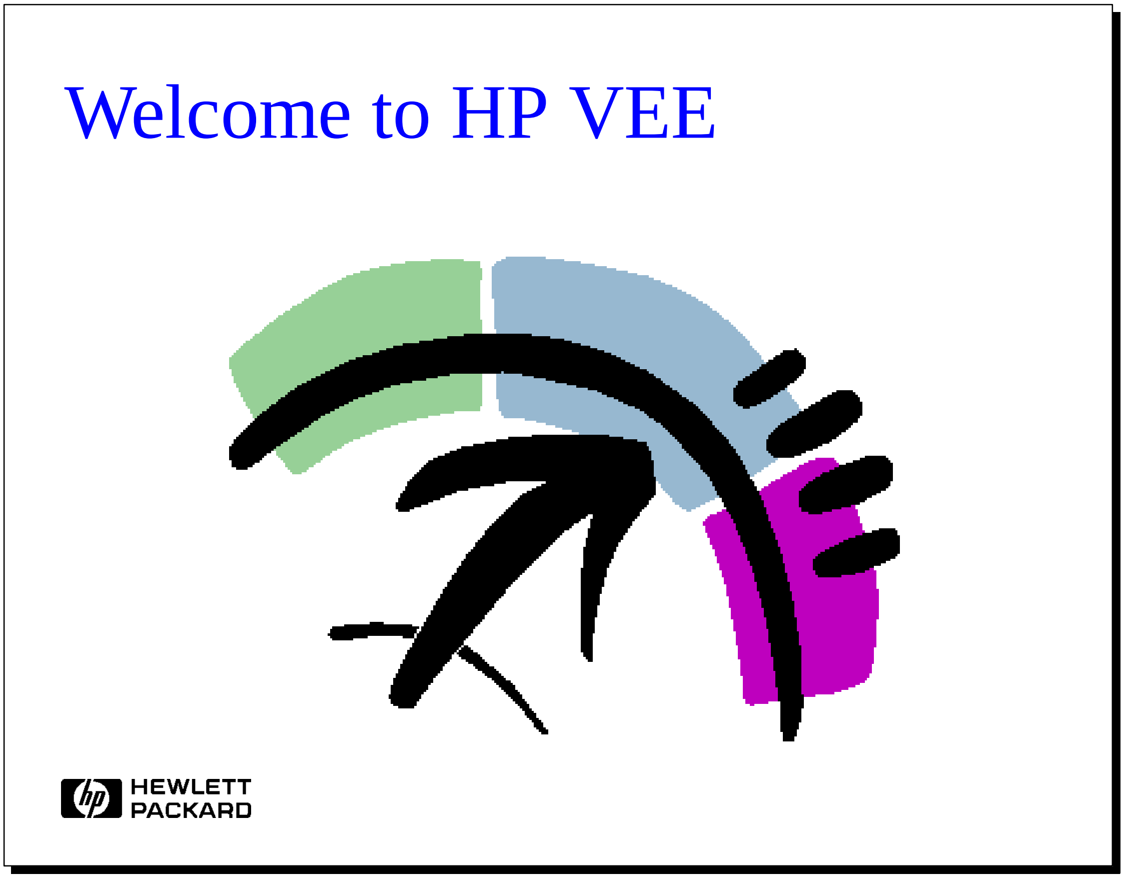
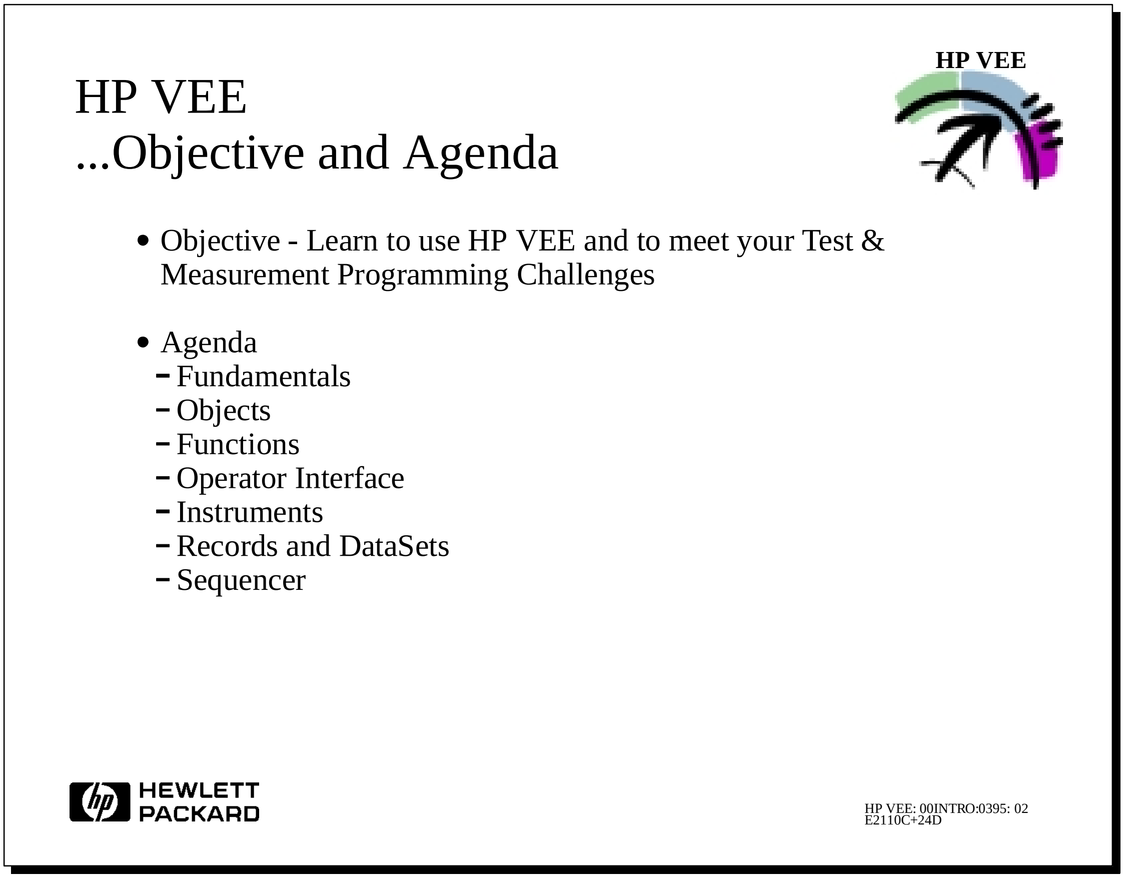
Introduction to HP VEE

Introduction to HP VEE
V3.1 ©1996

Introduction



Welcome to HP VEE



HP VEE

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

HEWLETT PACKARD

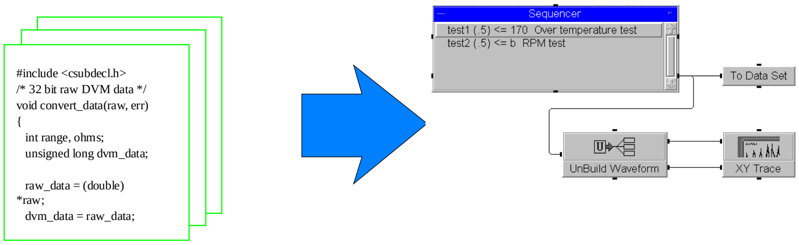
HP VEE: 00INTRO:0395: 02
E2110C+24D

You’re going to experience the power and functionality HP VEE has to help you meet your test and measurement programming challenges. You’ll learn about the only graphical programming language designed specifically for test. To help you understand how HP VEE works, the lab exercises are designed to give you hands-on experience in areas that are typically found in Test & Measurement applications.

HP VEE

What is HP VEE?

