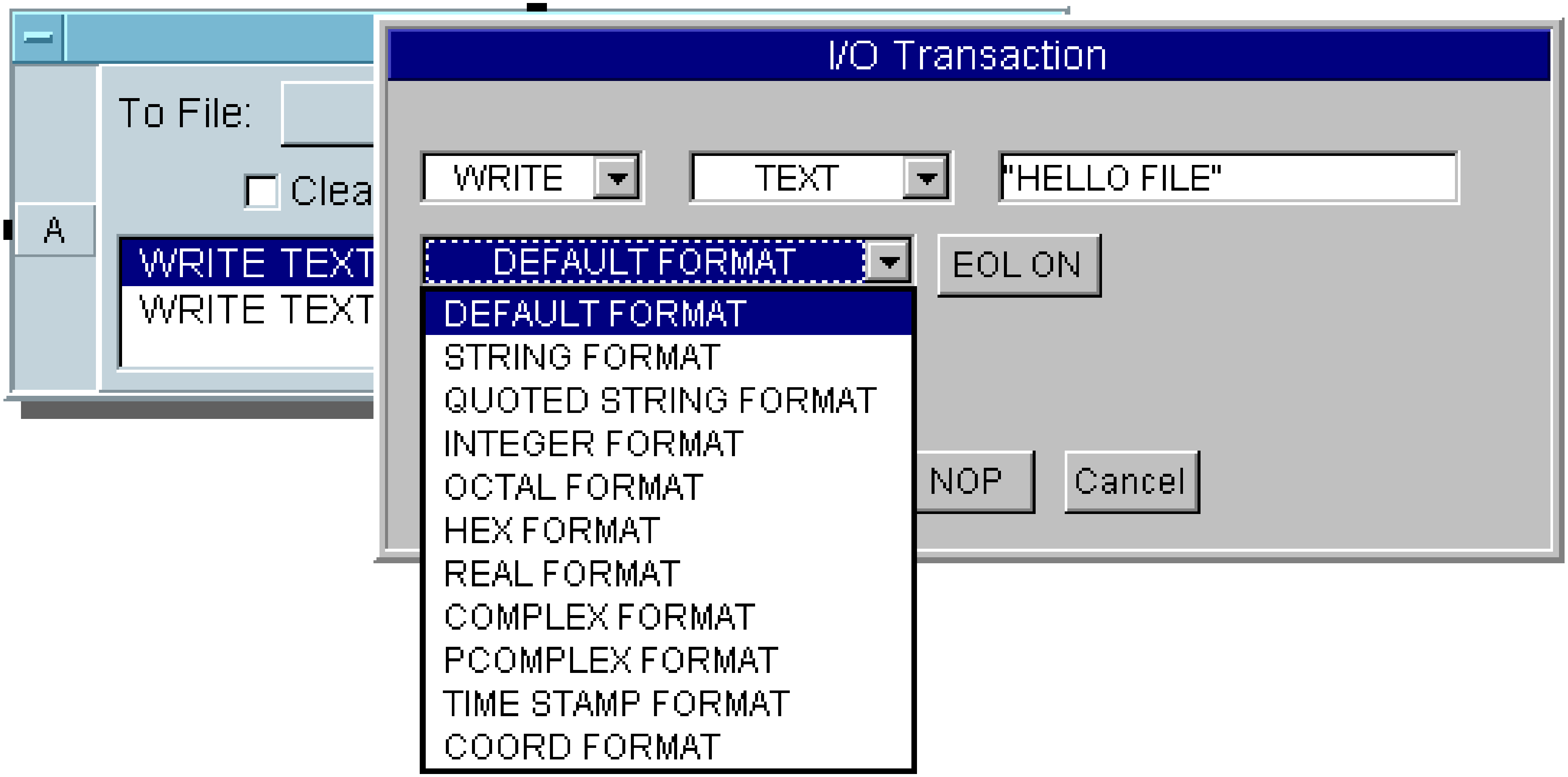
- COMPLEX - Expects two REAL values

- PCOMPLEX - As COMPLEX, except must specify RAD,
  DEG, GRAD to interpret angle

- COORD - Expects specified number of REAL
  values

Text Formats for WRITE

HP VEE



To File:

□ Clea

A

WRITE TEXT

WRITE TEXT

I/O Transaction

WRITE ▾ TEXT ▾ "HELLO FILE"

DEFAULT FORMAT ▾ EOL ON

- DEFAULT FORMAT
- STRING FORMAT
- QUOTED STRING FORMAT
- INTEGER FORMAT
- OCTAL FORMAT
- HEX FORMAT
- REAL FORMAT
- COMPLEX FORMAT
- PCOMPLEX FORMAT
- TIME STAMP FORMAT
- COORD FORMAT

HEWLETT PACKARD

Execute

To File and From File support EXECUTE commands

- REWIND
  - All further READ or WRITE operations start at beginning of file

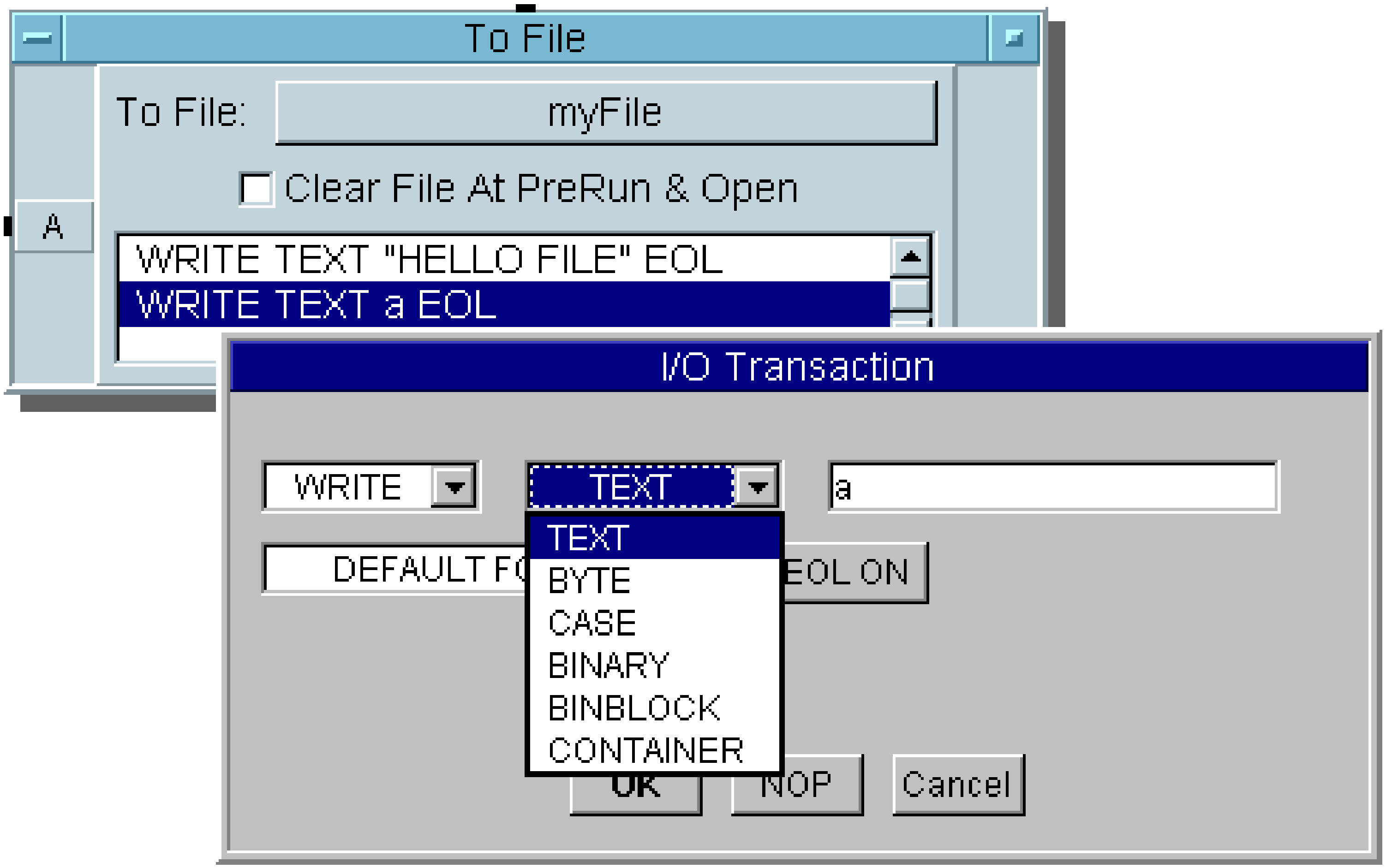
- CLEAR
  - Useful in OVERWRITE mode
  - Resets file to zero length (erases old data)

- CLOSE
  - Closes the file

- DELETE - Deletes the file

Data Encoding

HP VEE



To File

To File: myFile

□ Clear File At PreRun & Open

WRITE TEXT "HELLO FILE" EOL

WRITE TEXT a EOL

I/O Transaction

WRITE TEXT a

DEFAULT F[…] EOL ON

HEWLETT PACKARD

HP VEE

Data Encoding

- BINBLOCK - Data stream is sent as IEEE - 488.2 indefinite length block
  - A "#" character
  - A digit specifying the size of the length field
  - The length field specifying the number of bytes to follow

    ex: #12AB = a 1 digit length
    length = 2
    data = AB

    #2101234567890 = a 2 digit length
    length = 10
    data = 1234567890

HEWLETT
PACKARD

Data Encoding

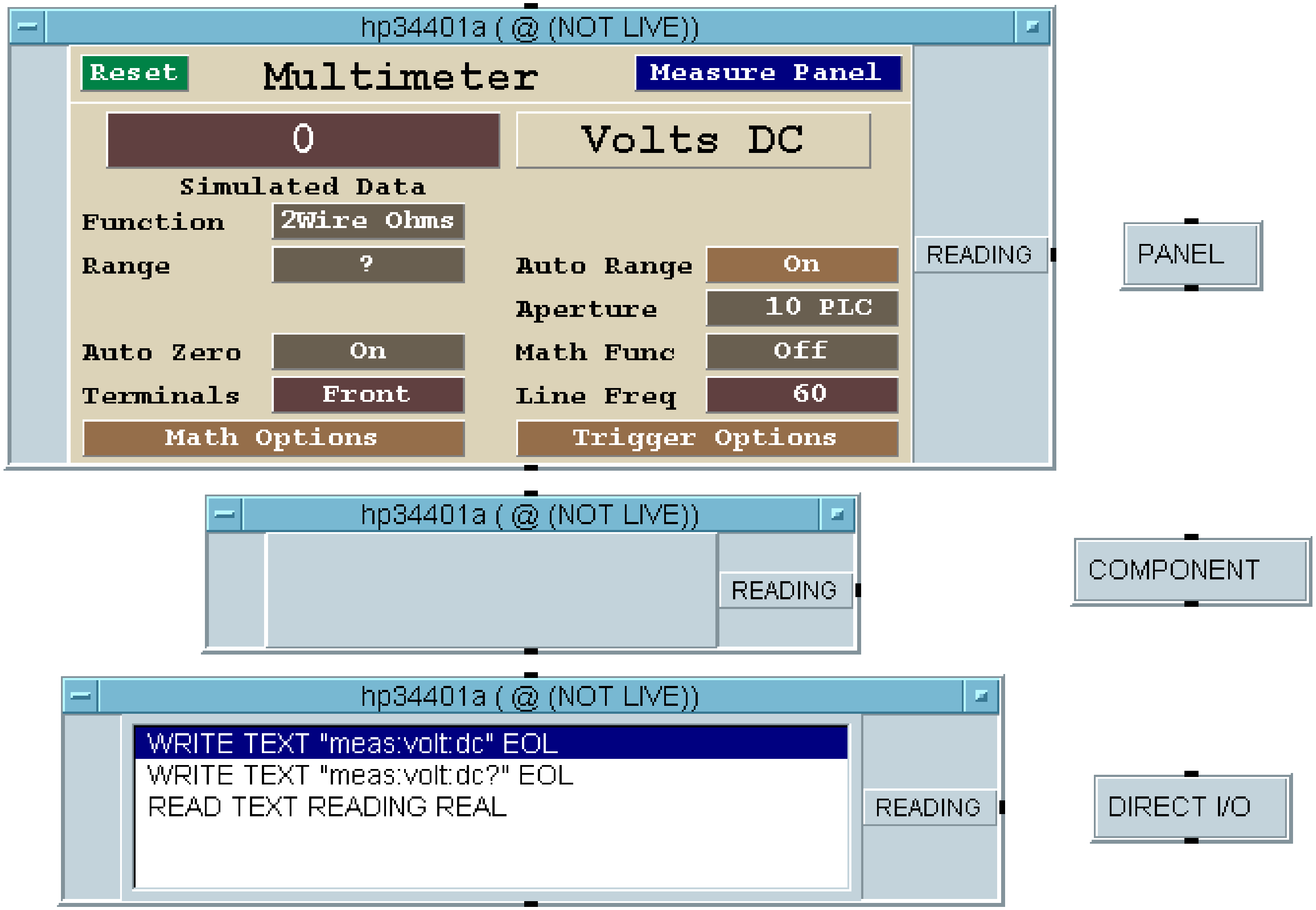
- CONTAINER - Data stream is sent to HP VEE internal format

```
ex:  ( INT 32
           ( numdims 1 )
           ( size 2 )
           ( data 1 2 )
     )
```

The Number Builder when Reading

- When numeric format is imposed on TEXT data stream, "number builder" attempts to extract numeric value from data
- Data is skipped while looking for numeric character
- Data is used by builder until EOL or non-numeric encountered
- Number is built
- Numeric means 0-7 for OCTAL
  0-9, a-f, A-F for HEX
  0-9 for INTEGER
  +, -, 0-9, e, E, decimal point for REAL

HP VEE Instrument I/O



hp34401a (@ (NOT LIVE))

Reset — Multimeter — Measure Panel

0 — Volts DC

Simulated Data

Function	2Wire Ohms		
Range	?	Auto Range	On
		Aperture	10 PLC
Auto Zero	On	Math Func	Off
Terminals	Front	Line Freq	60