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



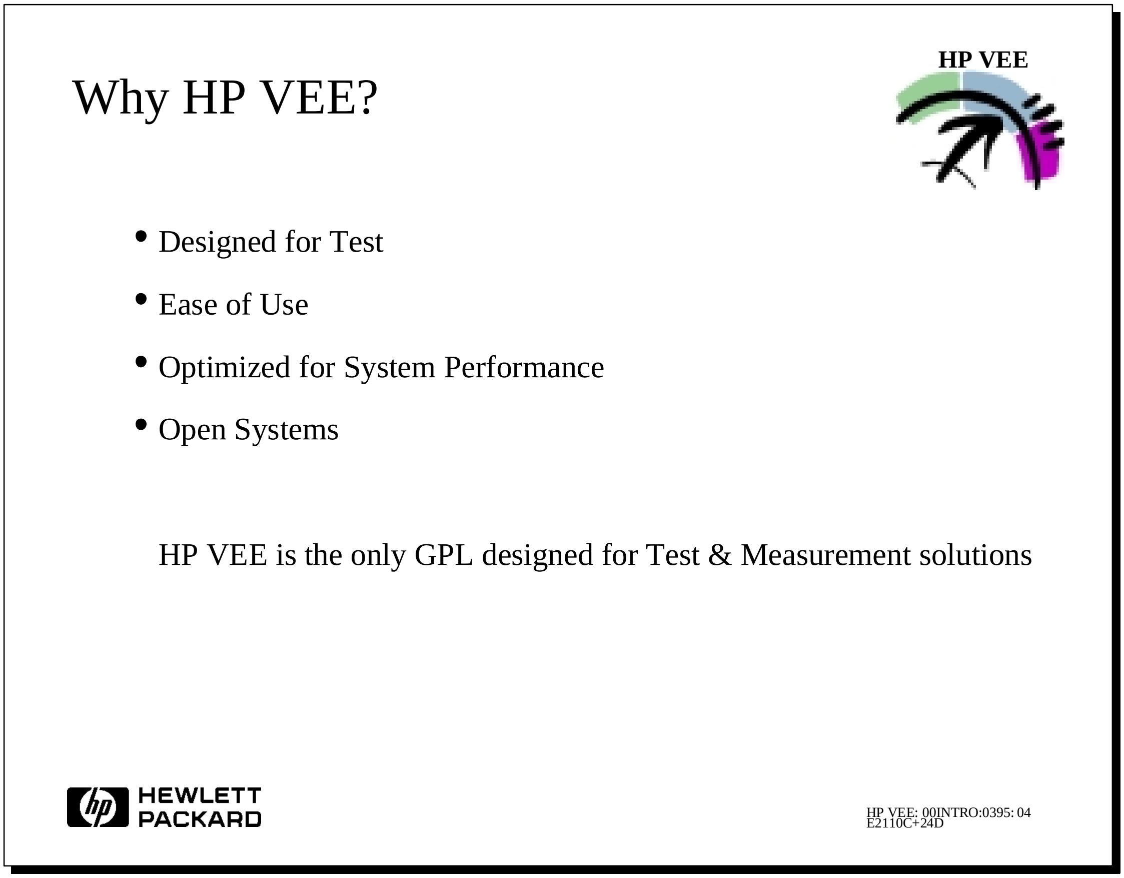
```
#include <csubdecl.h>
/* 32 bit raw DVM data */
void convert_data(raw, err)
{
   int range, ohms;
   unsigned long dvm_data;

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

Sequencer
test1 (.5) <= 170 Over temperature test
test2 (.5) <= b RPM test
To Data Set
UnBuild Waveform
XY Trace

HEWLETT PACKARD

HP VEE: 00INTRO:0395: 03
E2110C+24D

When first seeing HP VEE, you may feel that the graphical interface "hides" the functionality, however, you'll soon realize that **the graphical interface is the functionality**. HP VEE doesn't *generate* code **-** the objects in HP VEE *are* the code. Initially, it may not be easy to make the connection between a familiar textual language and a graphical language, but both perform similar functions.



HP VEE

Why HP VEE?

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

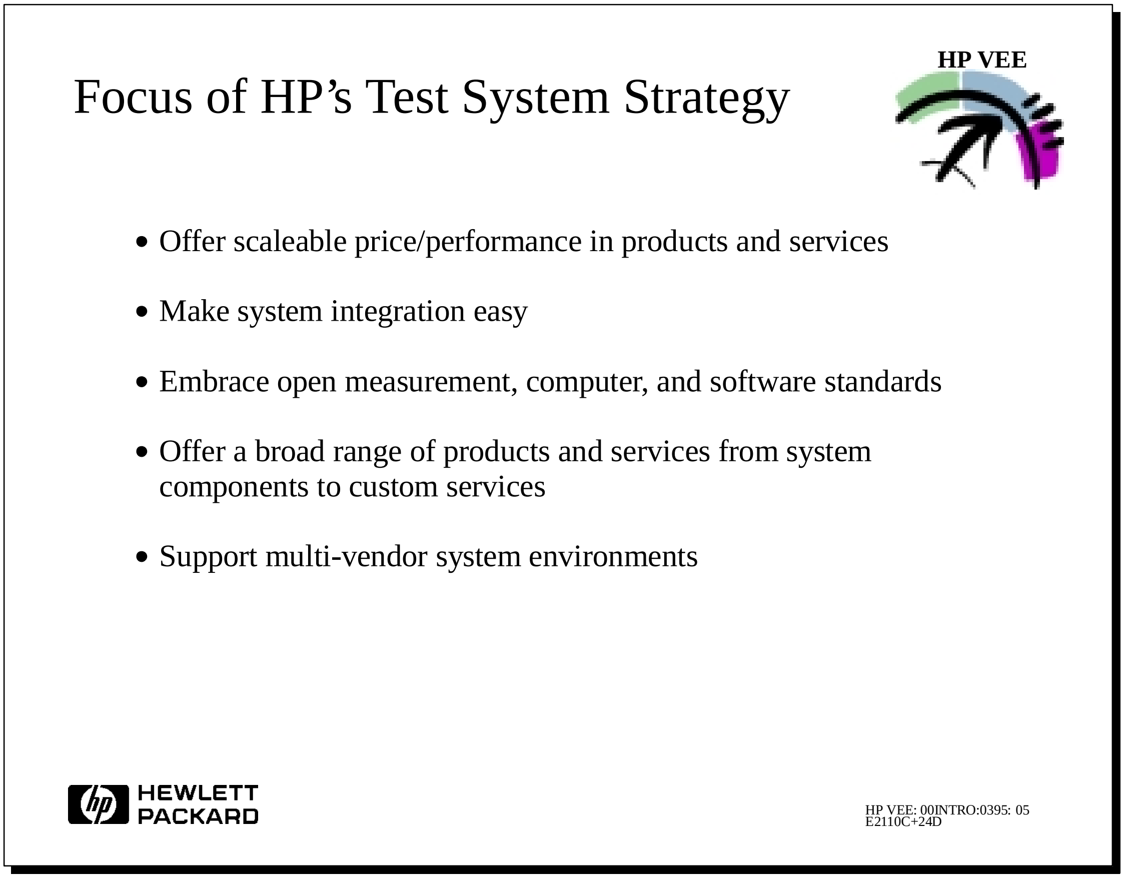
HP VEE is the only GPL designed for Test & Measurement solutions

HEWLETT PACKARD

HP VEE: 00INTRO:0395: 04
E2110C+24D

HP VEE is both a complete GPL and an interface with the measurement world. As an interface, HP VEE makes you more productive when you need to develop a complete test and measurement solution. HP VEE is your answer whether the solution includes graphical user interface (GUIs), instrument interfaces or seamless integration of your own text-based language.

HP VEE is part of Hewlett-Packard's overall test system strategy. This strategy consists of a broad line of hardware, computers, software, and services.



HP VEE

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

HEWLETT PACKARD

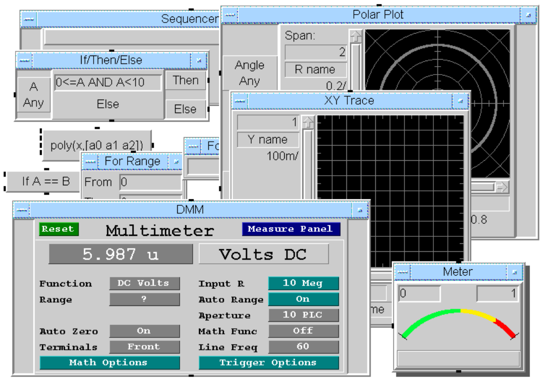
HP VEE: 00INTRO:0395: 05
E2110C+24D

Hewlett-Packard's test system strategy focuses on such key areas as performance, ease-of-use and standards. HP offers customers a wide range of products and services including system components, custom services and support of multi-vendor system environments.

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



DMM
Reset
Multimeter
Measure Panel
5.987 u
Volts DC
Function DC Volts
Range ?
Auto Zero On
Terminals Front
Math Options
Input R 10 Meg
Auto Range On
Aperture 10 PLC
Math Func Off
Line Freq 60
Trigger Options

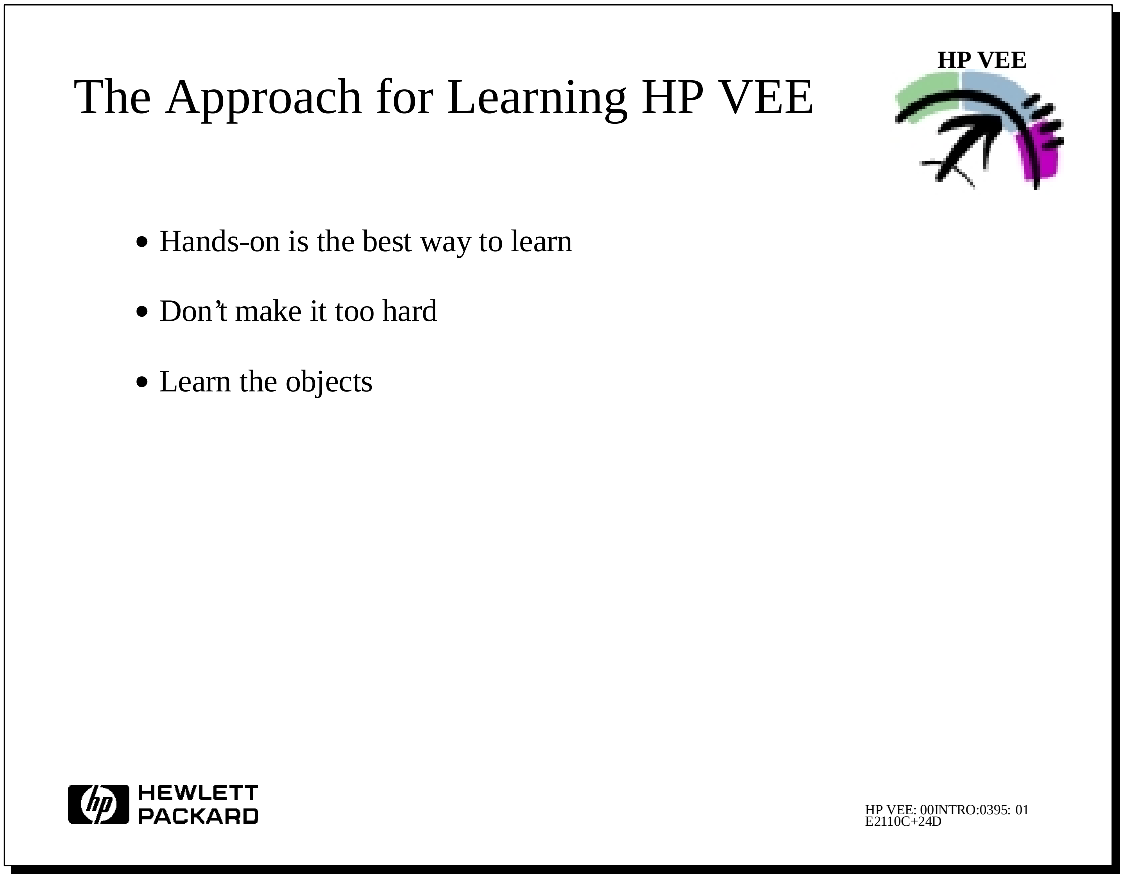
HP VEE: 00INTRO:0395:
E2110C+24D

When developing computer-aided test, three main areas need to be coded. To make this task easier, HP VEE provides three levels of instrument control: instrument panels, component drivers, and direct I/O. Besides the HP-IB I/O objects to send bus commands and retrieve data, HP VEE provides other data acquisition objects such as:

Read/Write Files
Read/Write Strings
Objects for user input
Objects to mathematically generate data

HP VEE provides math and analysis functions that range from elementary math to real/complex math and calculus, data handling and conversion, regression and digital signal processing, and more. When a solution has to be constructed from a long mathematical equation, a formula object is provided for you to key in the formula rather than building it from individual math objects. HP VEE also contains a powerful sequencer object that allows decisions to be made from the acquired data.

Because engineering and scientific data are better understood graphically, HP VEE provides a large number of graphical display objects. Each of these objects has dozens of modifiable options including multiple inputs, markers, auto scaling, etc. You can also display data in a text format with alphanumeric and scroll data objects, or with meter-type objects.



HP VEE

The Approach for Learning HP VEE

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

HEWLETT PACKARD

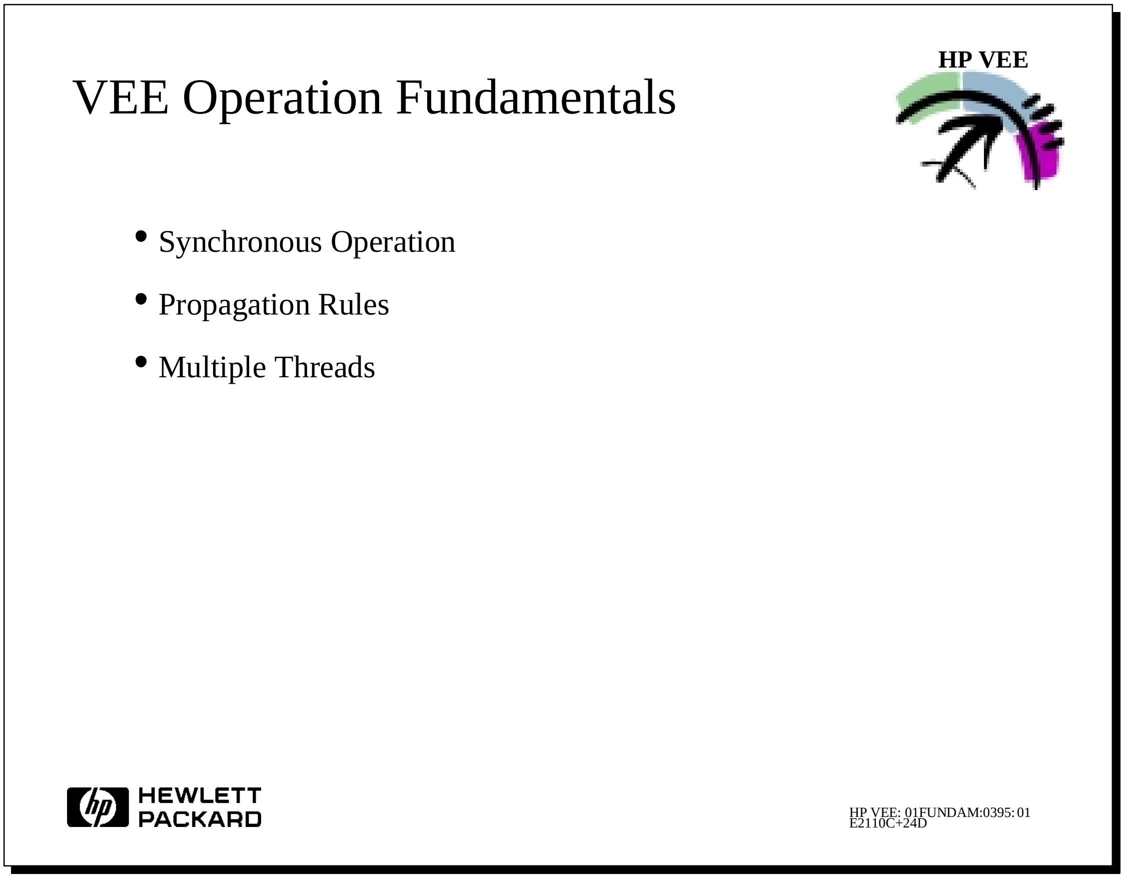
HP VEE: 00INTRO:0395: 01
E2110C+24D

At first; it's difficult to find correct objects to use; learn the objects.

As you proceed, the number of objects necessary becomes less to achieve a solution.

Hidden semantics of objects: finding what they can do.

Fundamentals



VEE Operation Fundamentals

HP VEE

- Synchronous Operation
- Propagation Rules
- Multiple Threads

HEWLETT PACKARD

HP VEE: 01FUNDAM:0395: 01
E2110C+24D

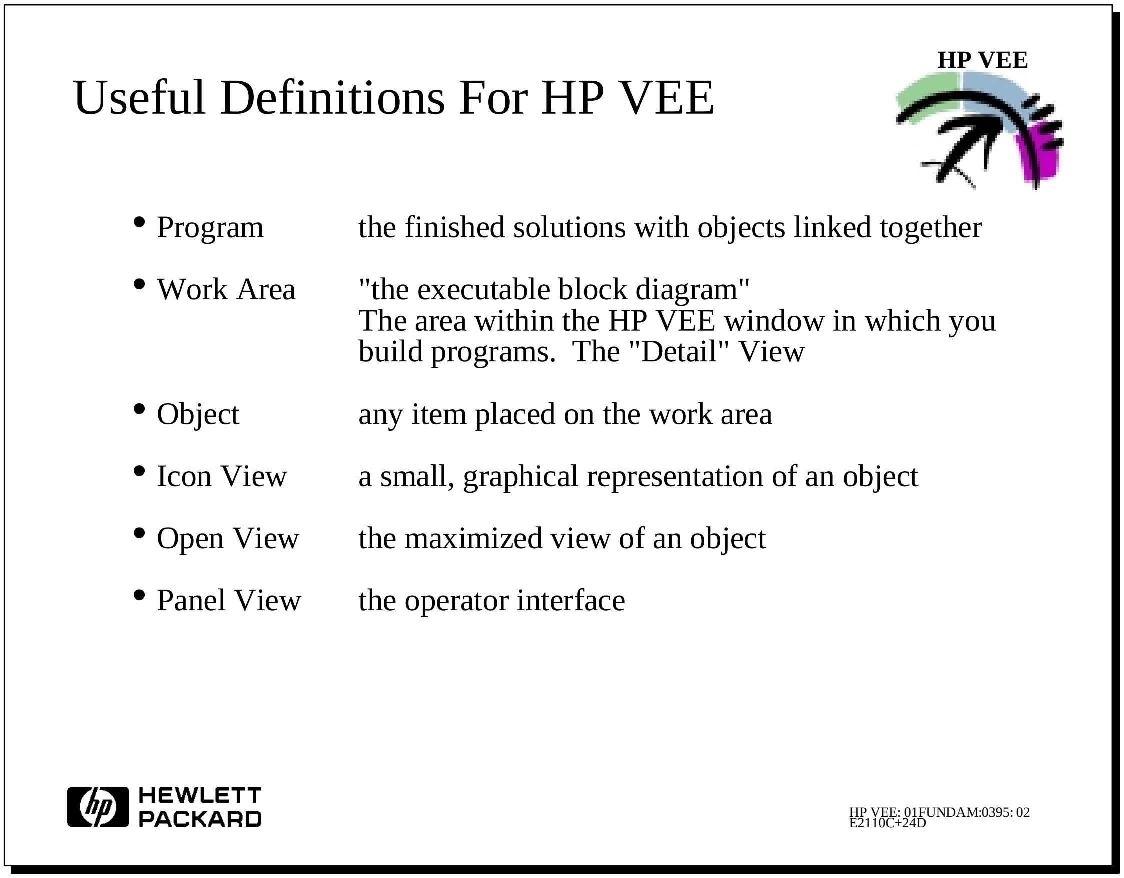
In this module we’ll discuss the fundamental principles of operation used by HP VEE.

We’ll cover:

Synchronous Operation, how the HP VEE objects function

Propagation Rules, how HP VEE decides which object operates next

Multiple Threads, how multiple sets of objects relate



HP VEE

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

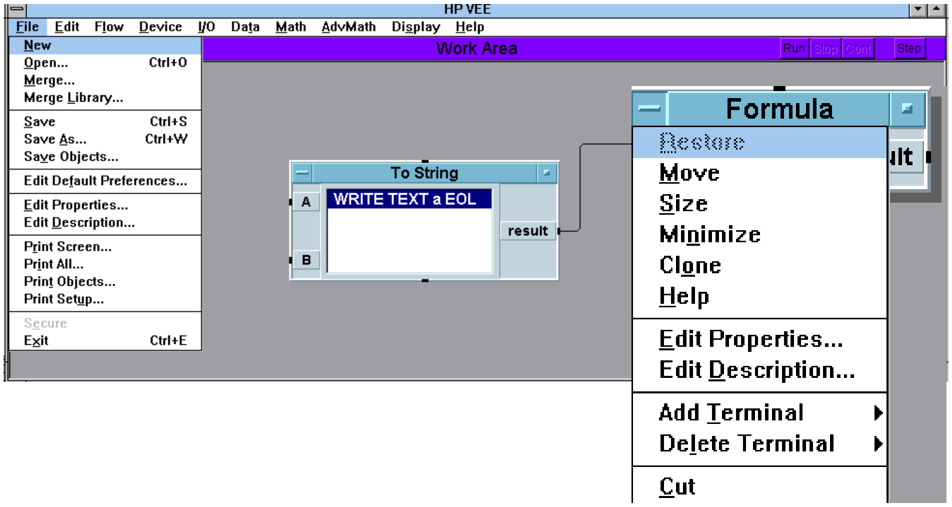
HEWLETT PACKARD

HP VEE: 01FUNDAM:0395: 02
E2110C+24D

First, let’s define some words commonly used with HPVEE.

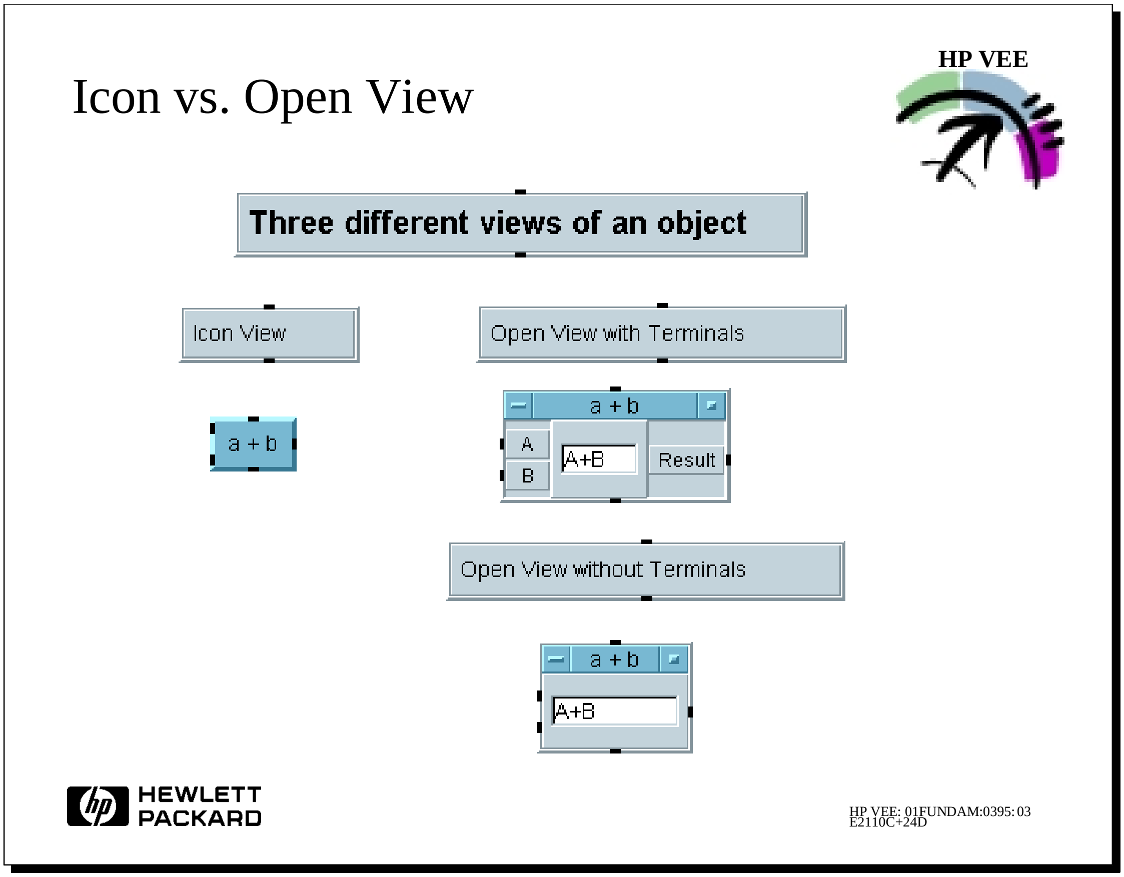
HP VEE Work Area

HP VEE



HEWLETT PACKARD

HP VEE: 01FUNDAM:0395: 2a
E2110C+24D



Icon vs. Open View

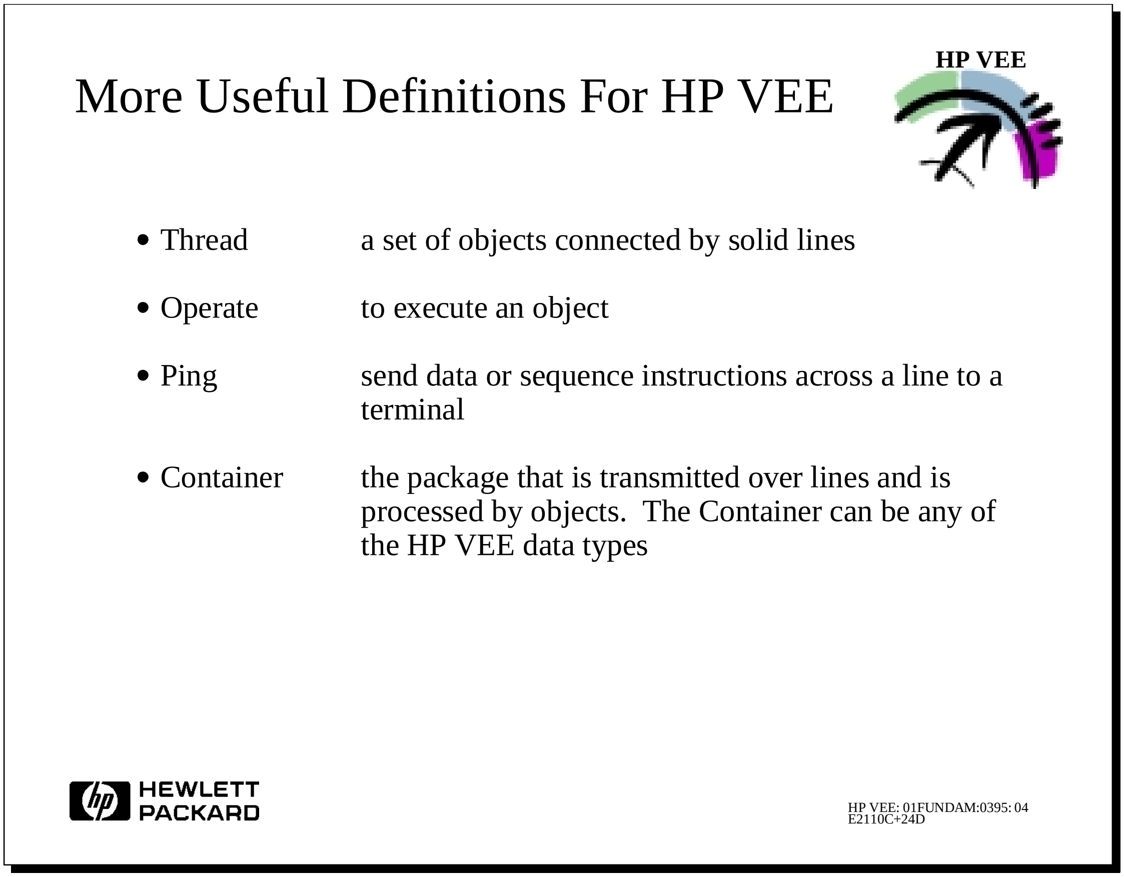
HP VEE

HEWLETT PACKARD

HP VEE: 01FUNDAM:0395: 03
E2110C+24D

Icon View This view has the object title, bitmap display (optional) and the object pins.

Open View This view has the Functions of the object. The title is optional, along with the object menu and minimize button. The object pins are visible, but the terminals are optional.



More Useful Definitions For HP VEE

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

HEWLETT PACKARD

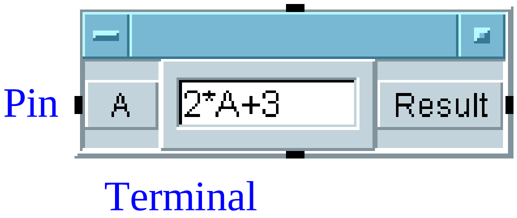
HP VEE: 01FUNDAM:0395: 04
E2110C+24D

Let’s define some more words commonly used with HPVEE.

Pins and Terminals

HP VEE

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



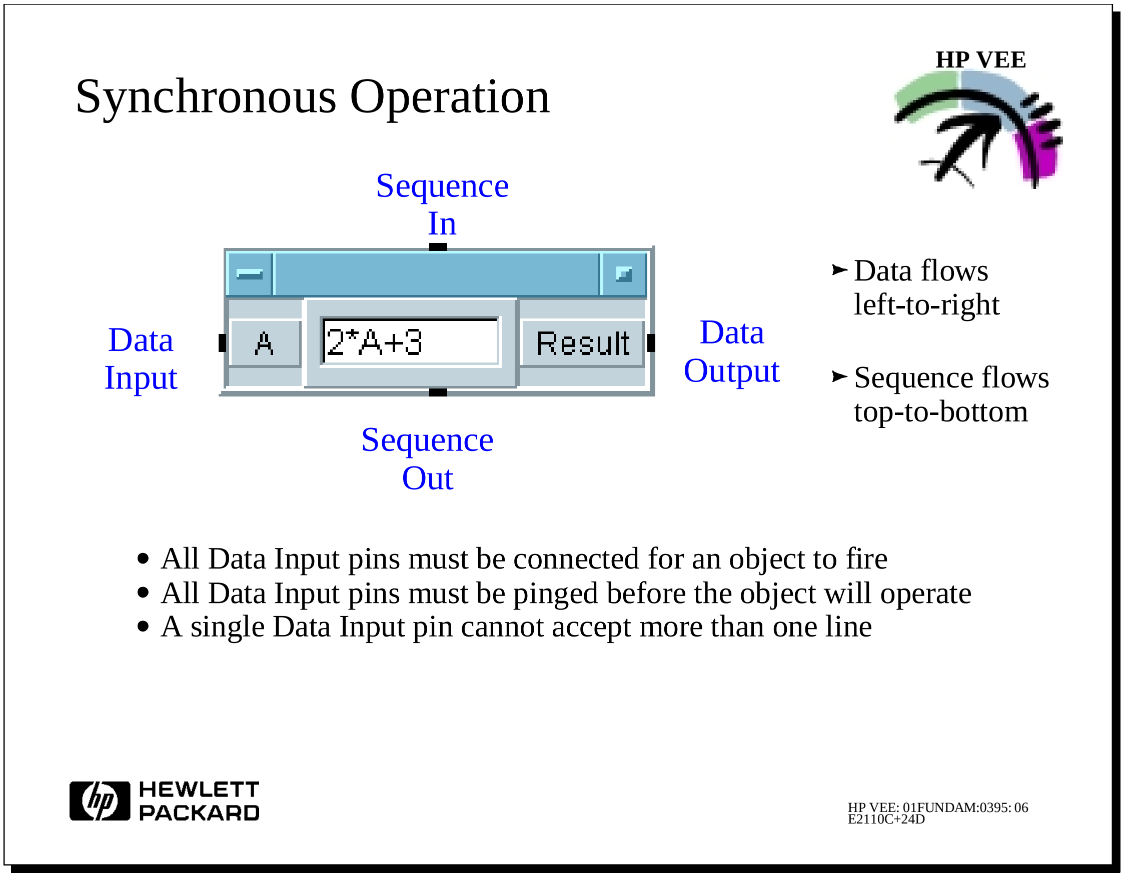
HEWLETT PACKARD

HP VEE: 01FUNDAM:0395:05
E2110C+24D

Pins and terminals provide input and output connections for data and control on HP VEE objects.

Pins provide connecting points between objects. Data pins receive and send data. Sequence pins control the execution flow of objects. You do not have to connect sequence pins for the object to operate. Asynchronous pins control the operation of the object. The action they control activates immediately. It doesn’t wait for the object to complete its operation. The Error Output pin sends error information and allows you to trap errors.

Terminals display data information for each pin. Note that sequence pins do not have a corresponding terminal.



Synchronous Operation

HP VEE

- ➤ 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

HEWLETT PACKARD

HP VEE: 01FUNDAM:0395: 06
E2110C+24D

All HP VEE objects operate in the same manner. Data flows from left to right through the object. Sequence flows from top to bottom.

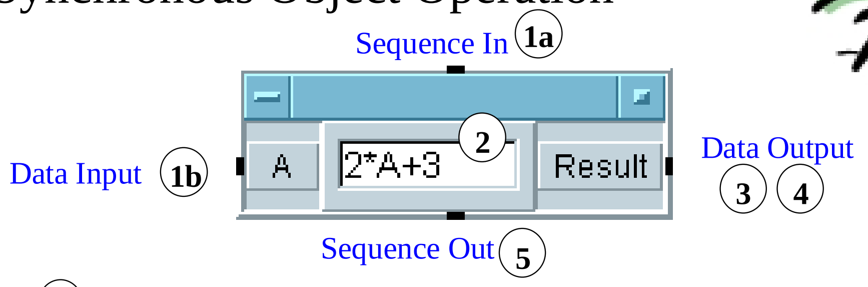
All Data Input pins MUST be connected. A Data Input pin cannot accept more than one input connection. Data output pins may connect to one or more objects.

Data Output and Sequence In or Out pins DO NOT need to be connected.

The object will not operate until all of its Data Input pins have been pinged.

The Sequence In is a hold off pin. If it is connected, the object cannot operate until the Sequence In is pinged.

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

HEWLETT PACKARD

HP VEE: 01FUNDAM:0395: 09
E2110C+24D

HP VEE objects operate in the following manner:

1. The object receives data input. Sequence Input is received (if connected).

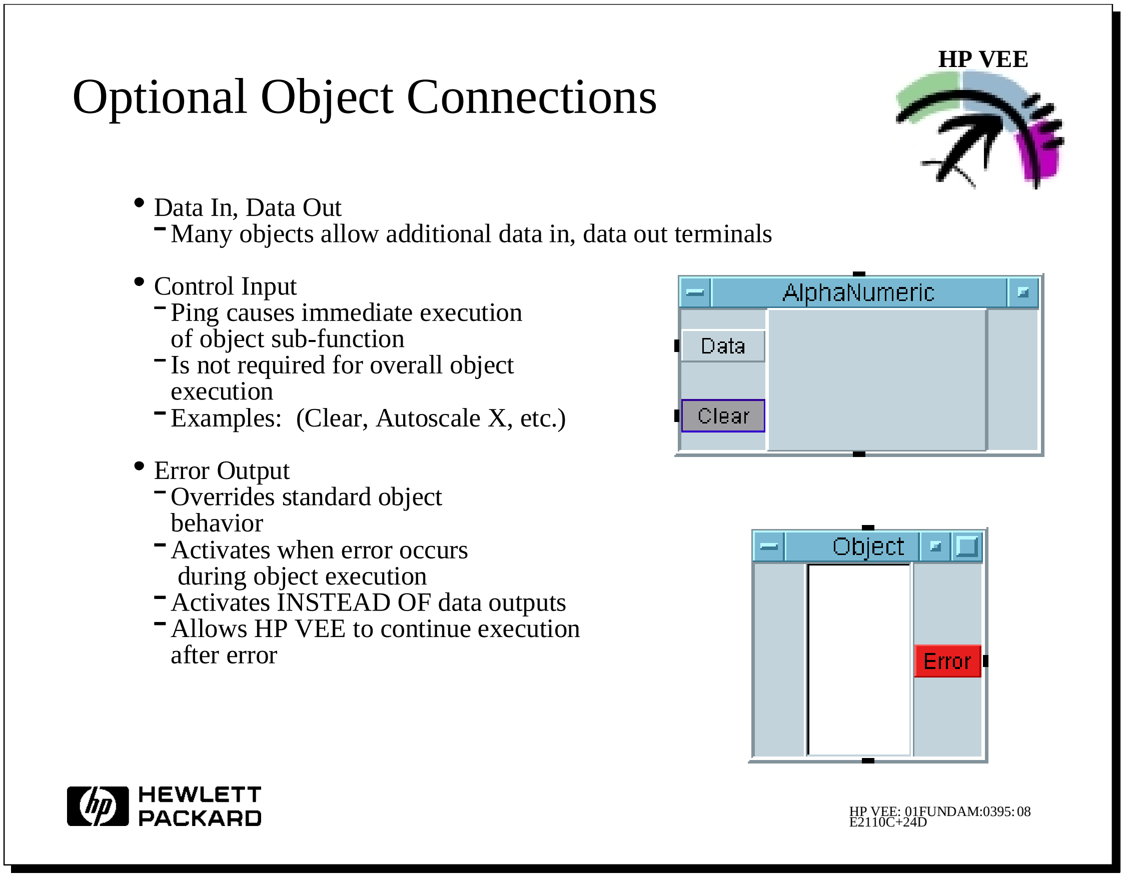
2. The object operates, performing its designated function.

3. The object sends data out to the next object.