Math Options — Trigger Options

READING

PANEL

hp34401a (@ (NOT LIVE))

READING

COMPONENT

hp34401a (@ (NOT LIVE))

```
WRITE TEXT "meas:volt:dc" EOL
WRITE TEXT "meas:volt:dc?" EOL
READ TEXT READING REAL
```

READING

DIRECT I/O

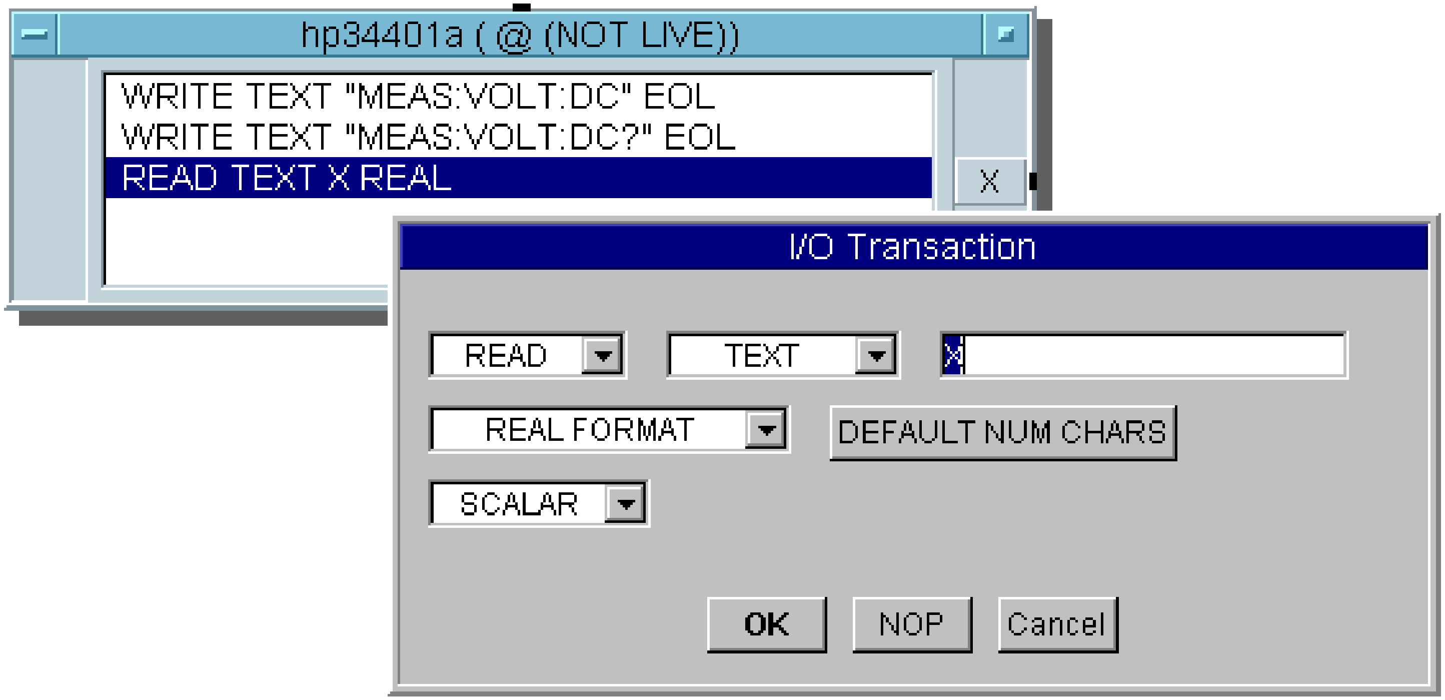
HP VEE Instrument I/O

- Direct I/O
  - Transaction interface consistent with other I/O transaction objects
  - Fast, flexible and powerful
  - For devices or instruments with no pre-developed drivers
- Panel "State" Drivers
  - Developed by HP for over 400 instruments
  - Easiest HP VEE instrument control
  - Most interactive
- Component Drivers
  - Allow efficient access to Panel components - relies on information in the "Panel" driver

Direct I/O

HP VEE

- Full instrument I/O functionality via transaction objects
  - READ and WRITE data in all formats
  - EXECUTE for control of interface and device
  - Wait



hp34401a (@ (NOT LIVE))

```
WRITE TEXT "MEAS:VOLT:DC" EOL
WRITE TEXT "MEAS:VOLT:DC?" EOL
READ TEXT X REAL
```

I/O Transaction

READ | TEXT | X

REAL FORMAT | DEFAULT NUM CHARS

SCALAR

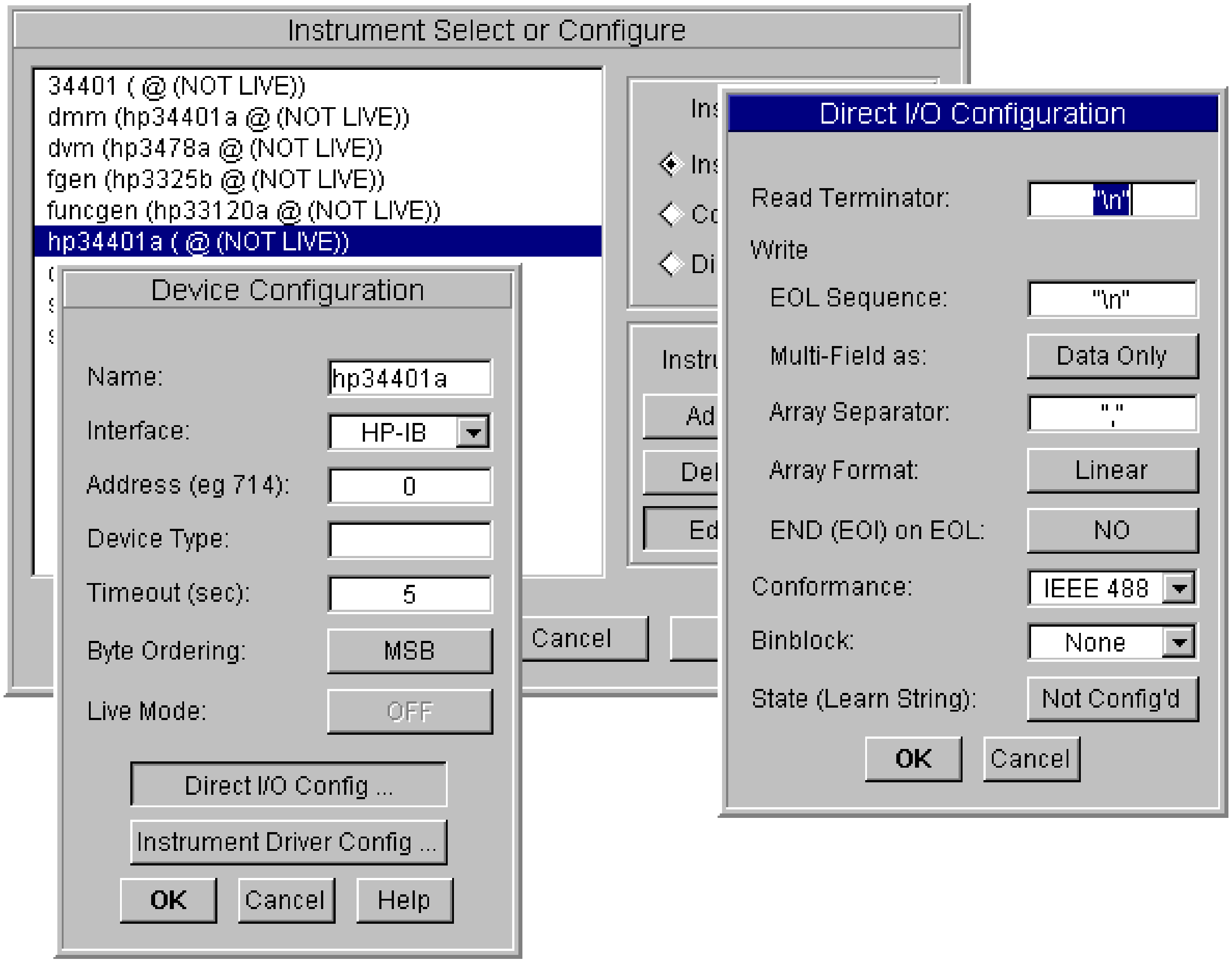
HEWLETT PACKARD

Trade Offs With Direct I/O Transactions

- Benefits
  - Highest performance I/O
  - Consistent usage with other I/O transactions
  - Used to access instrument functionality unavailable through instrument Panel Drivers
  - Access to registers of VXI register-based card
- Disadvantages
  - Requires familiarity with instrument programming
  - Time to create I/O transactions

HP VEE

Direct I/O Configure



Instrument Select or Configure

34401 (@ (NOT LIVE))
dmm (hp34401a @ (NOT LIVE))
dvm (hp3478a @ (NOT LIVE))
fgen (hp3325b @ (NOT LIVE))
funcgen (hp33120a @ (NOT LIVE))
hp34401a (@ (NOT LIVE))

Device Configuration

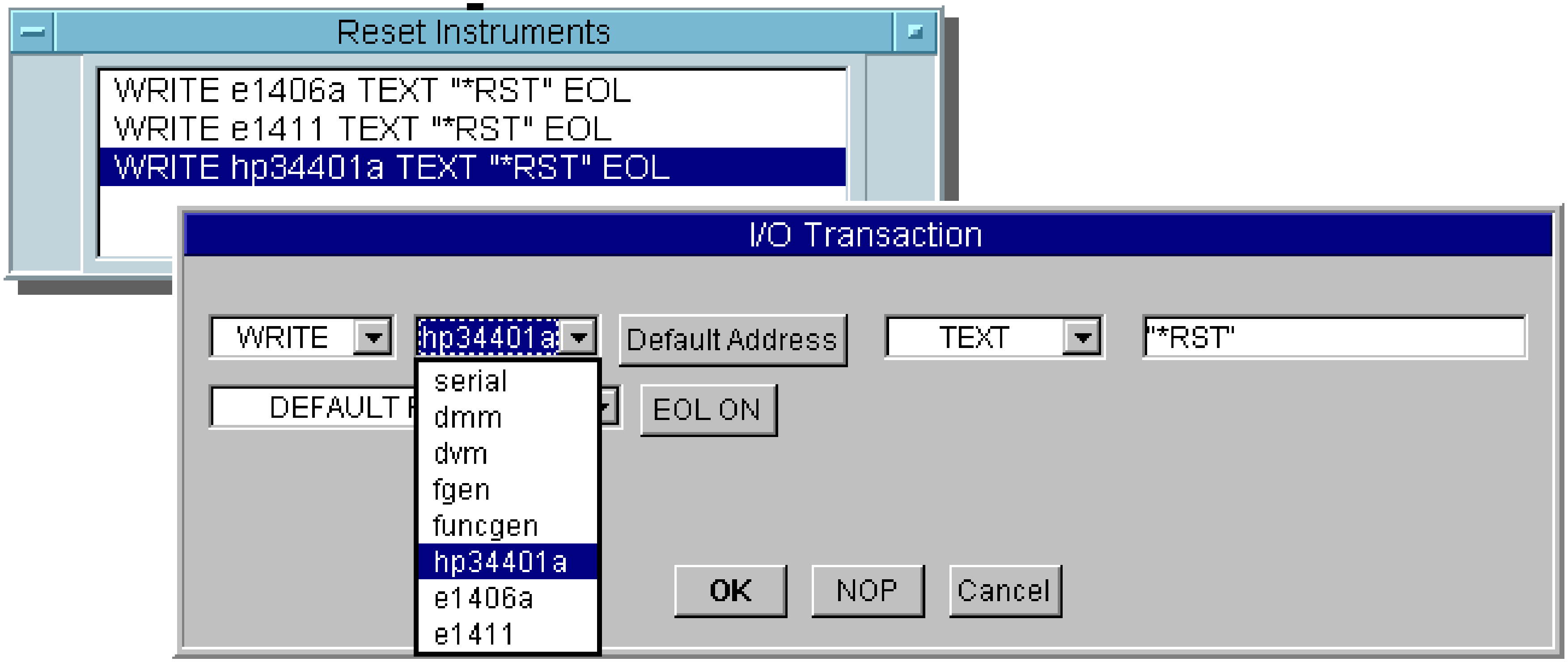
Name: hp34401a
Interface: HP-IB
Address (eg 714): 0
Device Type:
Timeout (sec): 5
Byte Ordering: MSB
Live Mode: OFF

Direct I/O Configuration

Read Terminator: "\n"
Write
EOL Sequence: "\n"
Multi-Field as: Data Only
Array Separator: " "
Array Format: Linear
END (EOI) on EOL: NO
Conformance: IEEE 488
Binblock: None
State (Learn String): Not Config'd