4. The object waits for the next object in the thread to complete operation and send a "receipt acknowledged" back to it. This insures that all objects in a thread operate before HP VEE executes the next thread or subthread.

5. The object activates its sequence out pin.

6. The object deactivates.



Optional Object Connections

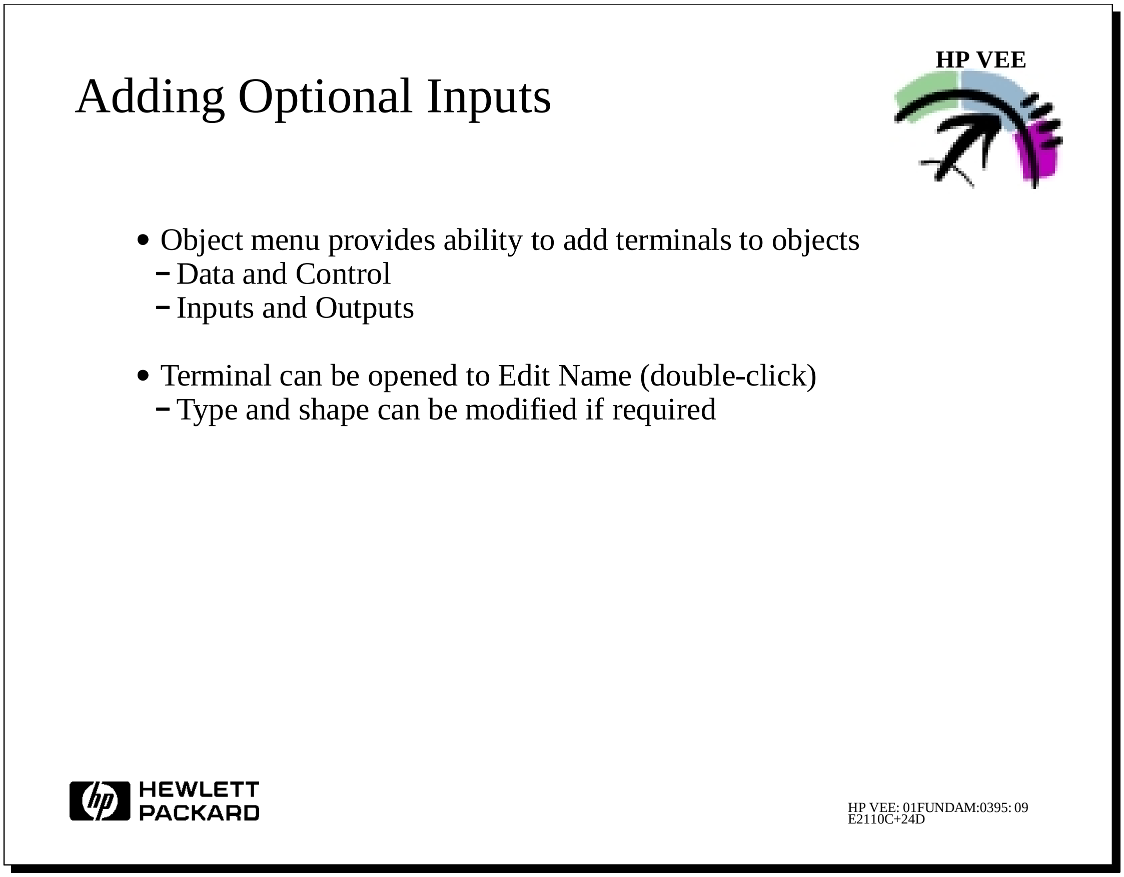
- Data In, Data Out
  - Many objects allow additional data in, data out terminals
- Control Input
  - Ping causes immediate execution of object sub-function
  - Is not required for overall object execution
  - Examples: (Clear, Autoscale X, etc.)
- Error Output
  - Overrides standard object behavior
  - Activates when error occurs during object execution
  - Activates INSTEAD OF data outputs
  - Allows HP VEE to continue execution after error

HP VEE: 01FUNDAM:0395:08
E2110C+24D

Each object starts with a set of default input and output pins. Many devices allow additional input and output data pins as well as control and error pins. Some objects only allow specific data pins to be added.

The control pins cause immediate execution of an object sub-function, such as Clear, Auto-scale, etc. The object does not need to receive input on a control pin in order to execute. However, it must receive input on each data pin in order to execute.

The error pin overrides standard object behavior in the event of an error. It sends out an error message when an error occurs. The object does not send out data on its data pins when an error occurs.



HP VEE

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

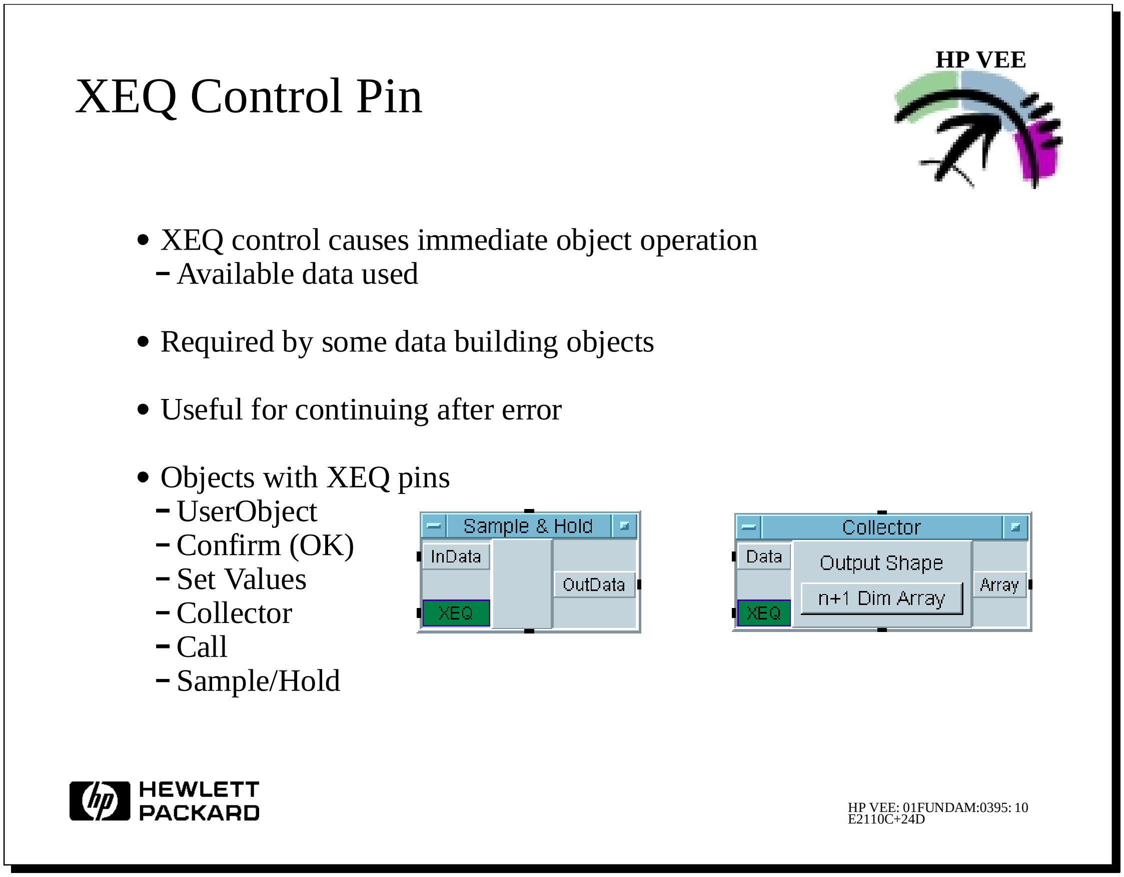
HEWLETT PACKARD

HP VEE: 01FUNDAM:0395: 09
E2110C+24D

The Terminals selection on the object menu allows you to add terminals to an object. Selecting Edit Properties... Show Terminals displays the terminal names on the object.

You can also open and edit terminal characteristics such as name, type and shape. Note that some characteristics are unalterable.

Depending on the type of object, Adding Inputs might be additional Data Inputs or object specific inputs, such as Frequency or Amplitude for the Device ==> Virtual Source ==> Function Generator. This will be the same for Data Outputs.



HP VEE

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

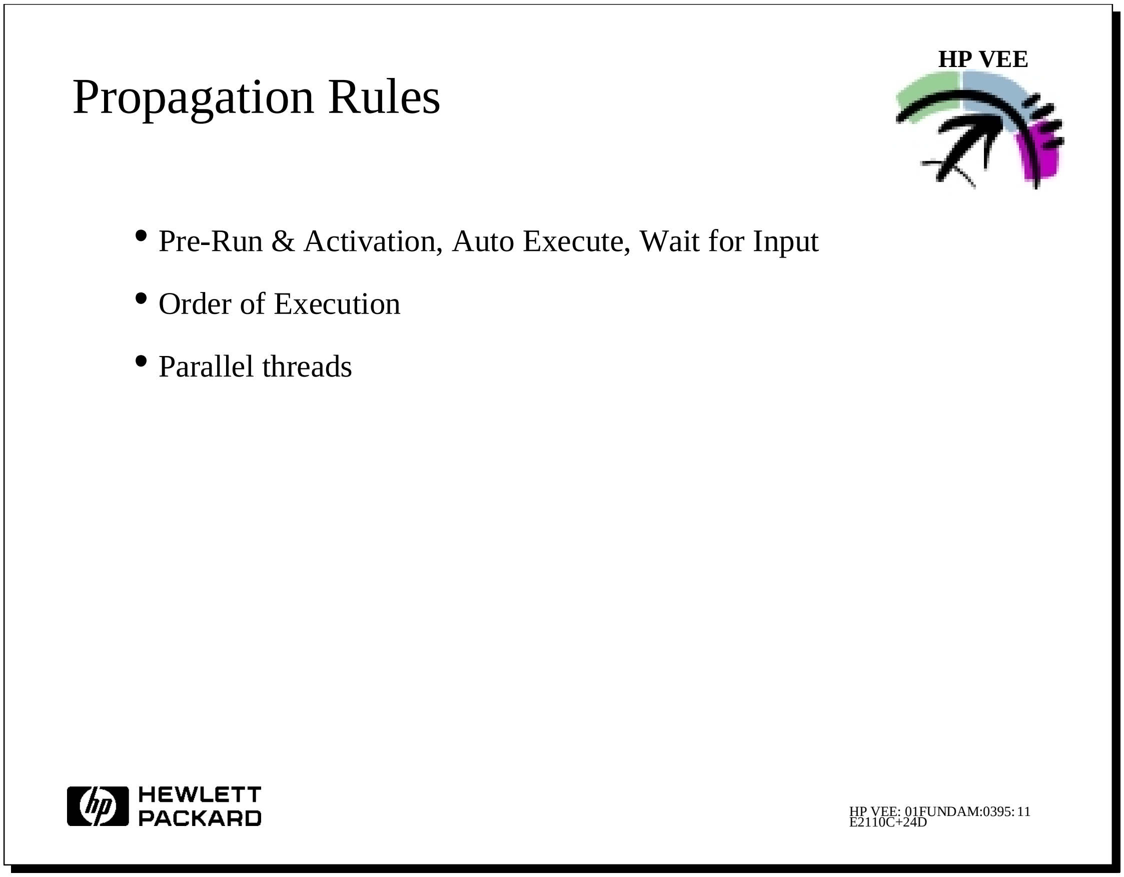
Sample & Hold
InData
XEQ
OutData
Collector
Data
XEQ
Output Shape
n+1 Dim Array
Array

HEWLETT PACKARD

HP VEE: 01FUNDAM:0395: 10
E2110C+24D

The XEQ pin causes the object to operate immediately. The object uses whatever data is available on the input pins. This is useful when continuing after an error. Some data building objects require the use of this pin to "finish" their operation. Note that XEQ pins are only available on the following objects:

UserObject
Confirm (OK)
Set Values
Collector
Call
Sample/Hold



HP VEE

Propagation Rules

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

HEWLETT PACKARD

HP VEE: 01FUNDAM:0395: 11
E2110C+24D

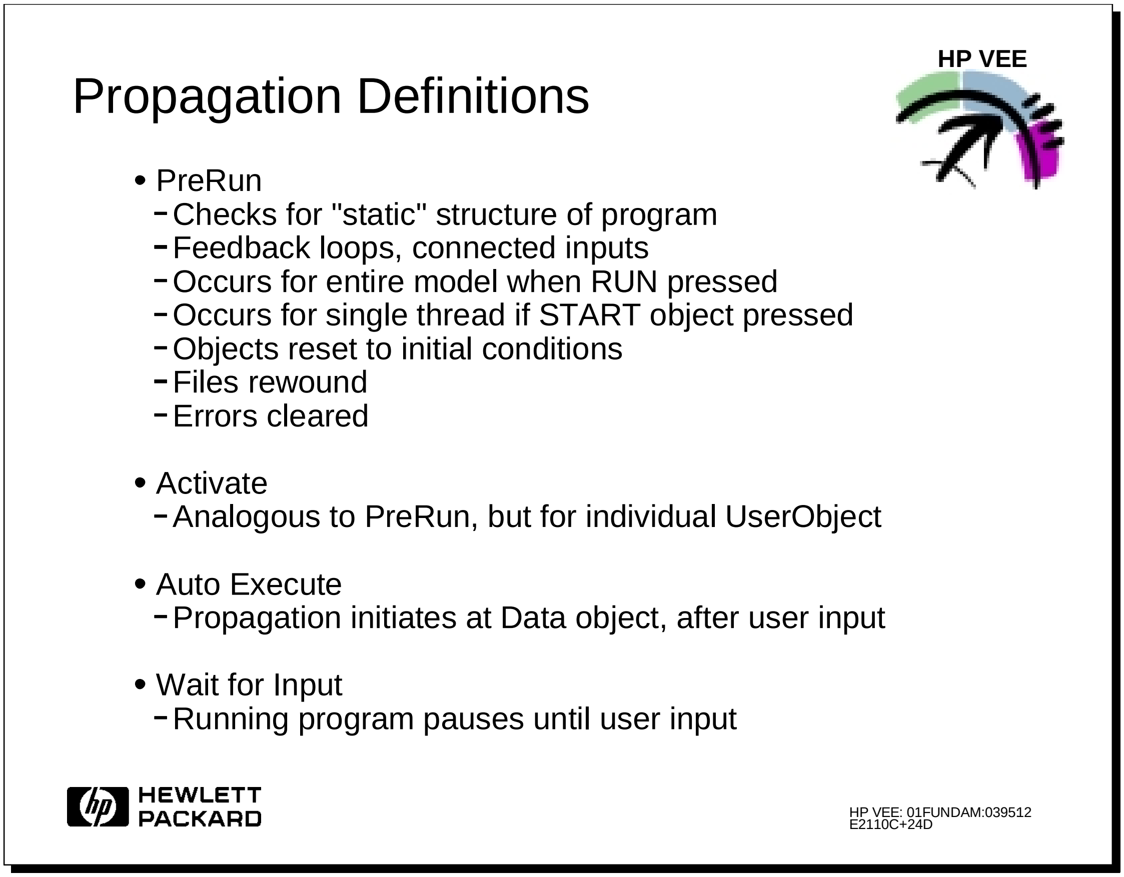
Now let’s take a look at the Rules of Propagation, or how HPVEE decides which object operates next. We’ll also examine:

PreRun, how HP VEE initializes objects and work areas

Activation, what happens when a work area begins operation

Auto Execute, where some objects can operate without pressing Run

Wait for Input, pauses for user input



HP VEE

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

HEWLETT PACKARD

HP VEE: 01FUNDAM:039512
E2110C+24D

HP VEE performs a PreRun (the following operations) when you press a Start or Run button:
Checks the structure of your program for proper construction and connectivity
Checks for Feedback loops, setting data input terminals on feedback loops to nil
Determines if all data and XEQ inputs are connected
Resets objects to initial conditions
Rewinds data files to their beginning
Clears errors

Note that HP VEE PreRuns your entire program when you press Run, but only an individual thread when you press Start on that thread.

Activation is analogous to a procedure call in a textual programming language. It initializes the state (PreRuns) of an individual UserObject each time the UserObject operates.

Auto Execute causes an object to propagate data after a user input. It is not necessary to press Start or Run.

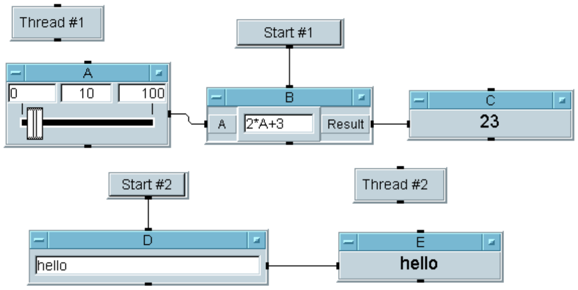
Wait for Input - pauses a program or thread that is running until the user has inputted data.

Auto Execute and Wait for Input are available on Sliders, Constants, and other data input objects.

HP VEE

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

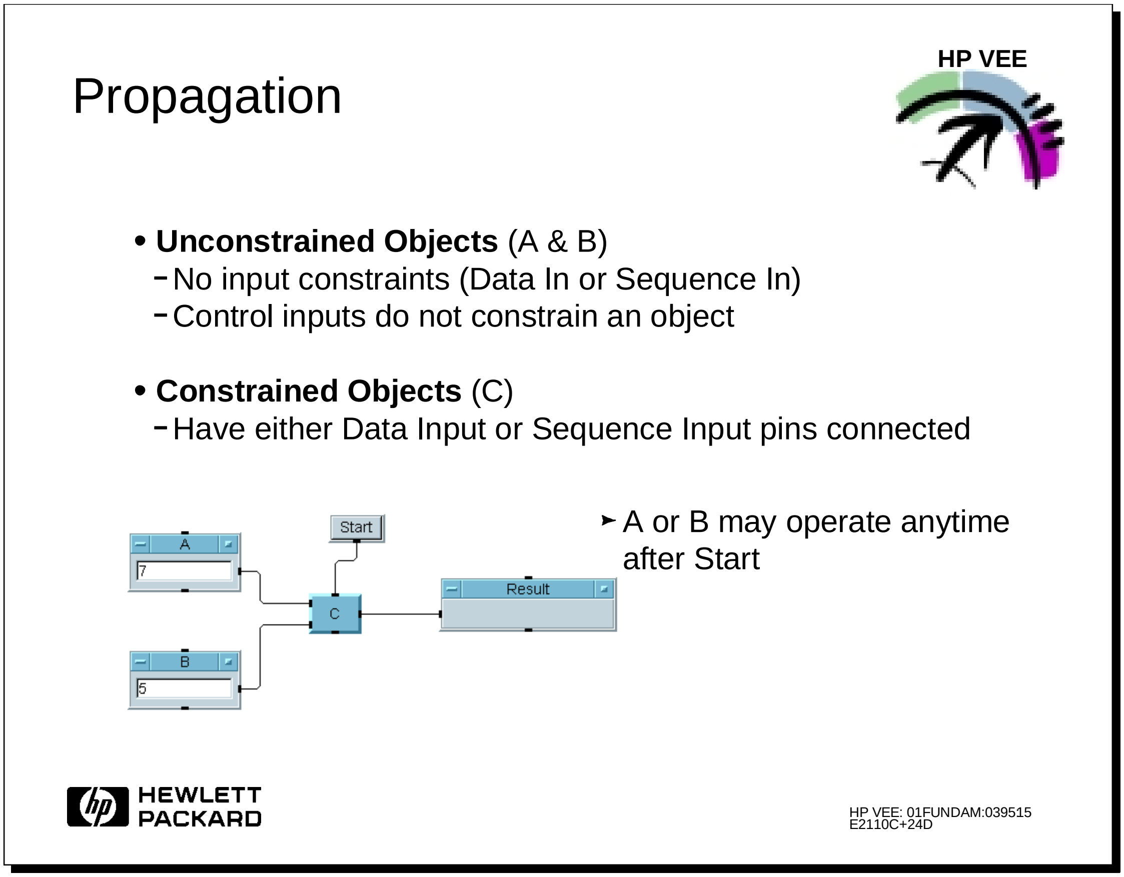


➤ Initiates thread propagation

➤ Pressing Start #1 does not affect Thread #2

HEWLETT PACKARD

HP VEE: 01FUNDAM:0395: 13
E2110C+24D



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: 01FUNDAM:039515
E2110C+24D

HP VEE models execute in the following order:

Start Objects operate first

Unconstrained Objects, objects with no data or sequence inputs, operate next [A & B]

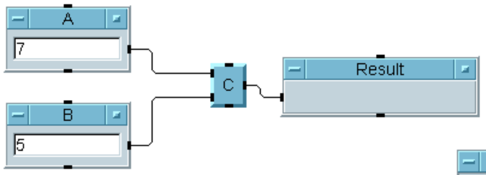
Constrained Objects then operate when their input constraints are satisfied [C]

In this example, A or B may operate any time after the Start Object operates. A and B will operate in an unspecified order relative to each other.

HP VEE

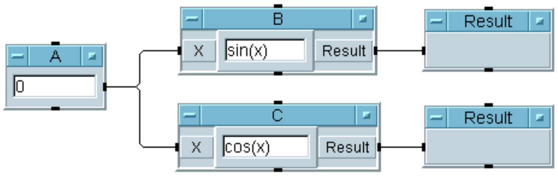
Propagation Rules

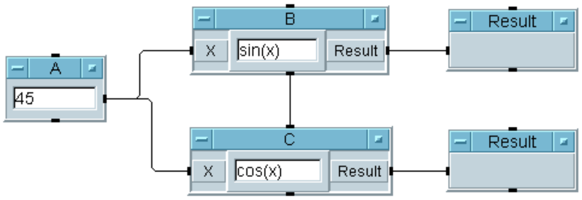
- Unconstrained objects may operate in any order



➤ 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

HEWLETT PACKARD

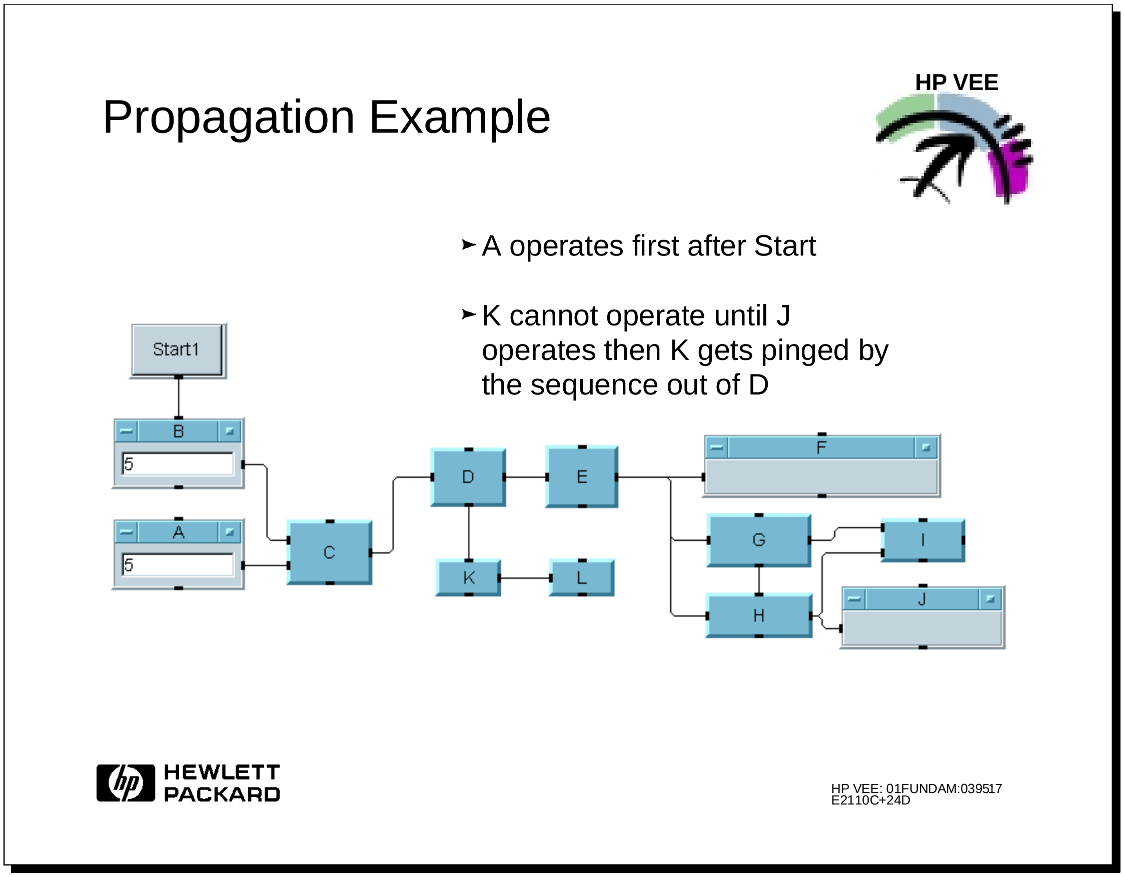
HP VEE: 01FUNDAM:0395: 16
E2110C+24D

Just to be sure you understand propagation in HP VEE, let’s look at some more examples.

In the first example, A or B may execute first, but C must wait until both A and B have completed their operation.

In the next example, A executes first. B and C must wait until A finishes. Then B and C execute. You cannot determine which one will go first.

If the order in which B and C operate is important, use a sequence connection as shown in the bottom example. In this case, C must wait until B finishes.



Propagation Example

HP VEE

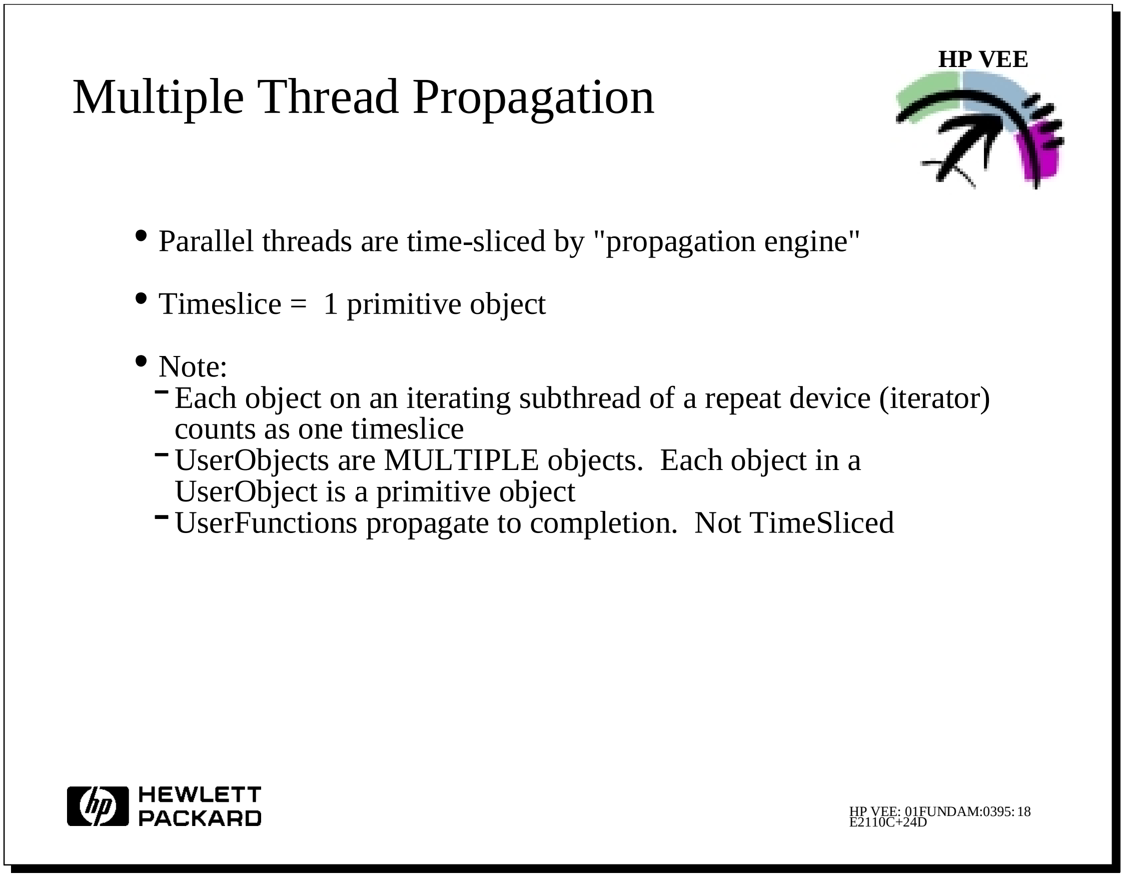
- A operates first after Start
- K cannot operate until J operates then K gets pinged by the sequence out of D

HP VEE: 01FUNDAM:039517
E2110C+24D

Here is a more complex example. Note that the objects operate in alphabetical order. Notice that D cannot activate its Sequence Out pin until as many objects as possible down- thread from it complete. A operates first because it is unconstrained.

In the case of G, it activates its Sequence Out pin before I completes because I cannot complete until H operates, providing data for I.

Note: F & G and I & J are indeterminate.



HP VEE

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

HEWLETT PACKARD

HP VEE: 01FUNDAM:0395: 18
E2110C+24D

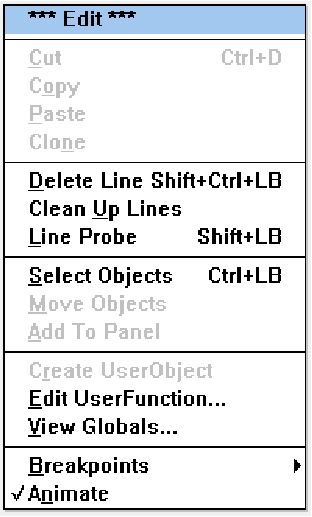
When multiple threads exist, HP VEE shares the processor among the various threads. The time slice is the time it takes to execute a single primitive object (i.e., Gate, Formula, Alpha ...).

Note that each object on an iterating subthread executes for one time slice. Also note that HP VEE treats UserObjects as multiple objects. Each object within the UserObject operates for one time slice.

Debugging

HP VEE

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



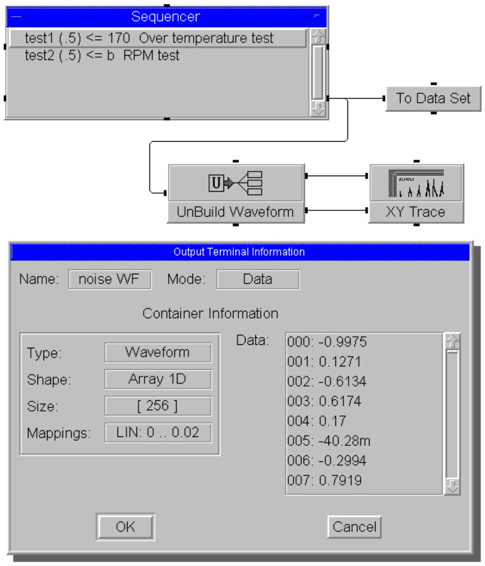
HEWLETT PACKARD

HP VEE: 01FUNDAM:0395: 19
E2110C+24D

HP VEE

Debugging

- 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

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
```