HEWLETT
PACKARD

MultiDevice Direct I/O



Reset Instruments

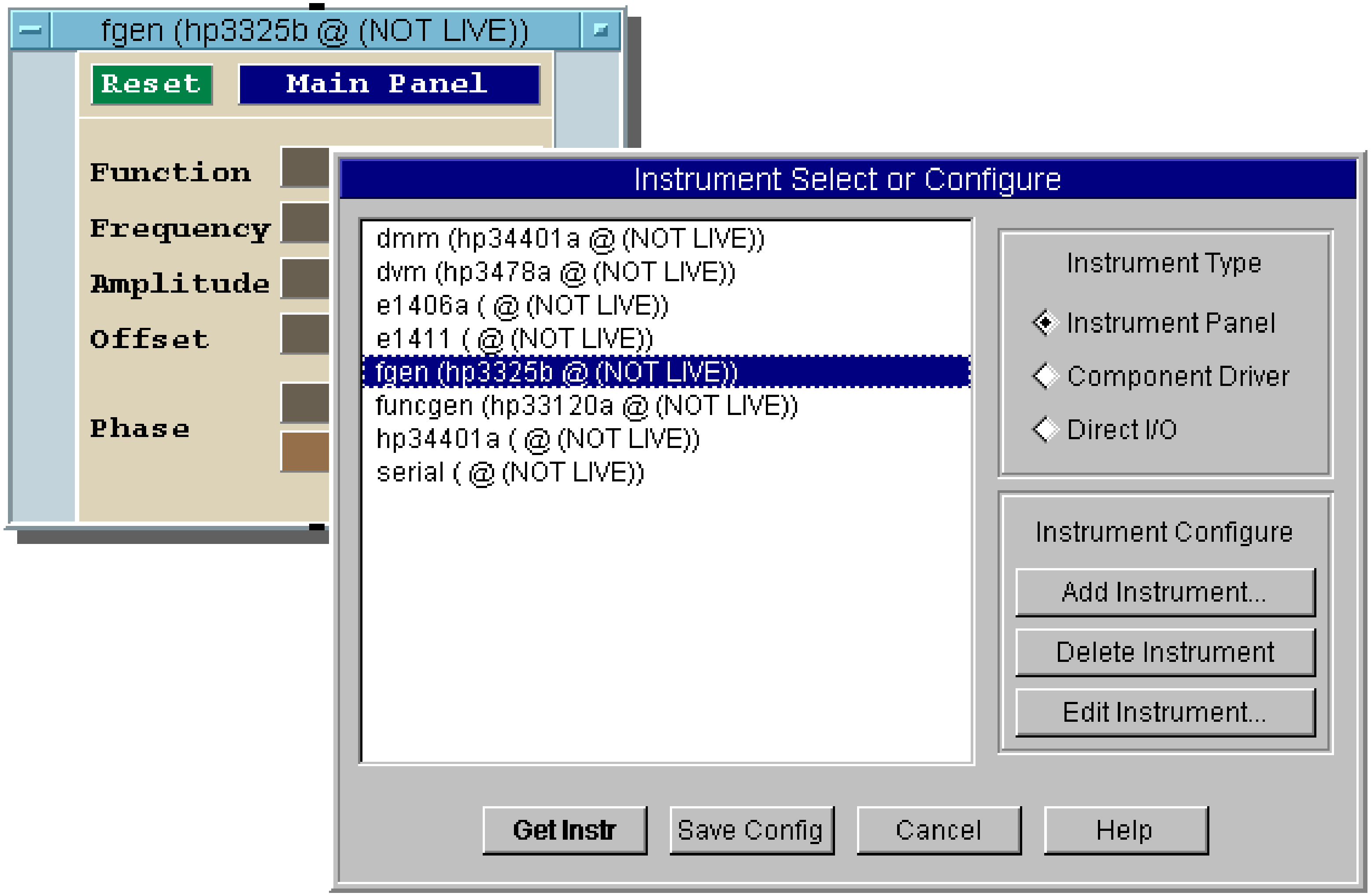
```
WRITE e1406a TEXT "*RST" EOL
WRITE e1411 TEXT "*RST" EOL
WRITE hp34401a TEXT "*RST" EOL
```

I/O Transaction

WRITE | hp34401a | Default Address | TEXT | "*RST"

DEFAULT F | EOL ON

Instrument Select or Configure



dmm (hp34401a @ (NOT LIVE))
dvm (hp3478a @ (NOT LIVE))
e1406a (@ (NOT LIVE))
e1411 (@ (NOT LIVE))
fgen (hp3325b @ (NOT LIVE))
funcgen (hp33120a @ (NOT LIVE))
hp34401a (@ (NOT LIVE))
serial (@ (NOT LIVE))

Instrument Type

● Instrument Panel
○ Component Driver
○ Direct I/O

HP Instrument Panel Drivers

- Text file that defines:
    1. Instrument components (or functions)
    2. Bus mnemonics to set components
    3. User interface for front panel interaction

- All drivers are compiled to allow fast loading

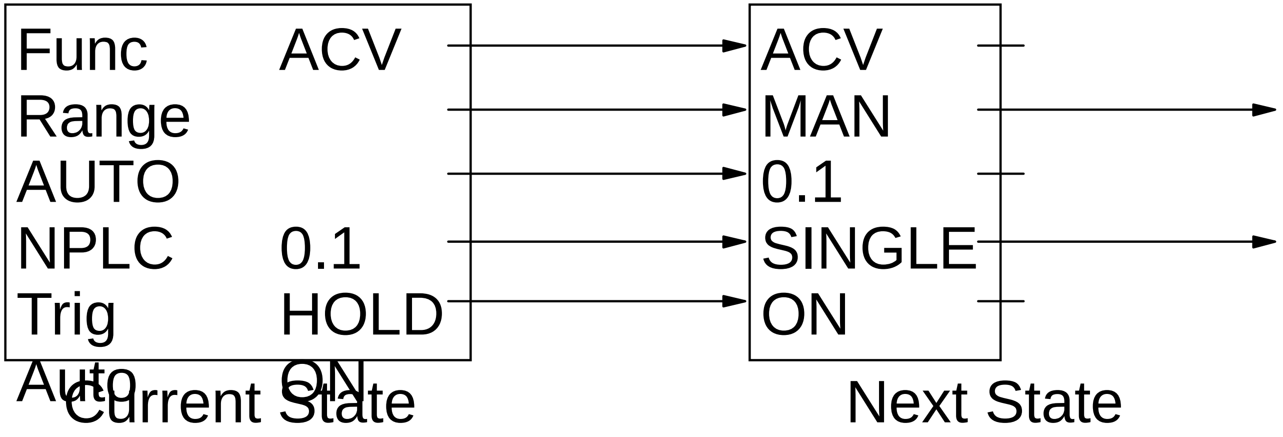
- Also contain function interrelation (coupling)

- HP instrument drivers provide access to most programmable functions available on the instrument

- Coupling allows incremental state programming

Incremental State Programming

- HP VEE maintains a state table of current instrument settings
- Users can request a single component or entire instrument state to be sent
- With Incremental Mode ON, only required components (commands) are sent to instrument



Func ACV
Range
AUTO
NPLC 0.1
Trig HOLD
Auto ON

Current State

ACV
MAN
0.1
SINGLE
ON

Next State

HP VEE Component Drivers

- Do not use graphical Panel interface
- Only required components are added to object
- Only added components have state maintained
- Because State Lookup is NOT done for every function, Component drivers execute much faster than State drivers

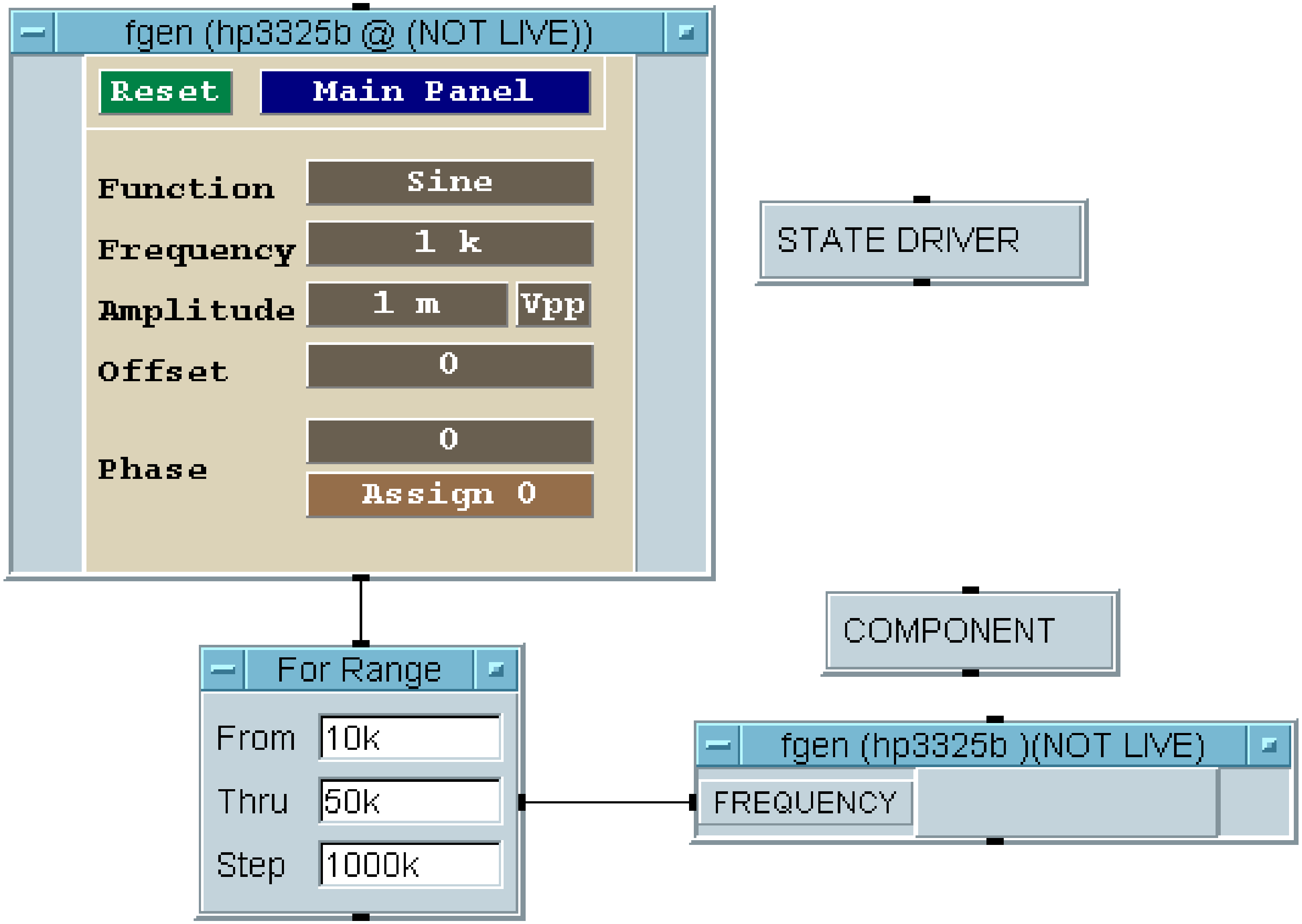
HP VEE Instrument Drivers

Summary

- State drivers for users wanting full graphical panels
- Component drivers to set/get specific components to optimize driver performance
- State drivers and Component drivers can be mixed and matched
- Multiple instances of same driver (state or component) to same instrument share state

HP VEE

State and Component Drivers



fgen (hp3325b @ (NOT LIVE))

Function: Sine

Frequency: 1 k

Amplitude: 1 m Vpp

Offset: 0

Phase: 0

STATE DRIVER

For Range

From: 10k

Thru: 50k

Step: 1000k

COMPONENT

fgen (hp3325b)(NOT LIVE)

FREQUENCY

HEWLETT PACKARD

Bus Monitor

- Works with HP-IB, GPIO, RS-232, VXI

- Records all traffic
  - Generated by Driver or Direct I/O
  - Received by Driver or Direct I/O

- Data is timestamped, displayed in text or hex, I/O direction indicated, and command bytes interpreted

  >VEE outbound traffic

  <VEE inbound traffic

- Includes a "TO FILE" and Buffer Size option

HP VEE Interfaces Supported

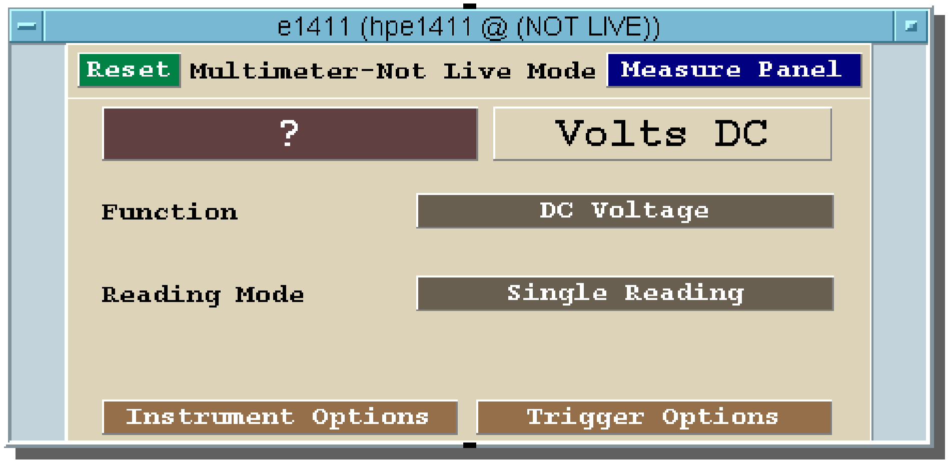
	HP-IB	RS-232	GPIO	VXI
PC	●	●		●
S300	●	●	●	●
S700	●	●	●	●
SUN	●	●		

VXI Support

V/382, V743, S700, EPC7, EPC8, and VXLink MXI Hardware

- Backplane access to message-based devices
  - ID’s and Direct I/O
- Backplane access to register-based devices
  - Direct I/O
  - ID’s with I-SCPI

VXI Register-Based Access



e1411 (hpe1411 @ (NOT LIVE))