HEWLETT PACKARD

HP VEE: 01FUNDAM:0395:20
E2110C+24D

HP VEE provides several tools under the Edit menu to help you debug your models.

Edit ==> Animate

Show Exec Flow	- Highlights the object that is currently executing. A highlighted border appears around the executing object.
Show Data Flow	- Shows the route that data takes through the program. A small square marked moves along the lines connecting objects to show the movement of a data container.
Set Breakpoints	- Stops execution of the program before it executes this object. Each object has a settable breakpoint. An object with breakpoint set has a black border.
Edit ==> Line Probe	- Displays the container information (data) transmitted on a line between two objects. Text, Scalar, Array, Record, etc.
I/O ==> Bus I/O Monitor	- Displays the interface bus traffic into and out of HP VEE.

Lab 1

Menu Structure

Objective: Become familiar with HP VEE's menu structure.

Notice the short cuts listed on the menus. For example,
File =>Save is **Ctrl** + **s** which means you simultaneously press the **Control** key
and then the s.
Note that **Device=>Virtual Source ->** reveals a cascading menu
Note that **File Open** ... is a dialog box menu

Next, look at the HP VEE on-line help system. The help system will give you information about each object along with it's location. There are also a how to section and a glossary. Be sure to check out the section **Help=>Contents - Using Keyboard Shortcuts.**

With **Animate** ON, (**Edit=>Animate)**, open the listed programs to observe synchronous operation and the rules of propagation.

proprule.vee
propbig.vee
startobj.vee

HEWLETT PACKARD

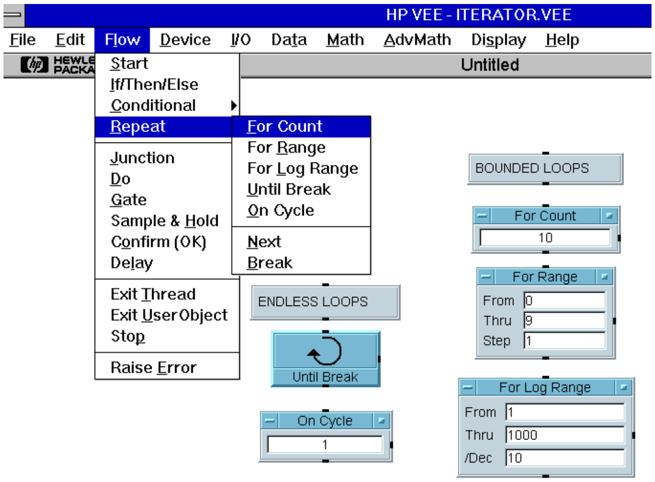
Objects and Data

Introduction to HP VEE
V3.1 ©1996

HP VEE

Repeat (Iterators)

- Repeatedly propagate data onto a subthread
- Bounded Loop
  - For Count
  - For Range
  - For Log Range
- Endless Loop
  - Until Break
  - On Cycle



HEWLETT PACKARD

HP VEE: 02DATAOB:0395:01
E2110C+24D

The Repeat objects let you repeatedly execute a subthread. The first set of objects repeat a bounded loop, executing a set number of times.

The For Count object executes this subthread the number of times defined by the count. Note that if you use the output of the Count object, it counts from zero 0 (not one).

The For Range object executes a subthread a number of times specified by a beginning value, an ending value and an increment.

The For Log Range object executes a subthread a number of times specified by a beginning value, an ending value and an increment. The output values are evenly distributed along the log 10 scale.

The second set of objects loop endlessly.

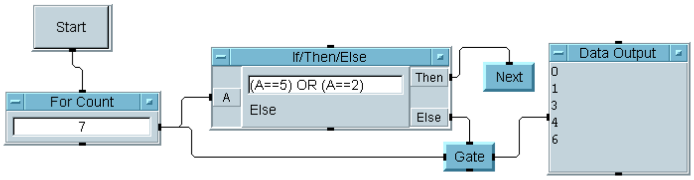
The Until Break object repeatedly executes a subthread until it encounters a Break object.

The On Cycle object repeatedly executes a subthread at a regularly timed interval.

HP VEE

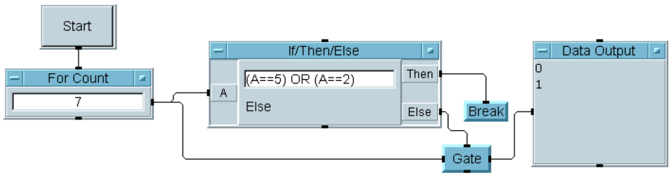
Early Loop Termination

- Next - terminates propagation of current iteration



‣ A executes repeatedly

- Break - terminates current and future iterations



‣ A executes repeatedly

HEWLETT PACKARD

HP VEE: 02DATAOB:0395:02
E2110C+24D

HP VEE provides two ways to terminate a loop early.

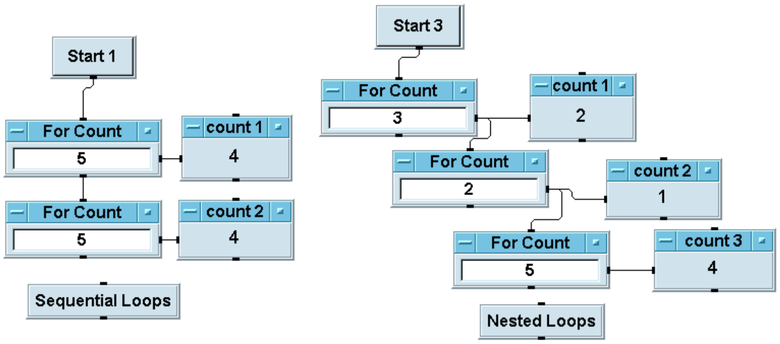
The Next object ends the current iteration and allows the iterator to go on to the next iteration.

The Break object ends the current and all future iterations. The iterator activates its Sequence Out pin and stops operating.

HP VEE

Thread Structures

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



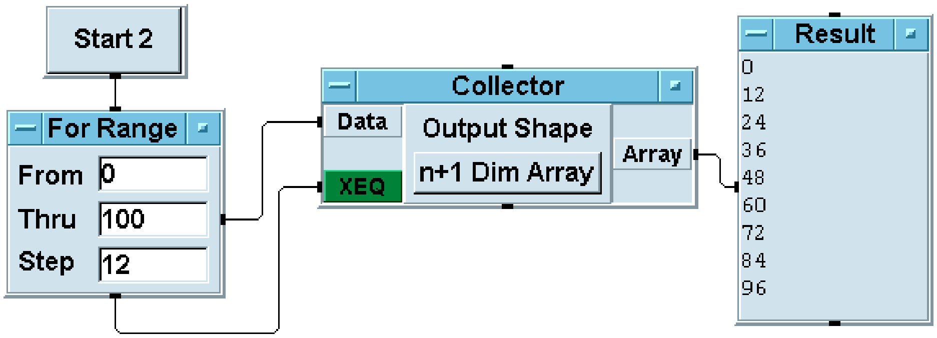
HEWLETT PACKARD

HP VEE: 02DATAOB:0395: 02a
E2110C+24D

An HP VEE subthread is the basic unit of your program. Each block in your block diagram becomes a UserObject in the HP VEE program. Here we illustrate how you structure subthreads for sequential and nested loops.

Collecting Interative Data

HP VEE



From 0
Thru 100
Step 12

Output Shape: n+1 Dim Array

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

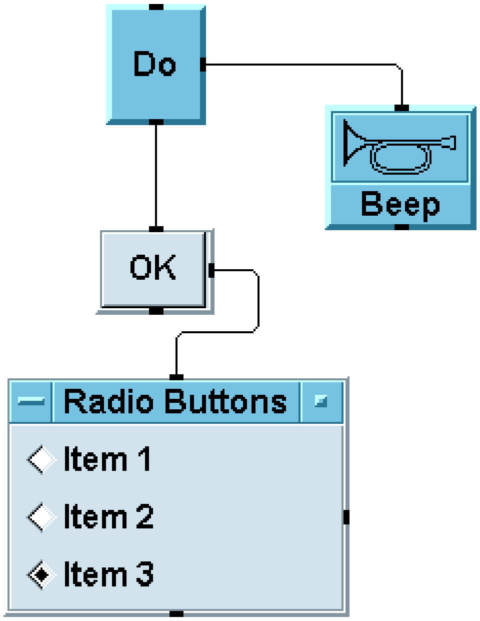
HEWLETT PACKARD

HP VEE: 02DATAOB:0395: 02b
E2110C+24D

Data ==> Collector
Useful for gathering data from an iterator. It takes data into it and once XEQ is pinged, it outputs n+1 dimensional data. If you send scalar data, 1 dimensional data comes out. The Collector is also used to optimize performance by minimizing the operation of objects. Several labs and examples will follow.

HP VEE

Confirm (OK) and Do



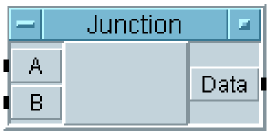
HP VEE: 02DATAOB:0395: 02c
E2110C+24D

The Confirm (OK) object is used to pause execution of a program until there is operator action. The Do object specifies sequential flow of a program. The Do is useful in execution timing with the Timer object.

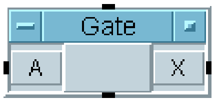
HP VEE

Flow (Data)

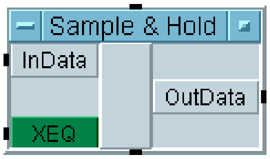
- Junction
  - Wired-OR which sends its most recent input data
  - Often used to send 2 or more data lines to the same input pin
  - Extra inputs are added as DATA inputs
  - Only data object with asynchronous inputs



- 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



HP VEE: 02DATAOB:0395:03
E2110C+24D

As you may recall, an input pin may only have one connection. However, you may occasionally want to connect two outputs to one input. The Junction object does this for you. It acts like a "Wired-OR," sending out the most recently received input data container. If you need to use more than two inputs, simply add more data inputs to the Junction.

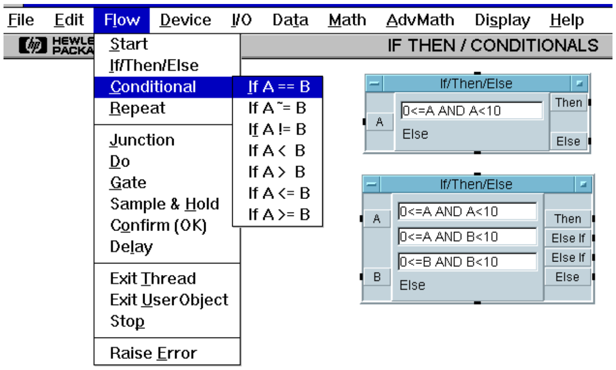
The Gate object is very similar to a latch. It holds its last input until its sequence in pin is activated. If the sequence in pin is not connected, it simply sends the data out immediately.

The Sample & Hold object holds the data and propagates every time the XEQ pin is activated.

Conditional Branching

HP VEE

- 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



If/Then/Else (input A): `0<=A AND A<10` → Then; Else

If/Then/Else (inputs A, B): `0<=A AND A<10` → Then; `0<=A AND B<10` → Else If; `0<=B AND B<10` → Else If; Else

HP VEE: 02DATAOB:0395: 04
E2110C+24D

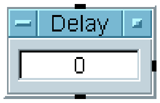
HP VEE provides the If/Then and Conditionals objects so that you can test data and branch accordingly.

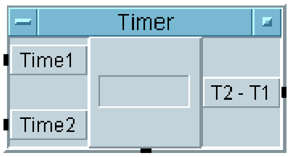
The If/Then object tests according to a user-provided formula. It also allows multiple inputs and the Else/If construct to provide multi-conditional tests.

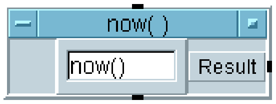
The Conditional are objects preformulated If/Then/Else objects.

Time Related Objects

- Delay
  - Delays propagation for n seconds
- Timer
  - Measures execution time between two objects
- Now
  - Indicates time of execution





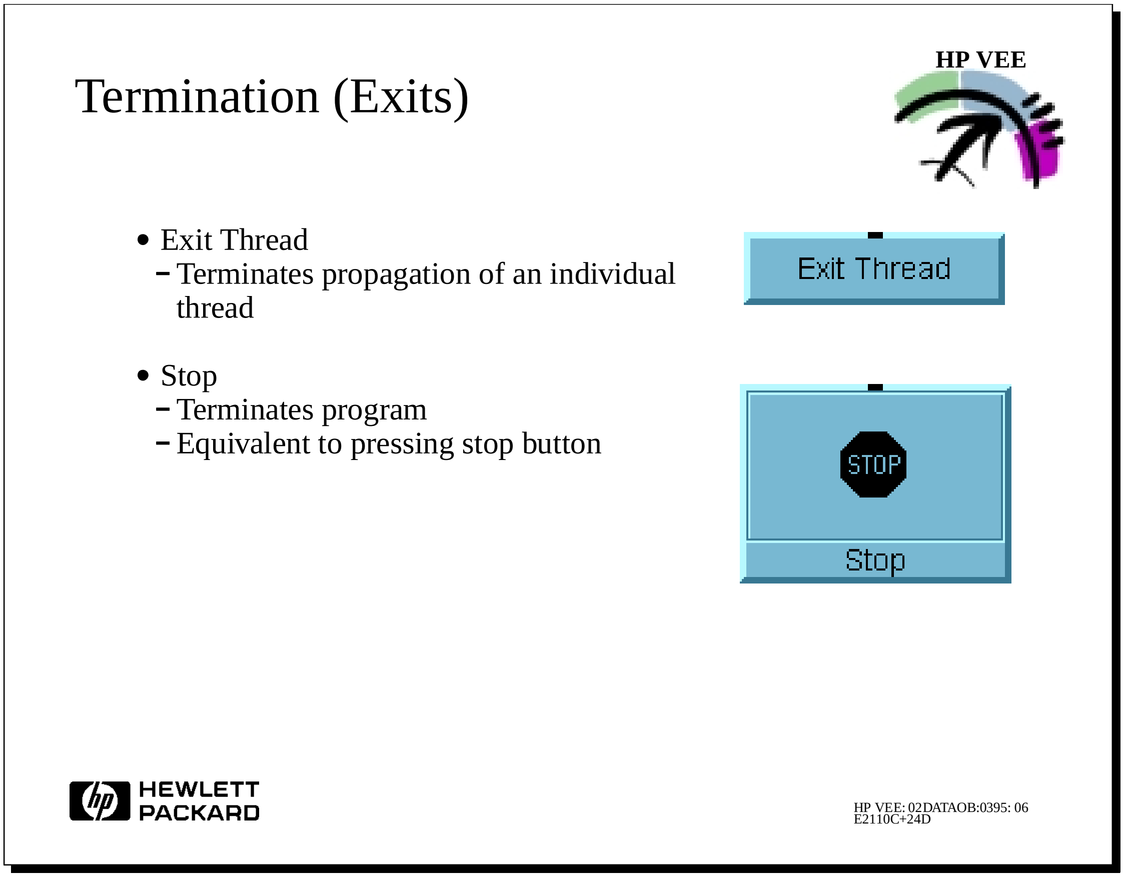


HP VEE: 02DATAOB:0395: 05
E2110C+24D

The Delay object puts the thread to sleep for n seconds, as defined by the user. After that time the thread continues.

The Timer object measures the time between receiving two data containers. You can use this to measure the execution time between two objects.

The Time Stamp has been replaced by now() which provides the real number corresponding to the system real-time clock.