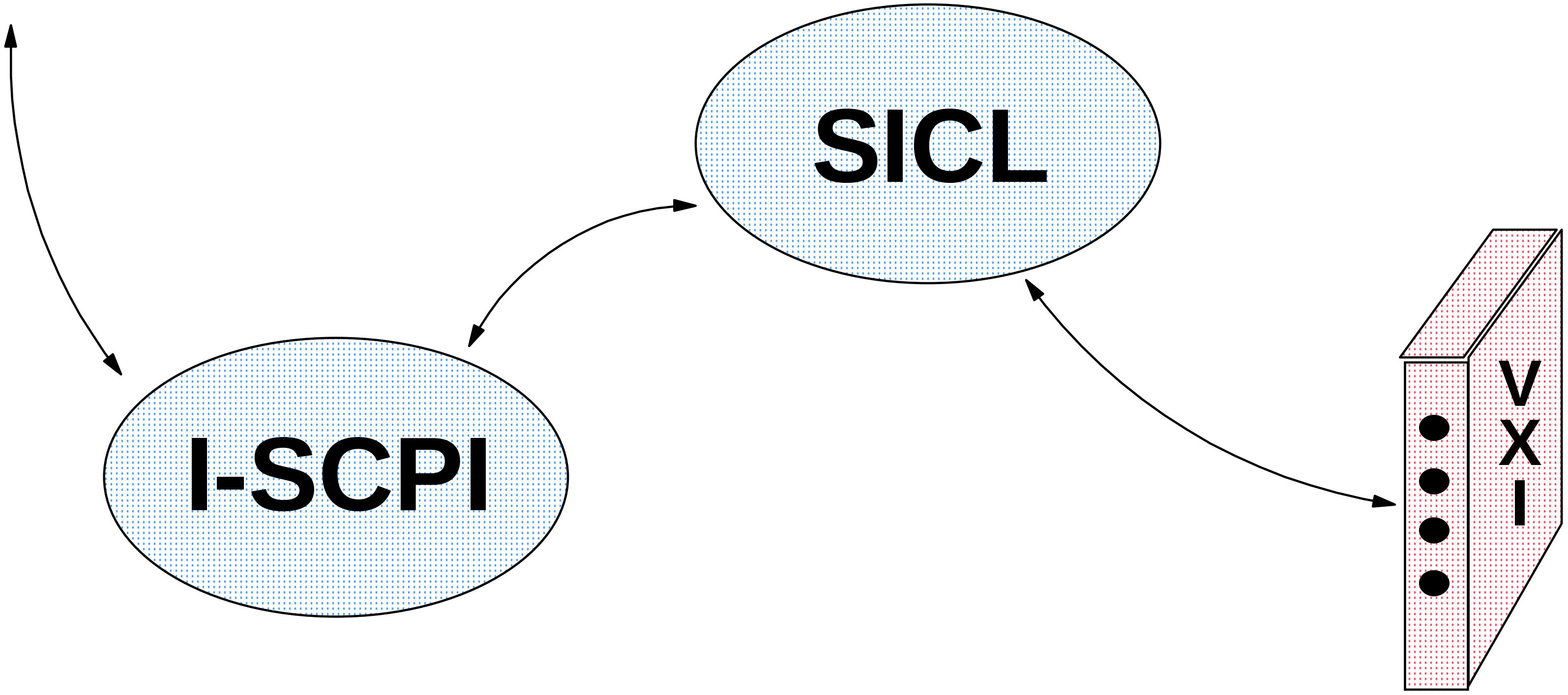
Reset | Multimeter-Not Live Mode | Measure Panel

? | Volts DC

Function: DC Voltage

Reading Mode: Single Reading

Instrument Options | Trigger Options

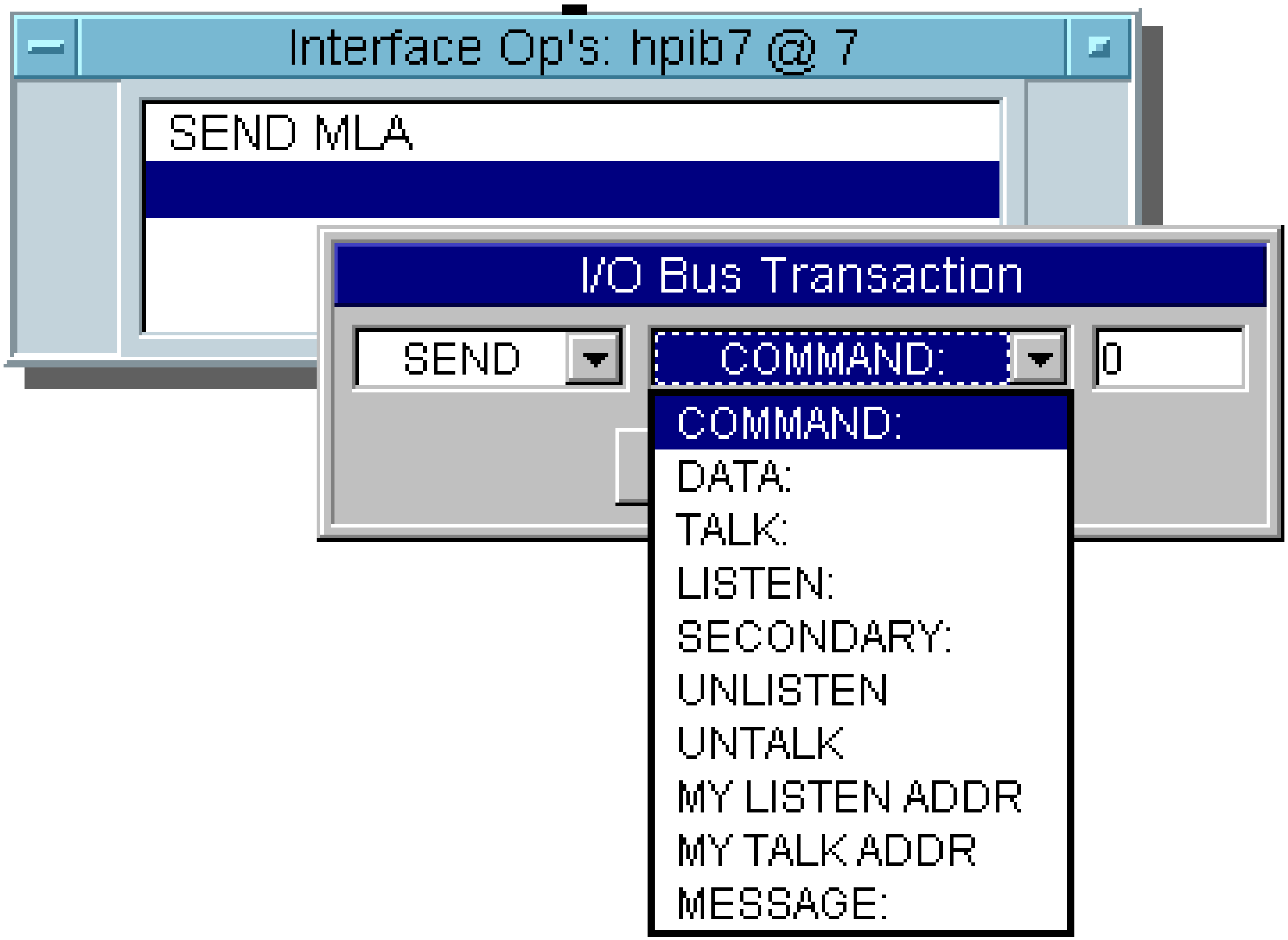


SICL

I-SCPI

VXI

Using Interface Operations



Interface Op's: hpib7 @ 7

SEND MLA

I/O Bus Transaction

SEND COMMAND: 0

Compiled Functions

- Develop specific functions or routines (data filters, etc.)
- Secure functions to protect unwanted access or proprietary technology
- Additional execution speed

Compiled Functions

- Allows you to link externally compiled functions directly into VEE
- Compiled functions are shared libraries or dynamic link libraries (DLL)
- Supported languages include C, FORTRAN, Pascal (S/700 and PC for Pascal)
- Compiled functions DO NOT allow you to access VEE internals
- Are imported and called the same as UserFunctions

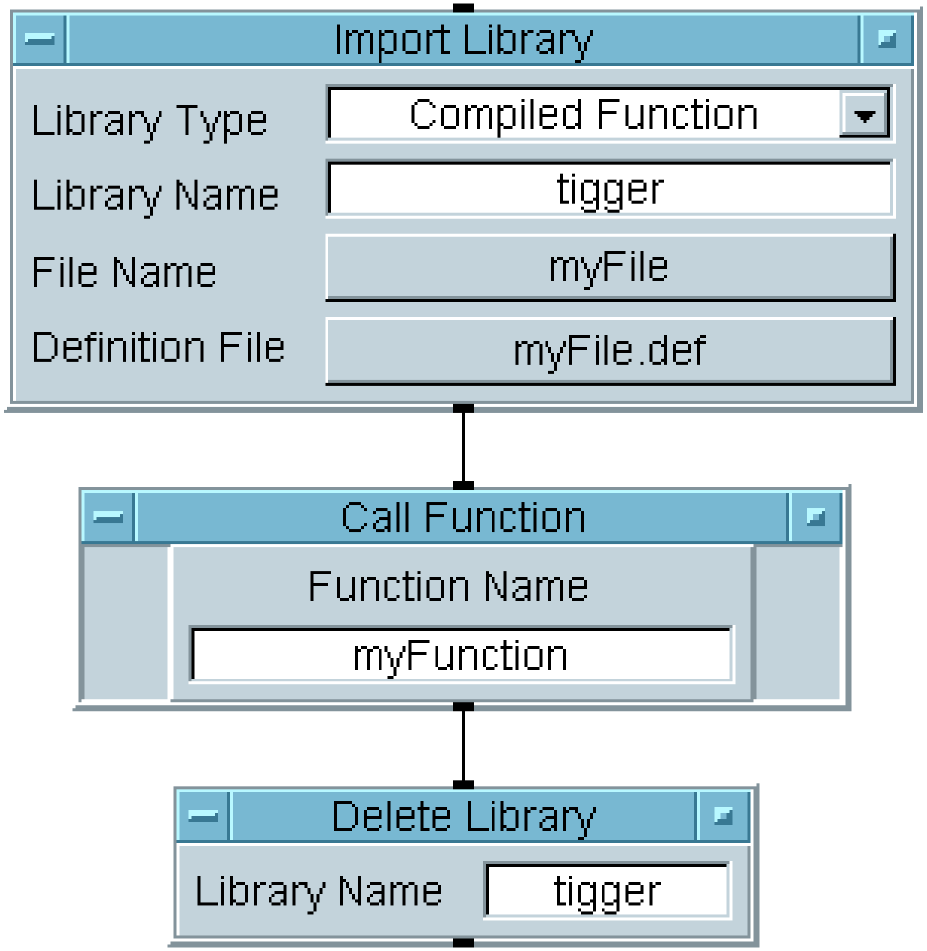
Pitfalls

- If your routine crashes, so does VEE
- Make sure your variables are of the correct type
- Free any dynamically allocated memory your routine creates and uses
- If you are working with arrays:
  - Pass the size of the array to your routine separately from the array itself
  - Use the return value to indicate the new size of the array you passed into your routine

Compiled Functions

Remember:
Load Lib choice lets you manually link in the C function

Remember:
"Select Function" lets you select the Function and its terminals. The Function will only show if you have IMPORTED it



Import Library

Library Type: Compiled Function

Library Name: tigger

File Name: myFile

Definition File: myFile.def

Call Function

Function Name: myFunction

Delete Library

Library Name: tigger

Compiled Functions vs. Execute Program

(HP-UX Escape)

+ Short start-up time
+ Communicates by passing data on the stack

- Shares memory space with VEE
- Runs synchronously
- Should not block or catch signals

+ Has protected memory space
+ Can service asynchronous events

- Much longer start-up time
- Communicates through pipes, sockets, DDE

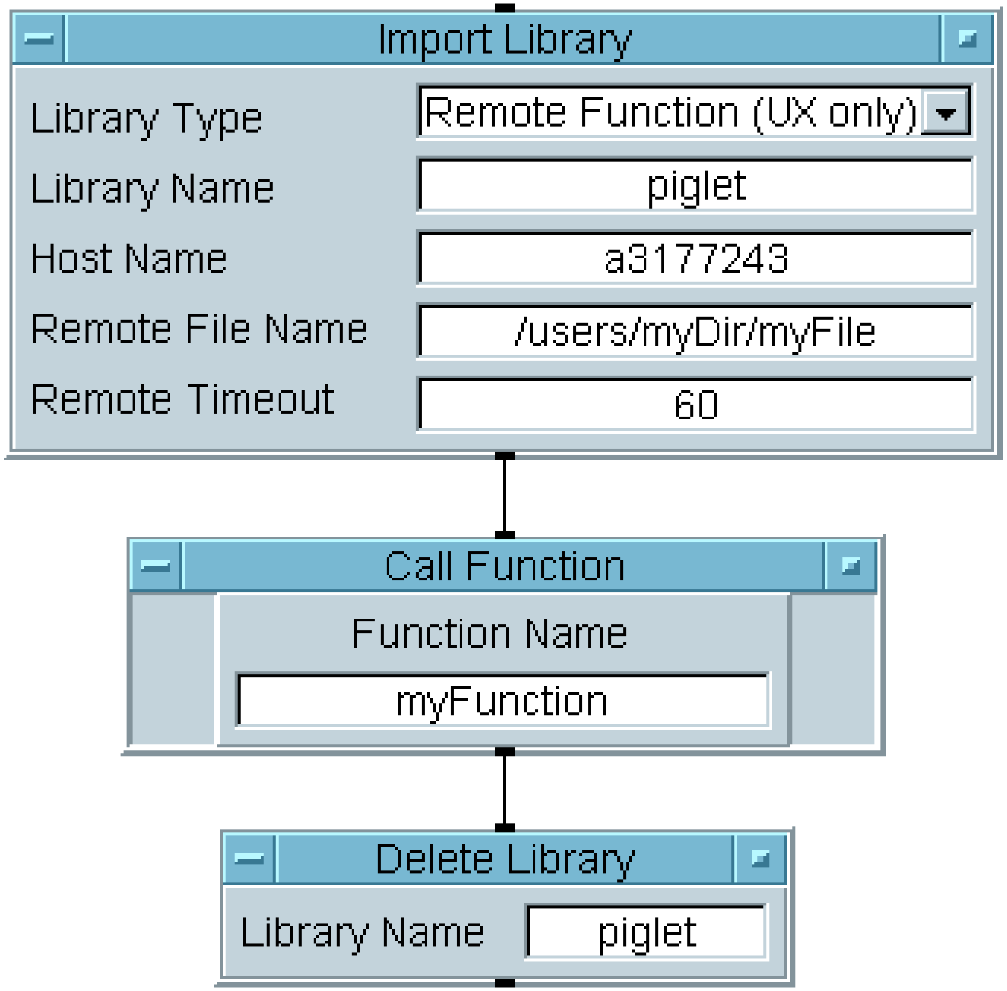
Bottom Line

- Compiled functions allow fast, direct access to your C functions; however, if you don’t understand using functions in C or you are integrating large, slow programs and applications, use execute program instead.