HP VEE

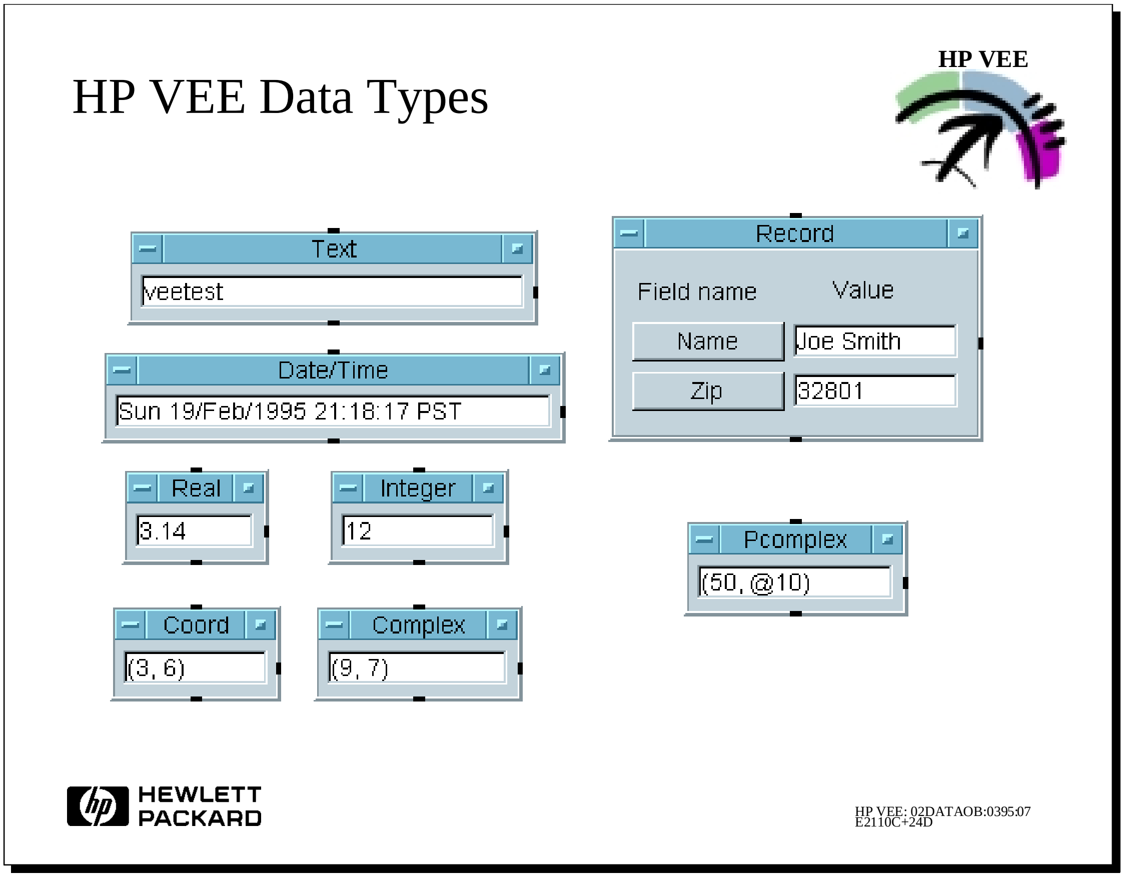
Termination (Exits)

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

Exit Thread
STOP
Stop

HP VEE: 02DATAOB:0395: 06
E2110C+24D

These two objects provide exits. Exit Thread terminates an individual thread. Stop terminates the execution of a program. Note that it stops the program immediately. No other objects operate. It is equivalent to pressing the Stop button.



HP VEE Data Types

Text: veetest

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

Record

Field name	Value
Name	Joe Smith
Zip	32801

Real: 3.14

Integer: 12

Pcomplex: (50, @10)

Coord: (3, 6)

Complex: (9, 7)

HP VEE: 02DATAOB:0395:07
E2110C+24D

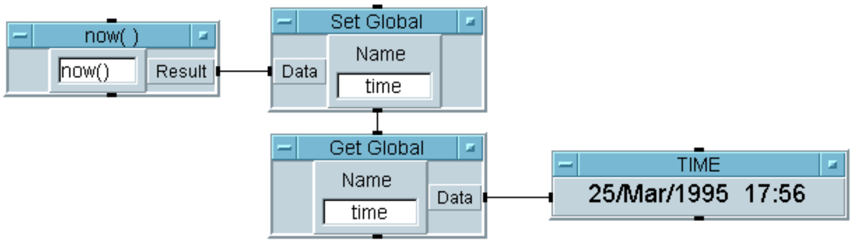
Now let’s look at the various data types and structures implemented by HP VEE. You can see that we provide many useful data types.

Text
Int32
Real
Coord
String
Complex
PComplex
Spectrum
Waveform
Record

HP VEE

Global Variables

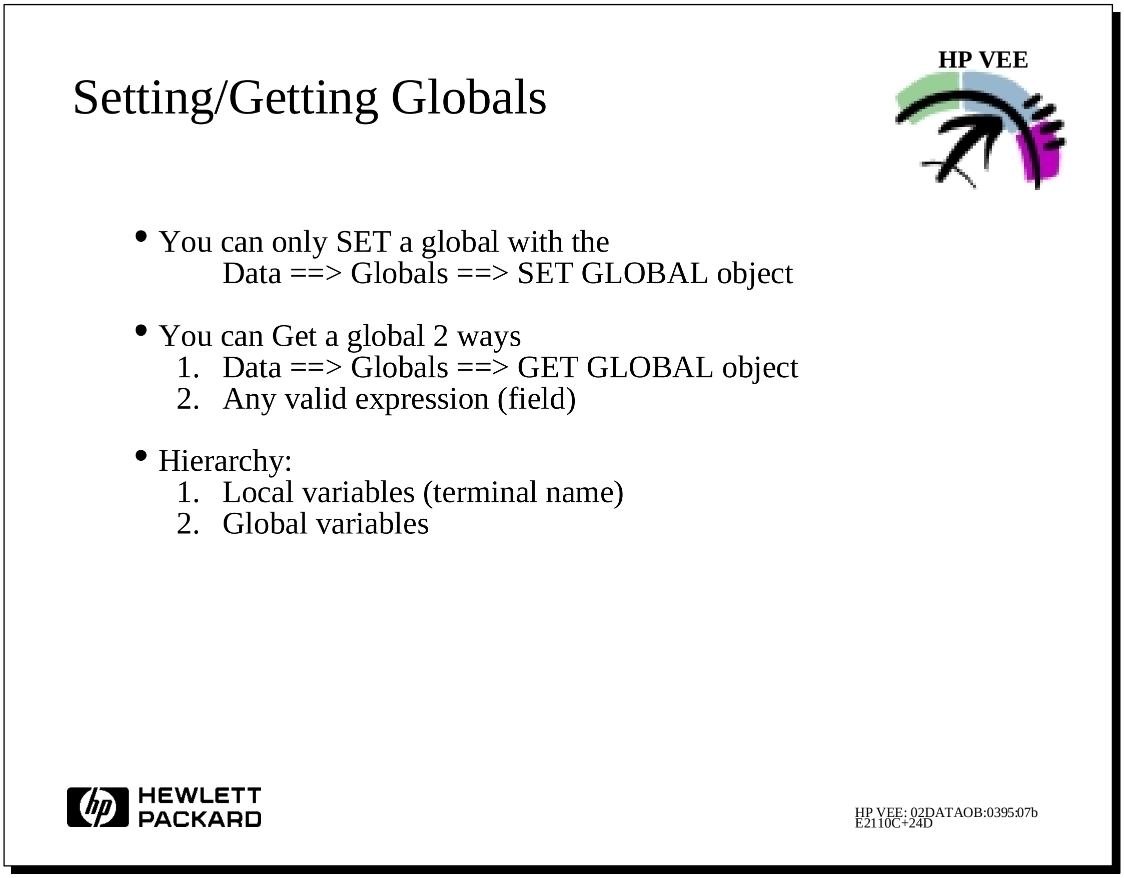
- Allows you to reference data that is maintained in a single place
- Can accept any VEE container



25/Mar/1995 17:56

HP VEE: 02DATAOB:0395:07a
E2110C+24D

Global variables allow you to set or read information that is maintained in a single place within your program. You might want to use this feature to implement a semaphore or flag scheme where the condition of some aspect of your program is available as a program-wide reference. An example would be setting an Operator Name or Serial Number variable that would be used throughout a program.



HP VEE

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

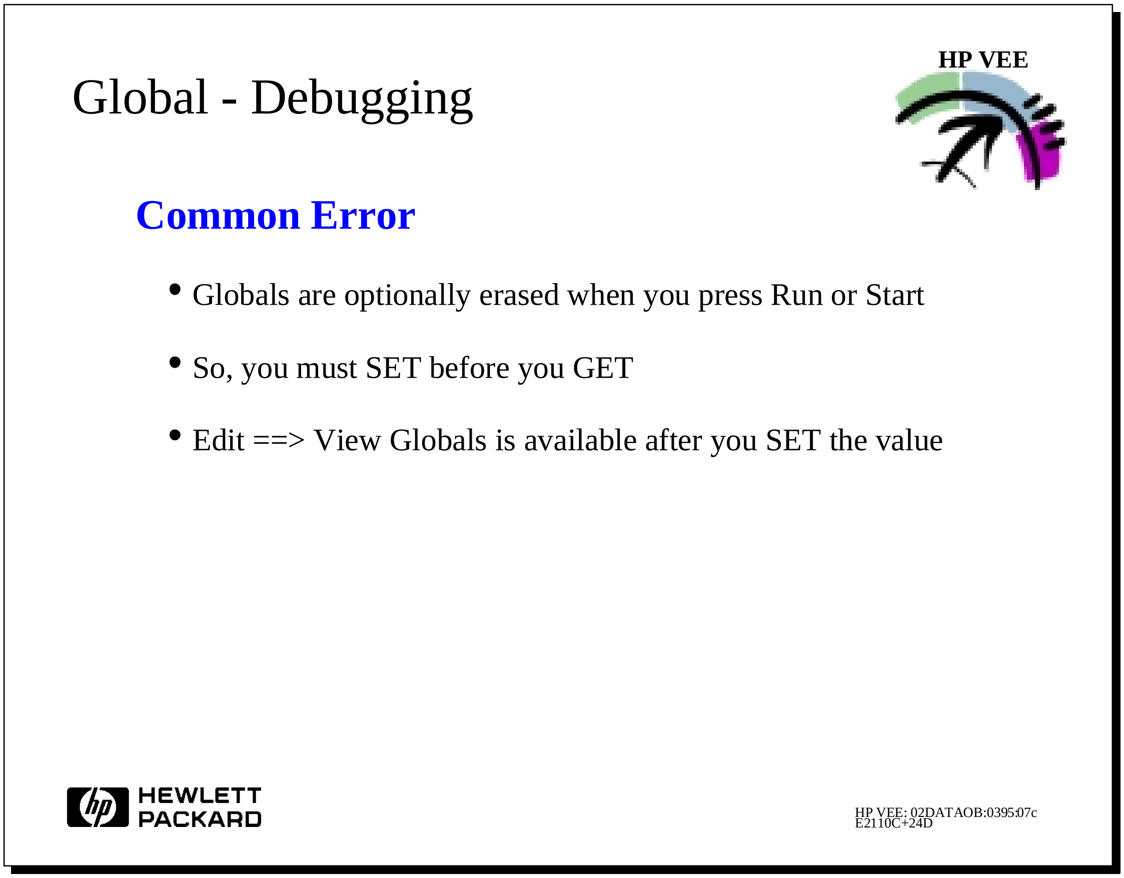
HEWLETT PACKARD

HP VEE: 02DATAOB:0395:07b
E2110C+24D

The object associated with the use of global variables are located under the Data ==> Set Global and Data ==> Get Global menu entries.

Globals are referenced by a name you associate with them. The default name is "globalA". The Set Global object allows you to make a variable, whose name you can choose, take on whatever value shows up at its Data In pin. The Get Global object allows you to propagate the value currently held in your global variable.

Both functions allow you to specify the variable name as either a type-in field or a control input. Globals can also be referenced by name in a Formula box or any other expression field.



HP VEE

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

HEWLETT PACKARD

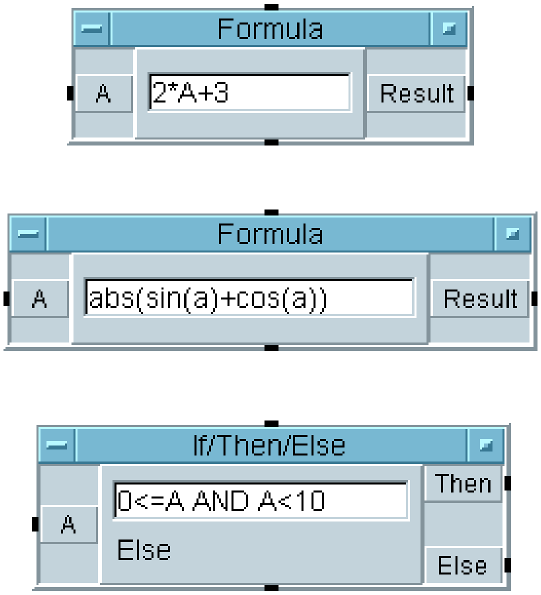
HP VEE: 02DATAOB:0395:07c
E2110C+24D

Globals will be erased when you press START or RUN. The File ==> System Properties dialog has a "Delete Globals at PreRun" flag if you want to keep the value of globals between program runs. It is important to keep in mind that your program's sequencing affects the values stored in globals. You might be certain that a specific global is "SET" before you attempt a "GET". One clear way to see how this works is to have a sequence line run between the Set Global and Get Global object. This prevents your program from trying to access the stored information before the information has been initialized. If we remove the sequence lines, we get an error saying that the global variable we referenced is not defined.

If we take this one step further, we can see that in multi-threaded programs, we could attempt to read a global that has been defined in another thread. If the global hasn't been set before we try to read it, we would get another error.

Located under the Edit menu is a selection to view the contents of your global variables. When you select this function, HP VEE will present a dialog box to allow you to choose the variable you wish to examine.

Formula and Expressions



2*A+3

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

0<=A AND A<10

HP VEE: 02DATAOB:0395:08
E2110C+24D

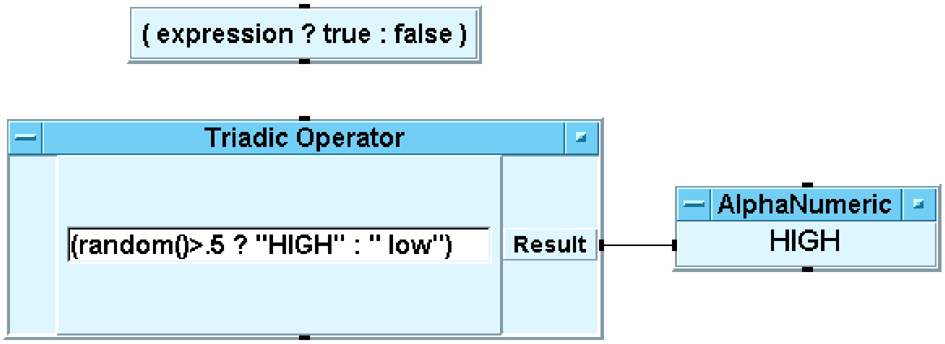
The Formula object accepts any HP VEE Math function including expression evaluation and conditional tests. With this object you can define any math function.

Use the Formula object to increase the efficiency of your program. Formula objects operate most efficiently since it will only execute one object instead of the many objects that would otherwise be needed. For example, instead of using several objects to create the math operation sin(A+(B/C)), simply type the formula into a Formula object.

Many objects have expression field, such as If/Then. These expression fields accept the same syntax as the Formula object.