Remote Functions

Load Lib will confirm the Host is there, start the service manager process, load VEE and the file

Select Function will show the functions available on that file



Import Library

Library Type: Remote Function (UX only)

Library Name: piglet

Host Name: a3177243

Remote File Name: /users/myDir/myFile

Remote Timeout: 60

Call Function

Function Name: myFunction

Delete Library

Library Name: piglet

Remote Function (continued)

- Use the same import scheme as User and Compiled Function
- Allow you to access the resources of another workstation as though they were local resources
- Allow you to create distributed environments
- Allow you to create I/O servers

Remote Functions (continued)

- Import object starts VEE process on remote host
  - Does not create a window or an icon
  - X windows need not be running on remote host
- Import object pauses the local program until connection is established
- Multiple Import objects that call the same host and the same file will only start one VEE process on remote host

Execute Program/Execute Program PC

- Allows use of HP-UX commands and other programs
  - Reusability of existing code
  - Optimized routines
  - System information

- Data can be sent to and received from Execute Program (HP-UX only)
  - Similar to To/From Stdio (HP-UX only)
  - Execute Program is child process of HP VEE (HP-UX only)
  - Child receives data via its stdin, sends data via stdout and stderr (HP-UX only)
  - PC can use Files, DDE, etc.

Wait for Child Exit

- YES:
  - New process starts whenever Execute Program activates
  - VEE executes transactions, sends EOF (by closing pipe) (HP-UX only)
  - VEE waits for process termination
  - Program MUST terminate for VEE HANGS!!

- NO:
  - Process is allowed to remain active after Execute Program completes
  - Repeated Execute Programs do not need to restart process
  - Process must be designed to cooperate with Execute Program
    - Continuous loop
    - No unexpected terminations
  - Process will be restarted as needed after Pre-Run

Interprocess Communication

- Multiple HP-UX processes to work in concert on a single problem
  - Individual processes are less complex
  - Individual processes may be optimized for task
- Benefit:
  - Complex systems built from less-complex modules
  - Less coupling means easier maintenance
  - Re-use existing programs

IPC Facilities

- HP VEE implements
  - Ordinary files
  - Pipes (HP-UX only)
  - Sockets
  - DDE (PC only)

- Other methods are very specialized - Access via HP-UX Escape object if required
  - Shared memory
  - Semaphores

Using Pipes for IPC (HP-UX Only)

- Pipes enforce FIFO message order
  - Multiple processes may write or read
  - Data can be read once ONLY

- Arbitration for multiple readers on pipe

- Pipes must exist locally (not NFS mounted)

Using Named Pipes

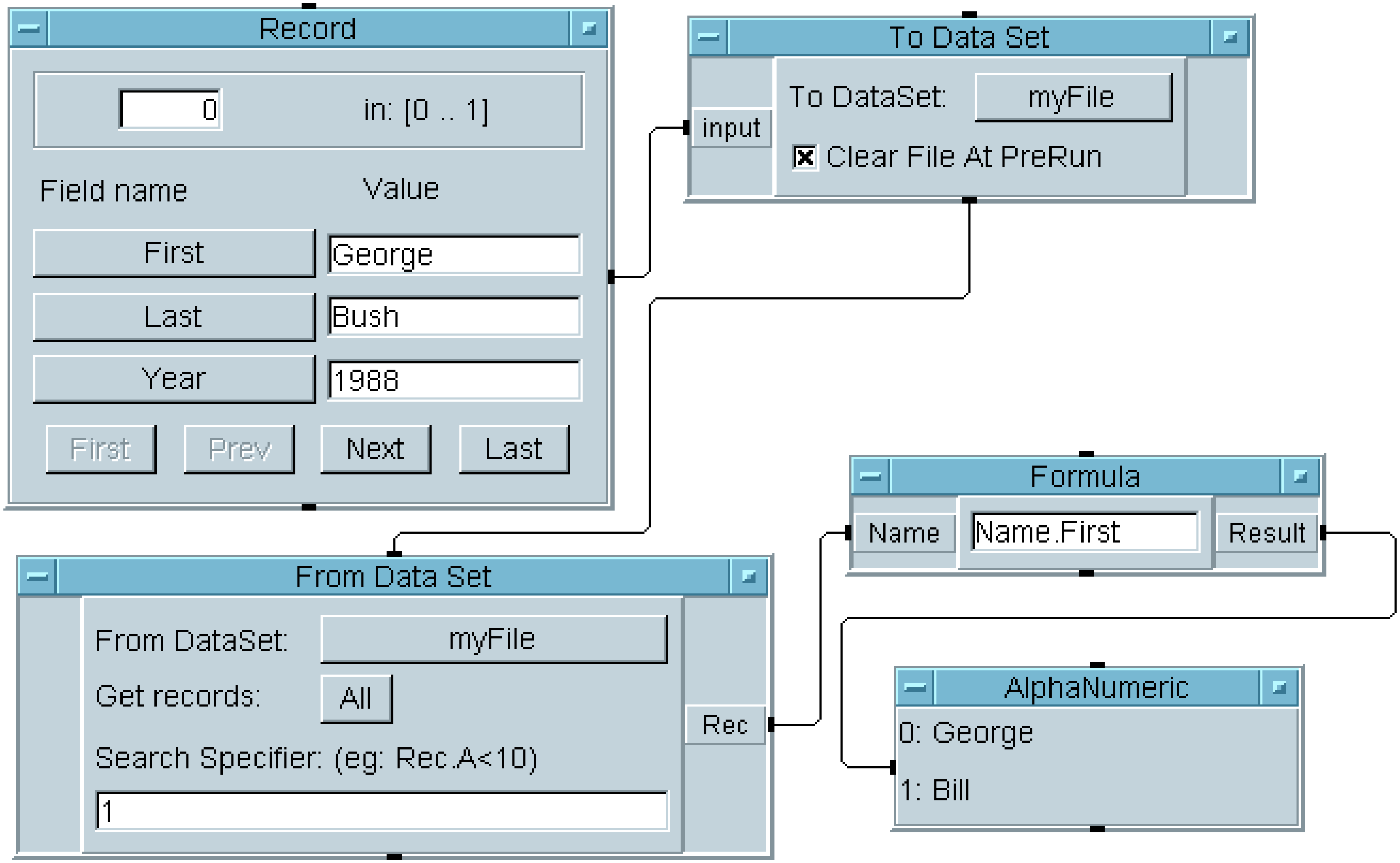
- Capacity of pipes is limited (4K - 8K typical)
  - Writing to full pipe "blocks" writer
  - Reading from empty pipe "blocks" reader
- Synchronizing is reliable if only one each reader/writer
  - Kernel suspends processes until both reader and writer exist
  - Blocking will synchronize later if needed

To/From Named Pipes

- Pipes are automatically created by first attempt to open
  - Read pipe opened read-only
    - Allows EOF detection
  - Write pipe opened write-only
- Pipes are closed upon termination of entire program
  - Not after each object deactivates
  - Never deleted
- Pipes opened as "blocking", but you can get the status to find out if data’s available
  - Needed for synchronization
  - Can hang waiting for data or space available

HP VEE

HP VEE Records and Datasets



Record

0 in: [0 .. 1]

Field name Value

First: George

Last: Bush

Year: 1988

To Data Set

To DataSet: myFile

☒ Clear File At PreRun

From Data Set

From DataSet: myFile

Get records: All

Search Specifier: (eg: Rec.A<10)

1

Formula

Name.First

AlphaNumeric

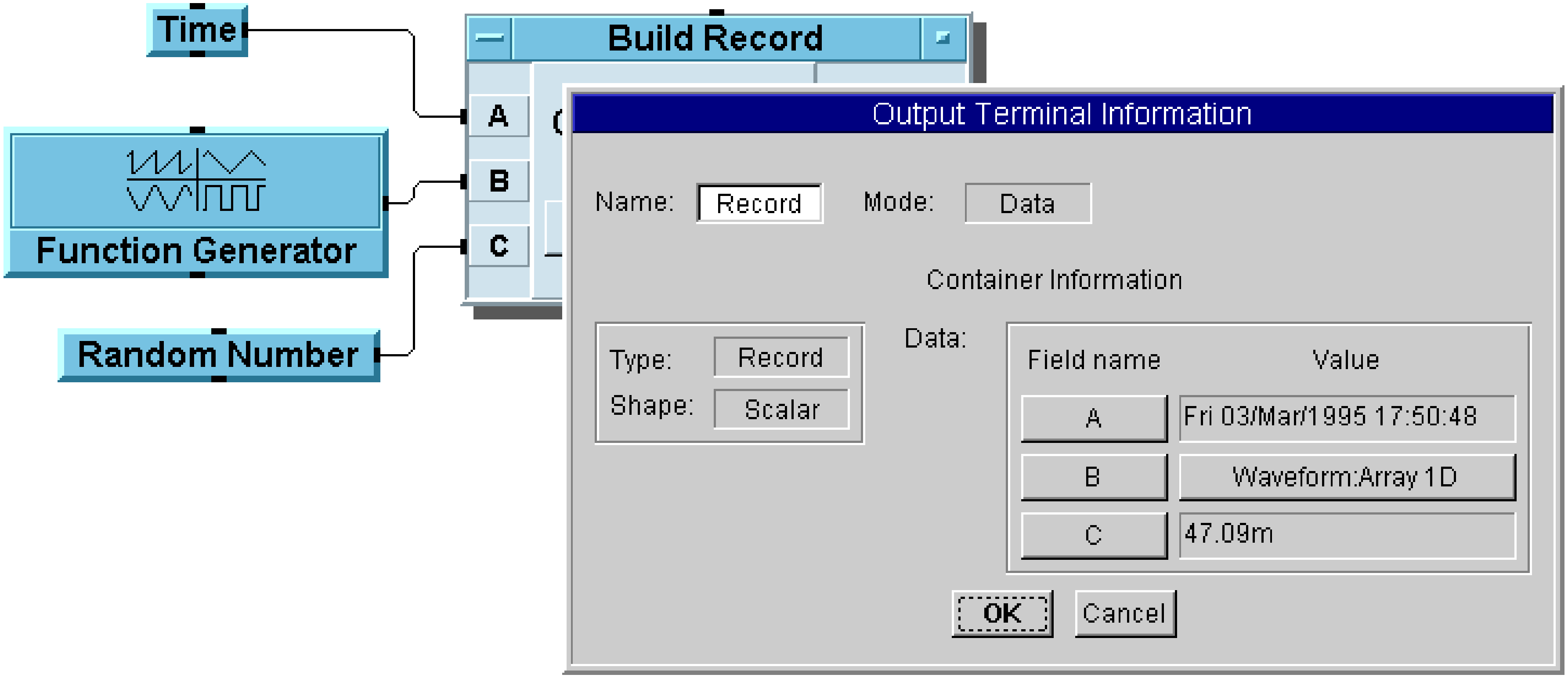
0: George

1: Bill

HEWLETT PACKARD

Records

- Reduce the number of data lines required
- Package related information
- Serve as a building block for personal data management



Output Terminal Information

Name: Record Mode: Data

Container Information

Type: Record
Shape: Scalar

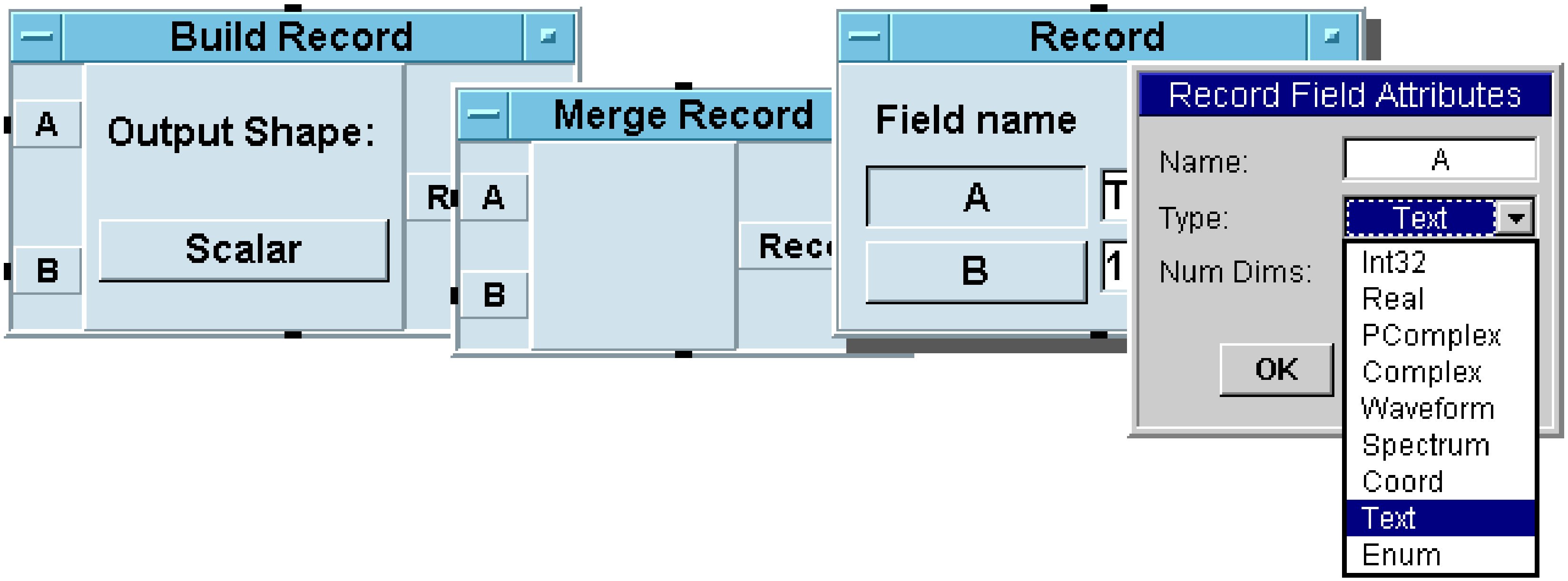
Data:

Field name	Value
A	Fri 03/Mar/1995 17:50:48
B	Waveform:Array 1D
C	47.09m

Records

Build 3 Ways

- Record Constant
- Data ==> Build Data ==> Record
- Merge Record



Build Record

Output Shape: Scalar

Merge Record

Record

Field name

Record Field Attributes

Name: A

Type: Text

Num Dims:

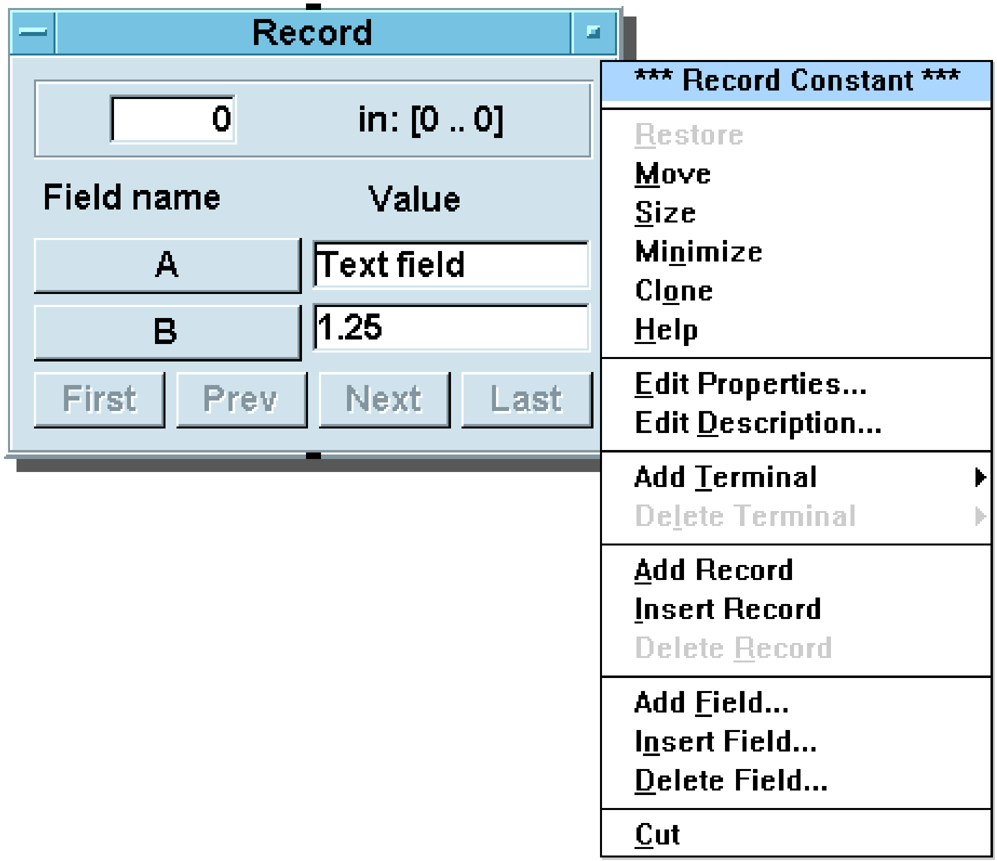
Record Constant

- Default is 2 field - A and B

- Field name is VERY important - that is how the data is modified or accessed

- Field type could include Real, Integer, Text or Enum

- Object menu includes Add, Insert, or Delete Fields



Record

0 in: [0 .. 0]

Field name	Value
A	Text field
B	1.25

Record Constant - Config

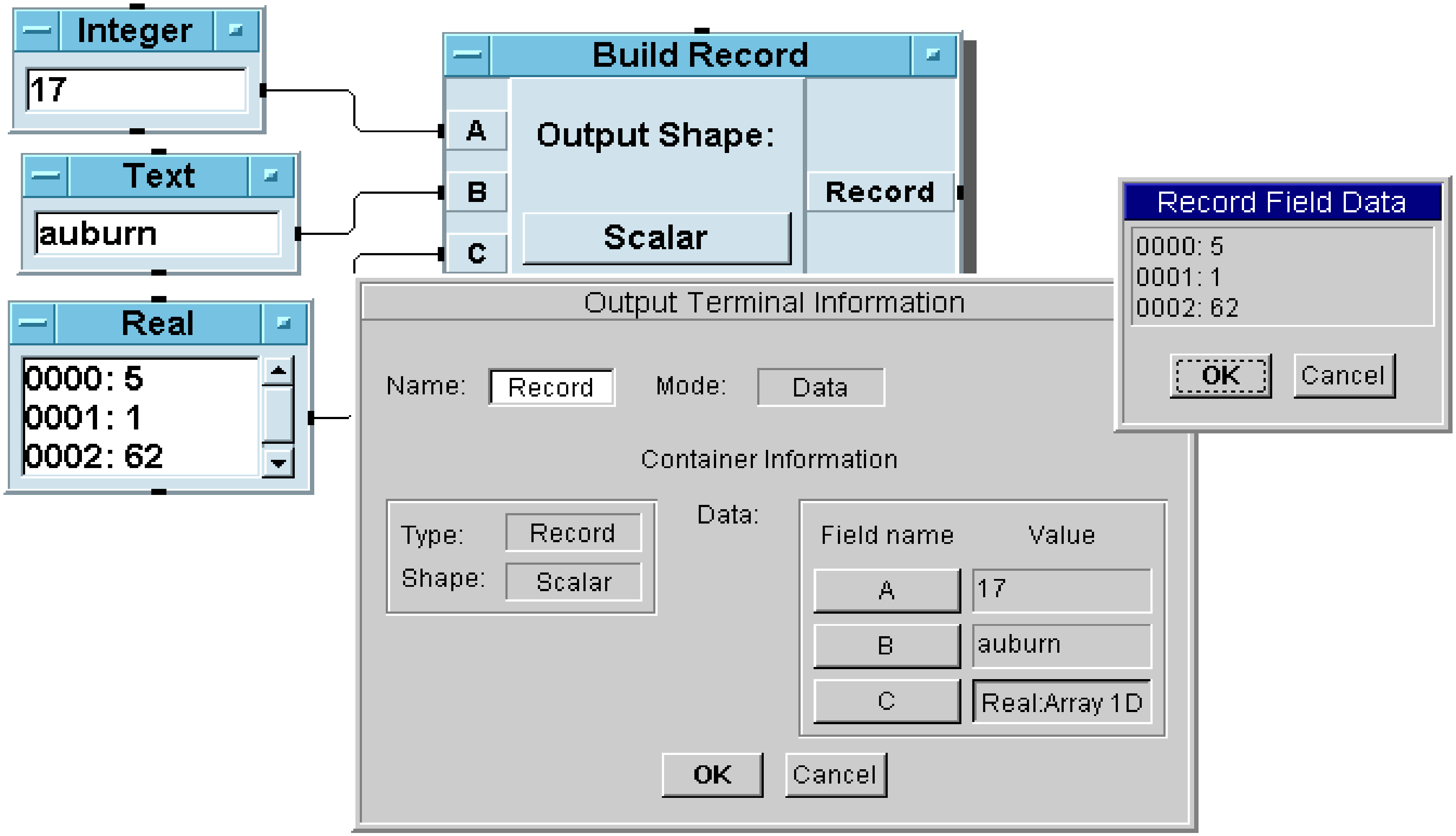
- Edit Properites... - Configuration
  - scalar/1D Array
  - size (number of elements if array)
  - size fixed
  - schema fixed

- Size Fixed assures the number of elements is not inadvertently changed and maybe breaks a program

- Schema Fixed locks the Field Name and Type from any changes

Build Record As Scalar

Terminal names on Build Record are the field names for the record



Integer: 17

Text: auburn

Real:
0000: 5
0001: 1
0002: 62

Build Record
A
B
C
Output Shape: Scalar
Record

Output Terminal Information

Name: Record
Mode: Data

Container Information

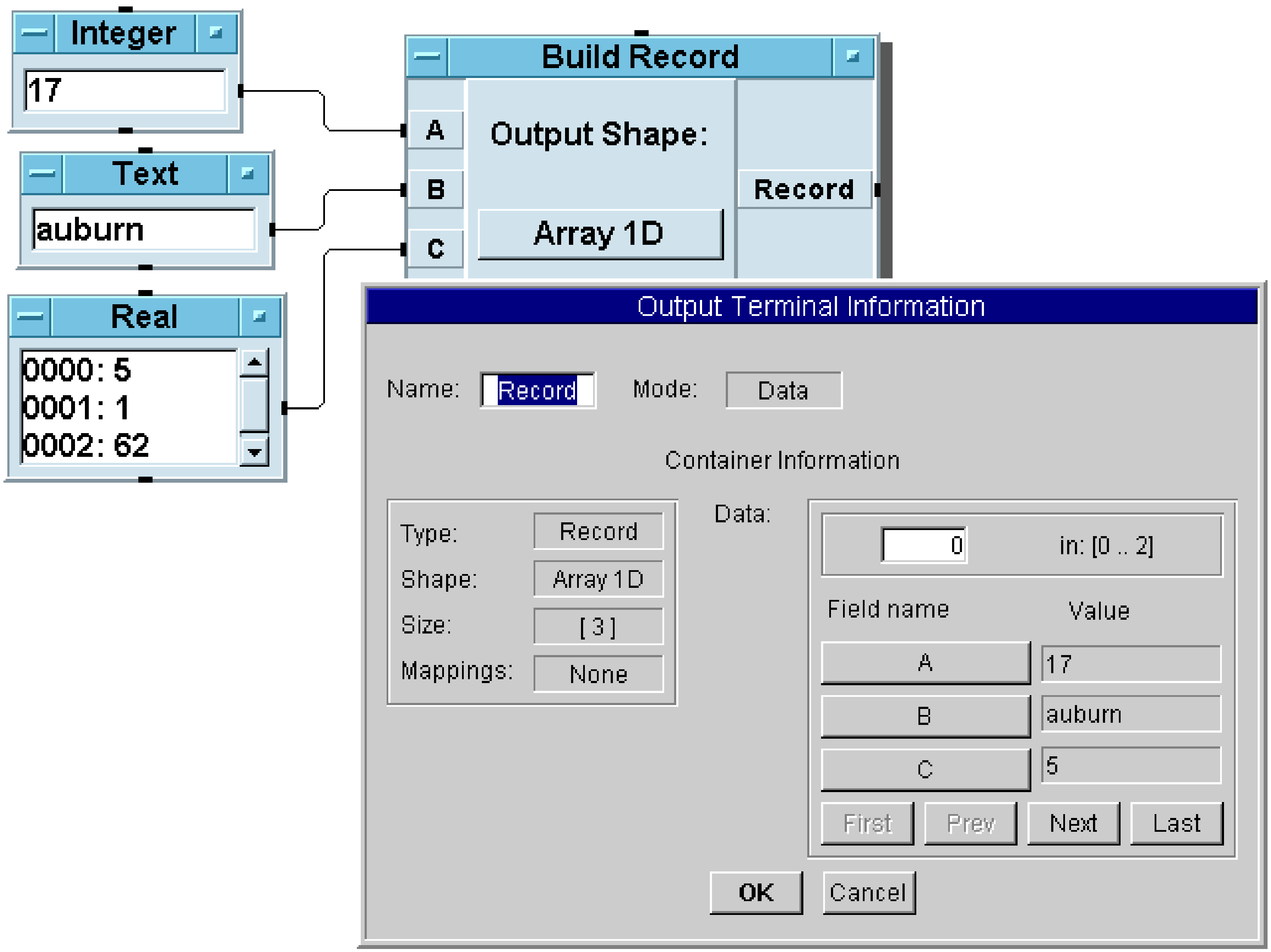
Type: Record
Shape: Scalar

Data:

Field name	Value
A	17
B	auburn
C	Real:Array 1D

Record Field Data
0000: 5
0001: 1
0002: 62

Build Record As Array



Integer: 17

Text: auburn

Real:

```
0000: 5
0001: 1
0002: 62
```

Output Shape: Array 1D

Name: Record Mode: Data

Type:	Record
Shape:	Array 1D
Size:	[3]
Mappings:	None

0 in: [0 .. 2]

Field name	Value
A	17
B	auburn
C	5

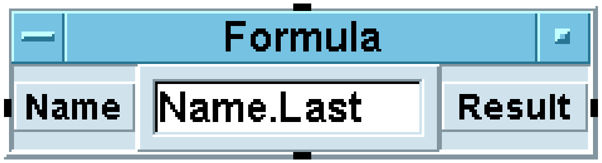
Merge Record

- Building Records: Data ==> Access Record ==> Merge Record

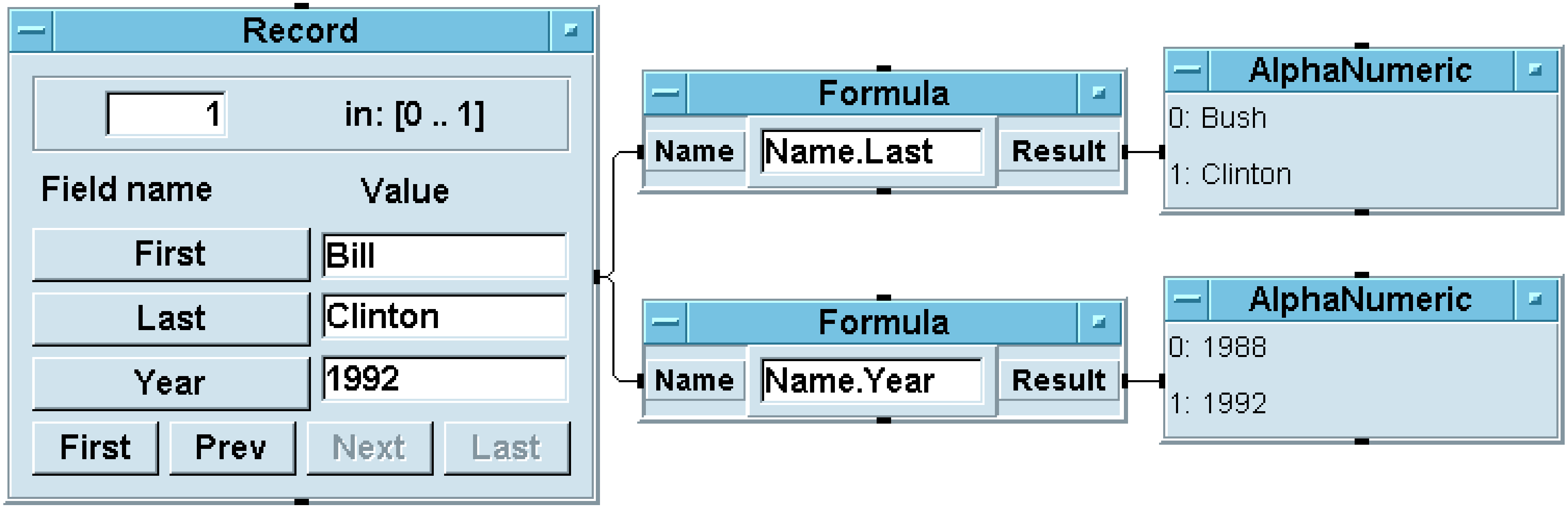
- Merge Record
  - Allows you to create an aggregate record from two or more individual records
  - The names associated with the Data In pins are ignored
    - One caution - suppose you have two Build Record objects, and you have kept the default name for both sets of data in pins?
    - Result - HP VEE would give you an error, because each field in a record must have a unique name

Records in Expressions

- Instead of UnBuilding Data - use formulas!



Name.Last



1 in: [0 .. 1]

Field name	Value
First	Bill
Last	Clinton
Year	1992

Name.Last

0: Bush
1: Clinton

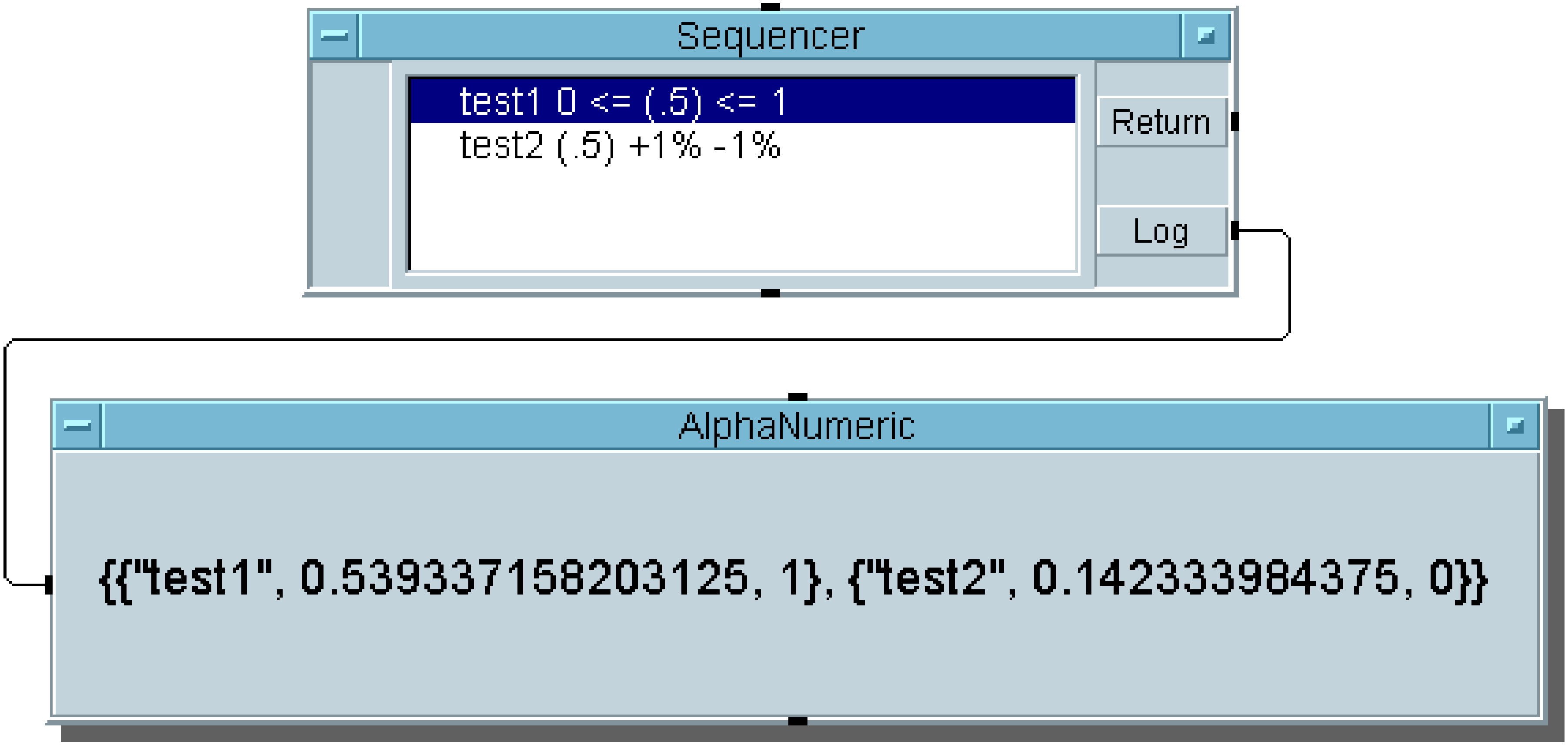
Name.Year

0: 1988
1: 1992

Data Sets

- File based array of records
- Allows you to easily store records or search through many stored records to retrieve specific records that match your criteria
- All records must have the same size and shape

HP VEE Sequencer



```
test1 0 <= (.5) <= 1
test2 (.5) +1% -1%
```

```
{{"test1", 0.539337158203125, 1}, {"test2", 0.142333984375, 0}}
```

HP VEE Sequencer

- Allows developer to:
  - Specify order of execution
  - Branch based on results

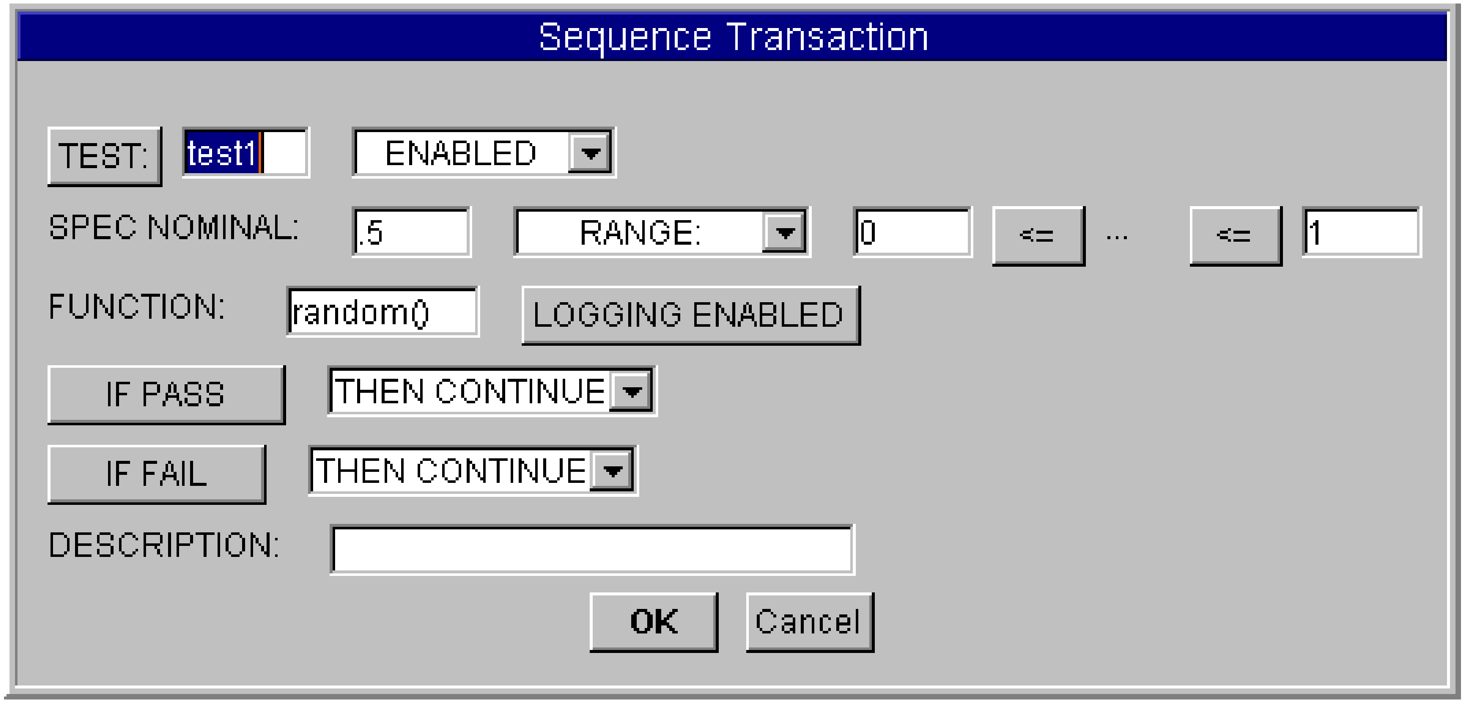
- During execution, the sequencer can:
  - Run pre-defined sequences
  - Interactively modify execution sequence (loop, retest, continue, stop, goto, etc.)

- Also includes:
  - Access to other sequencers for hierarchical structure
  - Logging of sequencer actions

- Can create a status panel and update it as the various tests are run

Sequencer

- Fundamental building block for a Test Executive
- Transaction-based Object
- Can call User Functions, Compiled Functions, Remote Functions or Expressions
- Compares values from the functions against your TEST SPEC (the same as a comparator object)
- Selects next transaction based on test results



Sequence Transaction

TEST: test1 ENABLED

SPEC NOMINAL: .5 RANGE: 0 <= ... <= 1

FUNCTION: random() LOGGING ENABLED

IF PASS THEN CONTINUE

IF FAIL THEN CONTINUE

DESCRIPTION:

Rules

- Transaction name must be unique
- Expression fields allow functions, globals, record math, terminals, etc.

Sequencer - Enabled Field

- Determines whether this step should run or be skipped
- Allows full expression evaluation
- Logs all 0s if test is not run
- Can test results from a previous test if logging is configured for that field

Sequencer - Conditions

• Then continue	execute next logical step
• Then return send the	stop at this step - fire output pins and specified return value
• Then goto	jump to the specified transaction
• Then repeat	run the test this # of times
• Then error	generate an error - don’t fire other pins
• Then eval step	evaluate a returned string to decide next

Debugging Sequencer Tests

- "SHOW ON EXEC" will show which transaction is currently executing
- Object Menu ==> step trans (ctl X) will execute 1 step at a time
- Edit ==> User Function and Edit ==> View Globals may be used to observe what is occurring in the functions that are being referenced

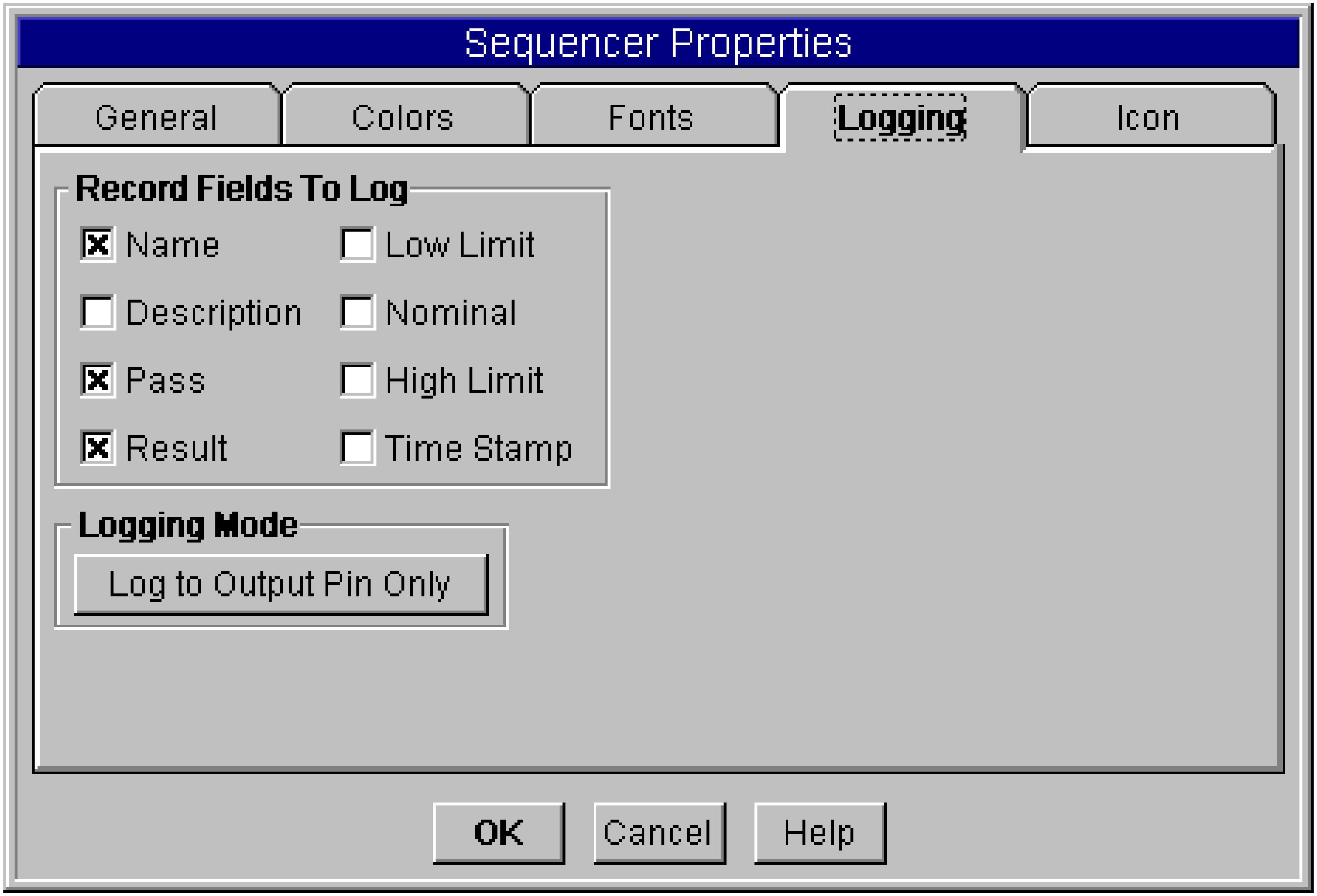
EXEC TRANS

- Data Pin to specify which test to execute
- Accepts text string or array or strings containing the test name
- Ignores Pass/Fail/Enabled conditions

Sequencer - Logging

- Each transaction logs a record
- Transactions don’t log if:
  - it is an EXEC transaction
  - logging is disabled for that transaction
- Each transaction is temporarily logged to "thistest" for use in expressions
- After the sequencer completes - output terminal.log will send a RECORD of RECORDS!
- The output data will only be from the last time the transaction executed

Sequencer - Logging Configuration



Record Fields To Log

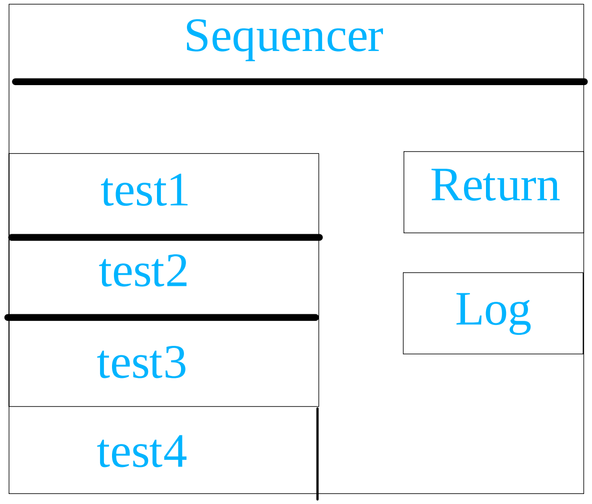
☒ Name	□ Low Limit
□ Description	□ Nominal
☒ Pass	□ High Limit
☒ Result	□ Time Stamp

Logging Mode

Log to Output Pin Only

HP VEE

Sequencer Logging After One Execution



Sequencer

test1

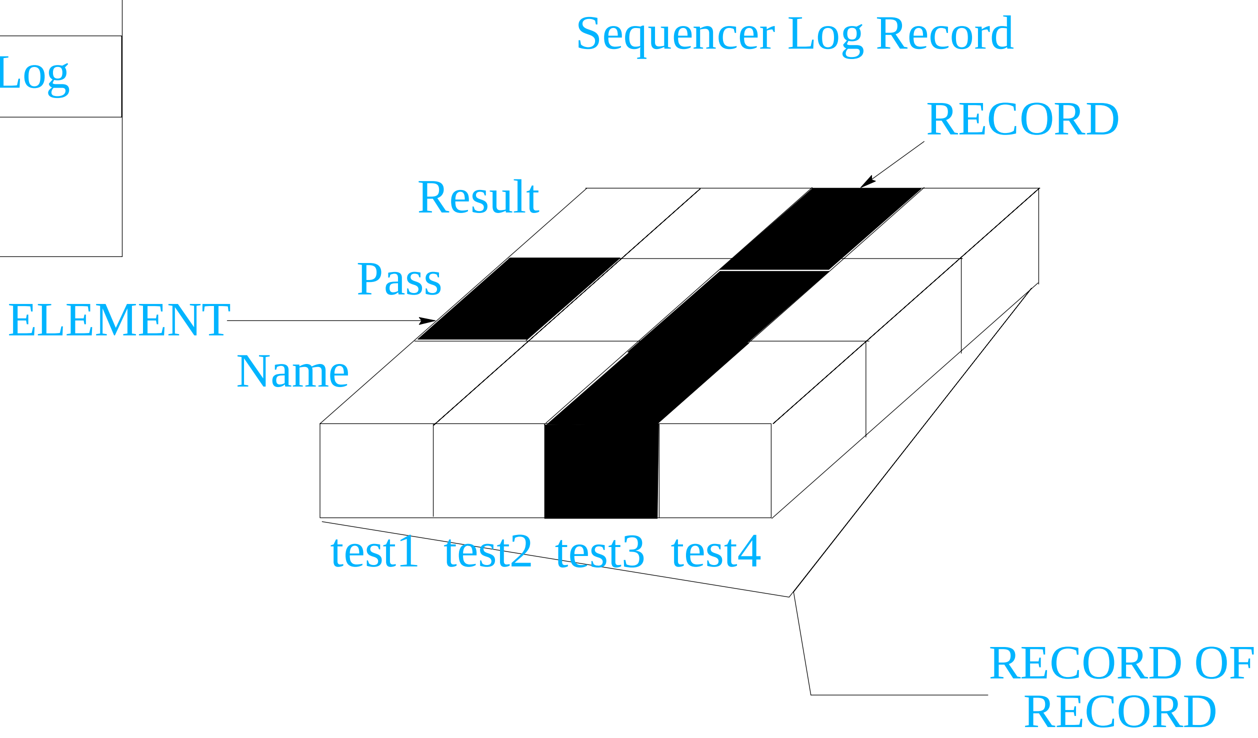
test2

test3

test4

Return

Log



Sequencer Log Record

RECORD

Result

Pass

ELEMENT

Name

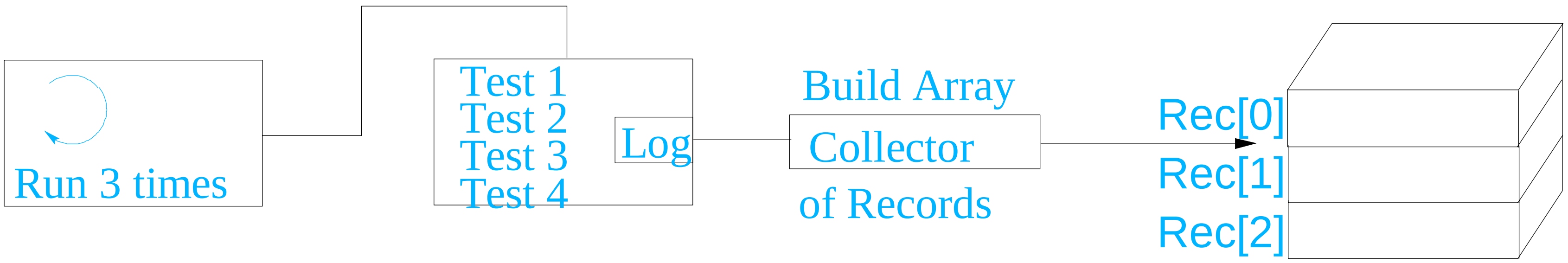
test1 test2 test3 test4

RECORD OF RECORD

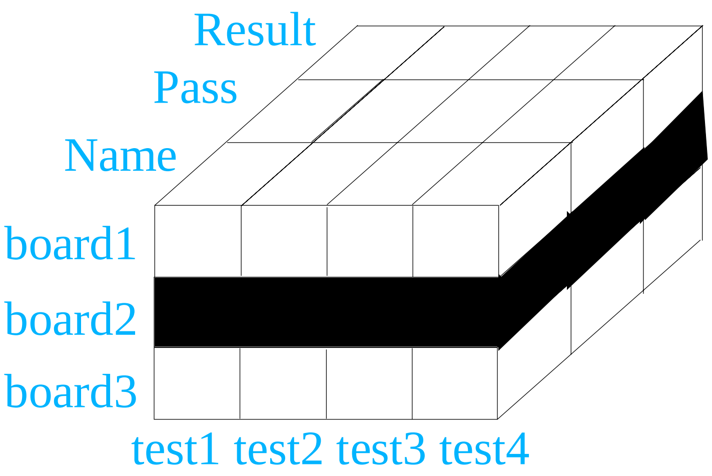
Rec.test1.pass = 1
Rec.test3 = Record
    name = "test3"
    pass = 0
    result = 3.142

HEWLETT PACKARD

Sequencer Logging (continued)

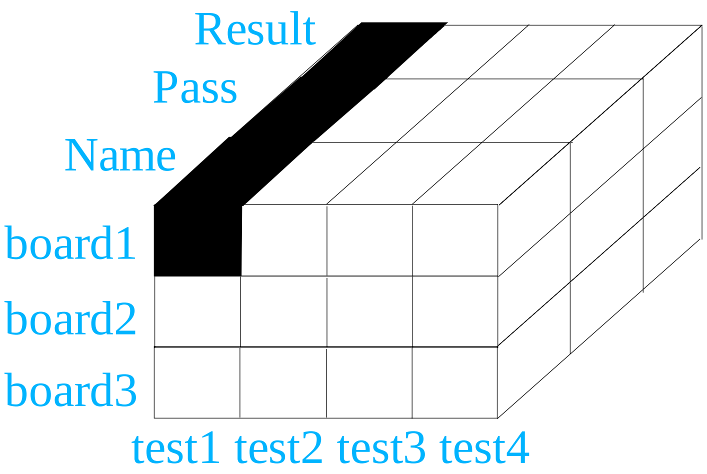


Sequencer Data Set



Rec[1]

Sequencer Data Set



Rec[0].test1

SCPI*

- HP-IB and RS-232: "HOW" to talk to devices

- VXIbus: "HOW" to talk to VXI devices
  - addressing devices (M/B or R/B)
    - Message-Based (Word Serial Protocol)
    - Register-Based with I-SCPI only

- Interface standards do not define "WHAT" to say

- SCPI: Standardizes "WHAT" to say

* Standard Commands for Programmable Instruments

Benefits of SCPI

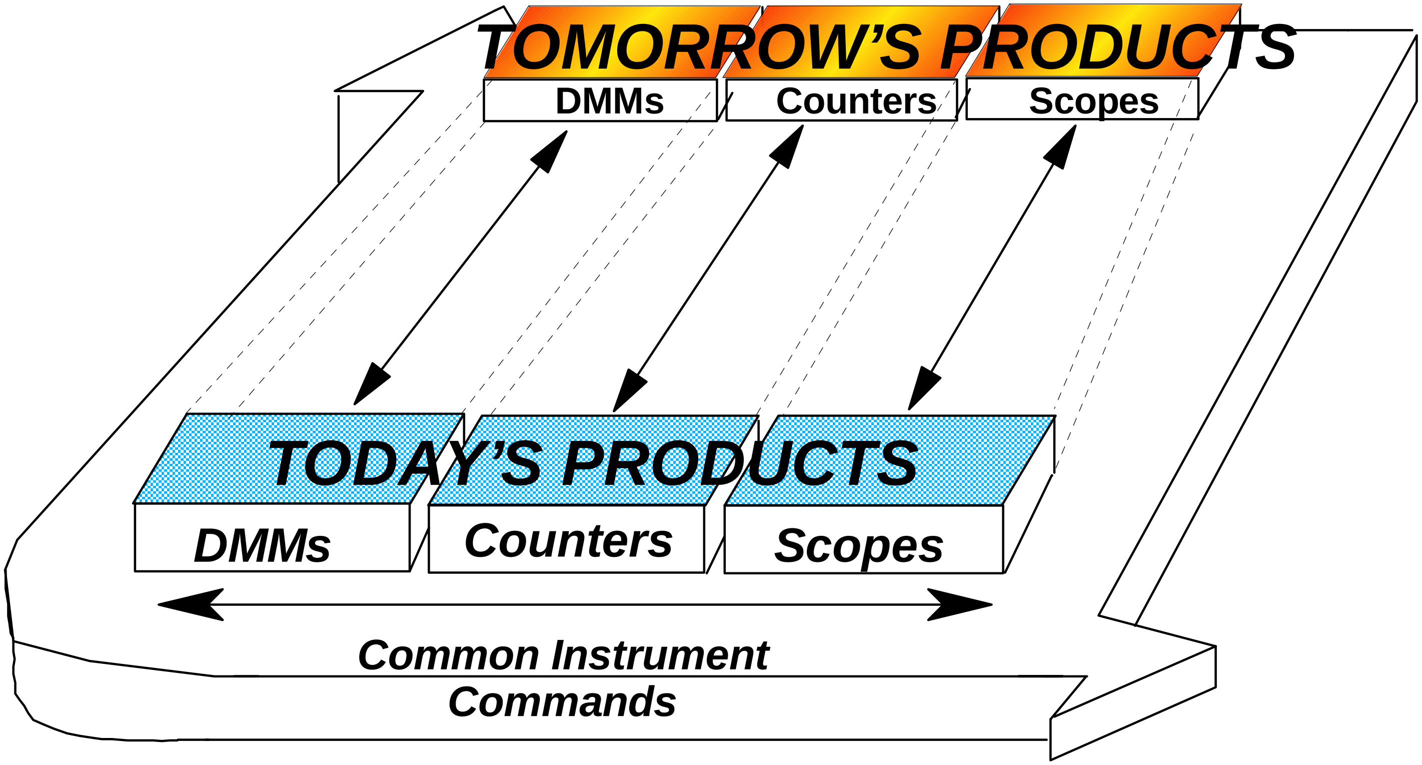
- Protects user written software

- Reduces learning/re-learning curve
  - Easier to learn (R&D, Mfg and Support)
  - Easier to bridge instrument knowledge

- Parsing can be faster
  - Efficient parsing algorithms

HP VEE

SCPI - Provides Compatibility Horizontal and Vertical Compatibility



HEWLETT PACKARD

Examples of Compatibility

Different types of instruments: same attributes

Different generations: Same instrument type

- Horizontal Compatibility:
  - OUTPUT@ Dmm;"TRIG:SOURce EXT"
  - OUTPUT@ Arb;"TRIG:SOURce EXT
  - OUTPUT@ FGen;"TRIG:SOURce EXT"

Programs the same function on all instruments

- Vertical Compatibility:
  - OUTPUT @34701A;"*RST"
  - OUTPUT @34703A;"*RST"
  - OUTPUT @34705A;"*RST"

Same IEEE 488.2 command resets all instruments