Triadic Operator

HP VEE



(expression ? true : false)

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

HIGH

HEWLETT PACKARD

HP VEE: 02DATAOB:0395:09
E2110C+24D

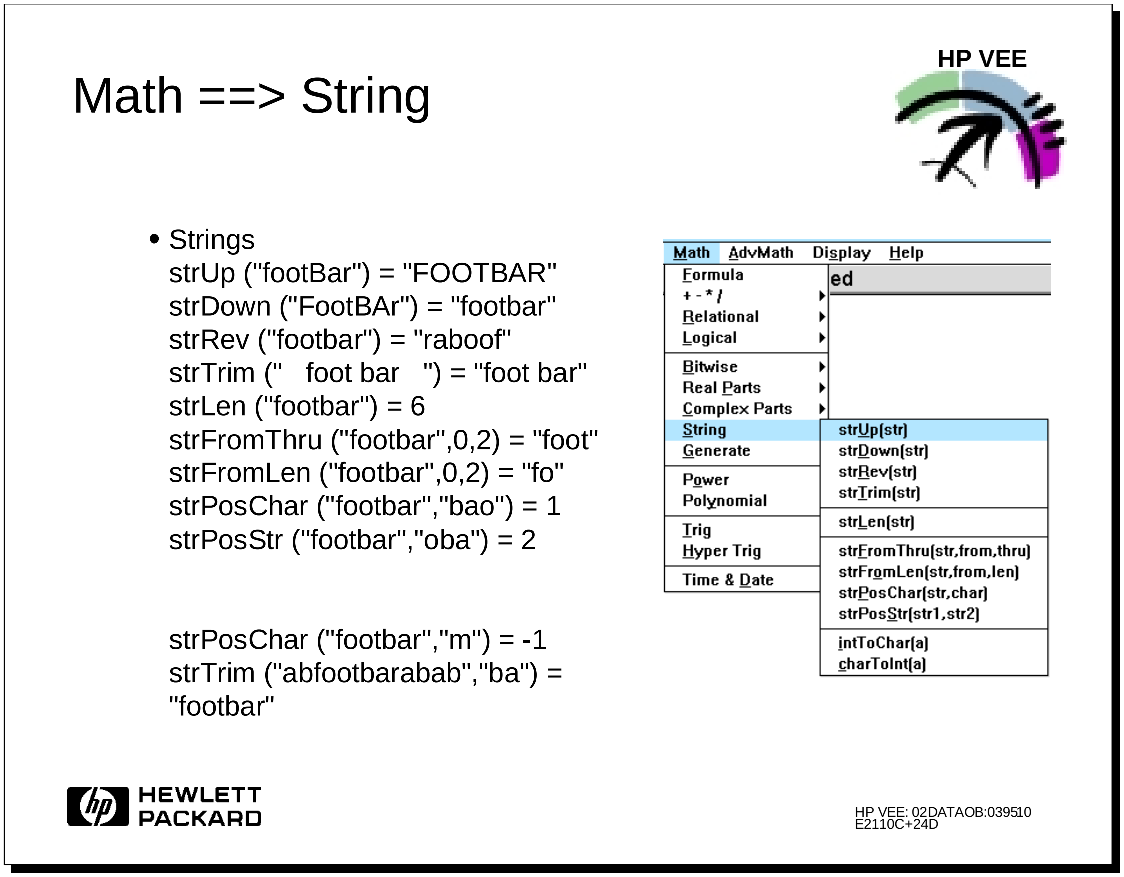
The Triadic Operator is a useful tool for incorporating conditional decision making within an object. Allows you to get the effect of an if/then/else in an expression without the use of an additional object.

They can be nested within themselves.

The format is:

(evaluation expression? true response: false response)

The parenthesis are part of the syntax.



Math ==> String

HP VEE

- 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"

HEWLETT PACKARD

HP VEE: 02DATAOB:039510
E2110C+24D

String Operators

String operations provide the capability to perform comparisons and string manipulation on textual data.

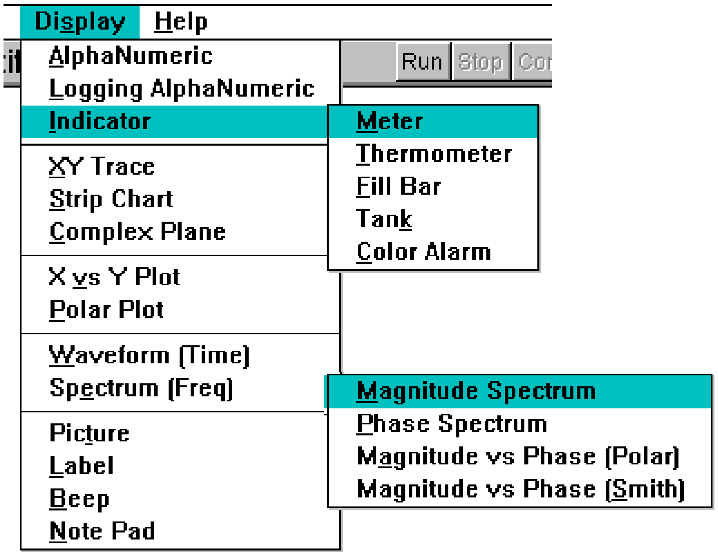
There are a couple of key rules to remember when manipulating strings:

String indexes are zero based

These functions work only on strings and will convert numeric parameters to strings and then perform the appropriate operation.

HP VEE Display Objects

HP VEE



HEWLETT PACKARD

HP VEE: 02DATAOB:0395: 10a
E2110C+24D

AlphaNumeric displays a single data container in its text format

Logging AlphaNumeric displays continuous data containers in their text format

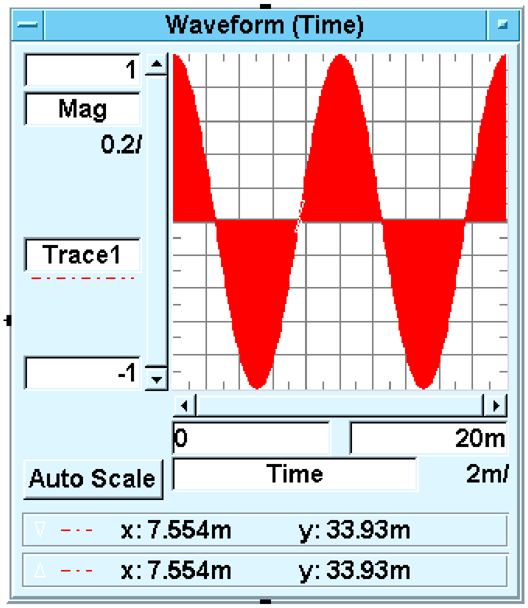
Indicator displays single numeric data generally used to indicate a State, such as *passes* or *failed*

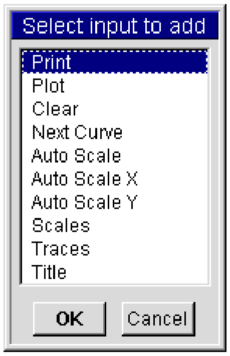
Spectrum displays frequency based information

Waveform displays time based information

HP VEE

Display Control Pins





HEWLETT PACKARD

HP VEE: 02DATAOB:0395: 10b
E2110C+24D

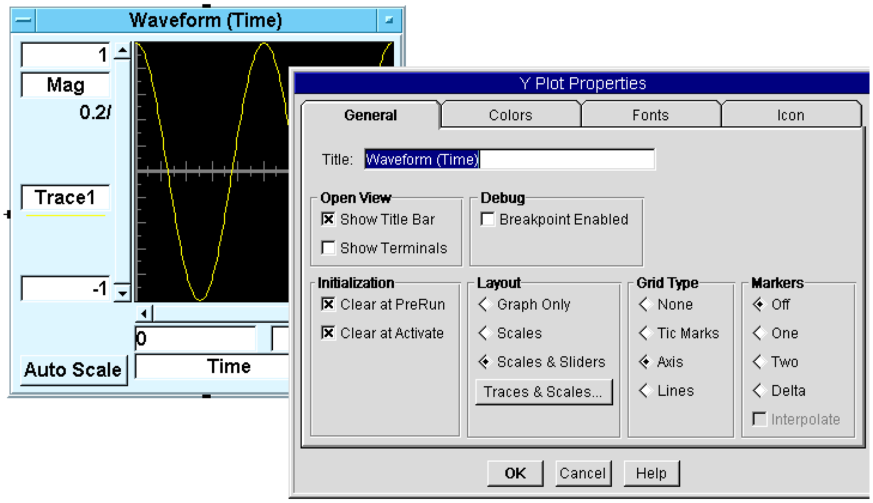
You can customize the display objects in many ways. For multiple graphs, add more data inputs. Clear and Auto Scale can be control inputs.

Plot multiple data instances by using the Next Curve control.

You can magnify the view of the data by using the Zoom command.

HP VEE

Display Properties



Y Plot Properties

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

HEWLETT PACKARD

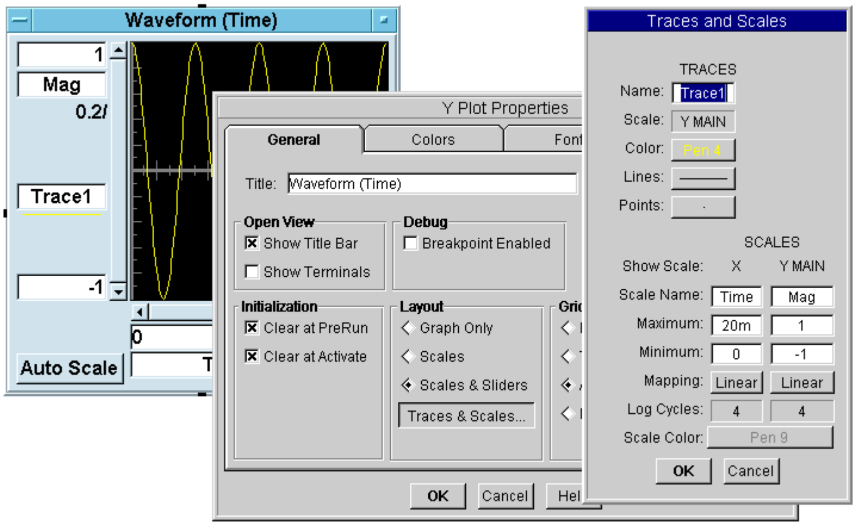
HP VEE: 02DATAOB:0395: 10c
E2110C+24D

You can also modify the panel layout, grid type, trace color and line type used on the displays.

Note that many of these functions may be added as control pins to the object.

Display Traces and Scales

HP VEE



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

HP VEE: 02DATAOB:0395: 10d
E2110C+24D

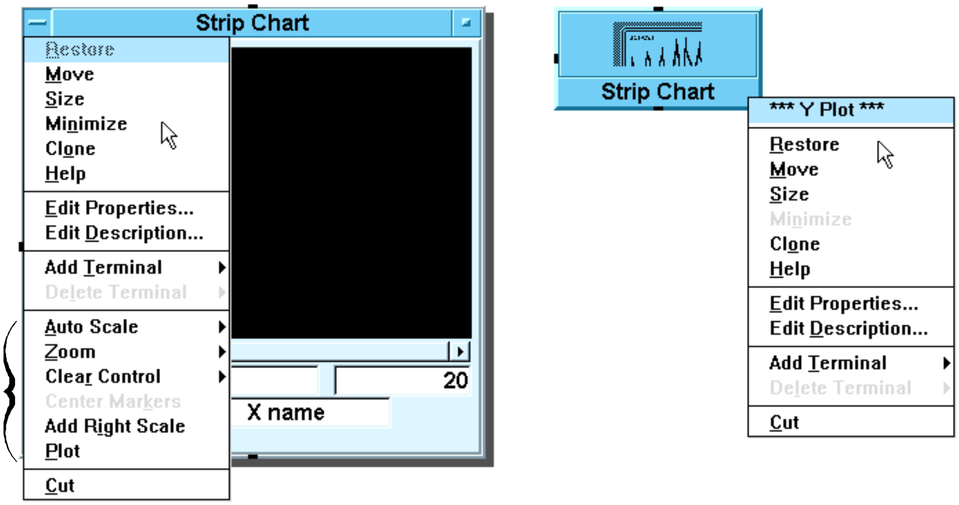
You can modify the Trace Name, Color, Line type and for Scales, Name, Mapping type ...

HP VEE

Object Menu

OPEN VIEW

ICON VIEW



HEWLETT PACKARD

HP VEE: 02DATAOB:0395: 11
E2110C+24D

The object menu allows you to interact with an object. Note that it is slightly different between the icon view and the open view. With this menu you can:

Change the size of the object

Move it

Clone it

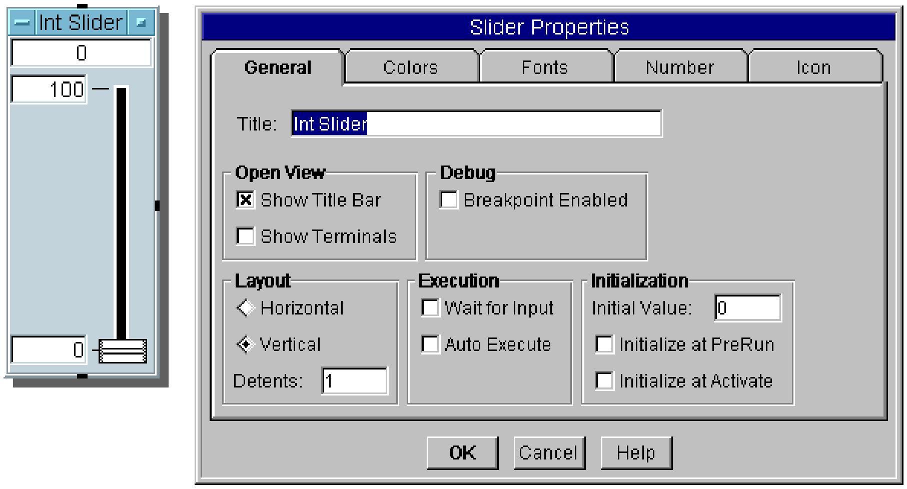
Change the properties of an object
- change the colors/fonts used
- change the layout
- set and clear a breakpoint
- other object specific operations

Show the user comments about the object (Show Description)

Add Pins and Terminals

Cut the object

Object 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: 02DATAOB:0395:12
E2110C+24D

Properties allow you to customize an object’s appearance and/or behavior.

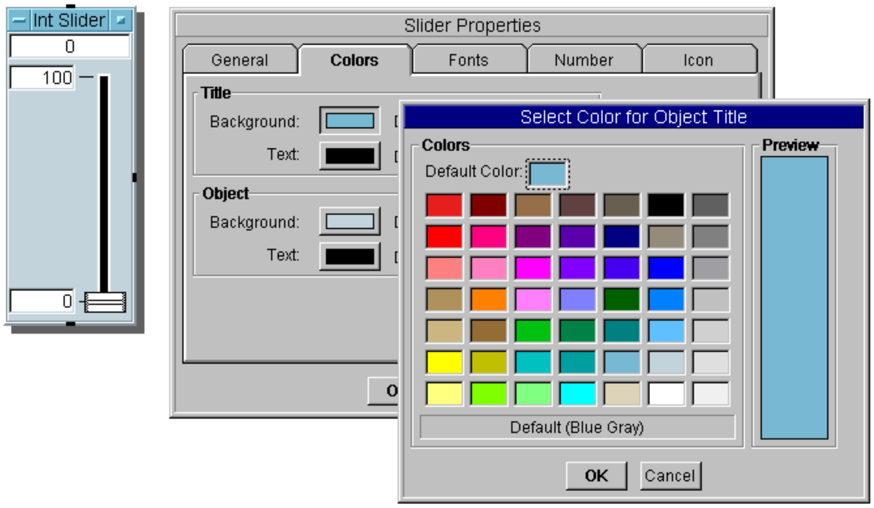
All objects have properties to control attributes generic to objects.

In addition, objects may include properties specific to a particular object.

You can get to the Edit Properties... dialog by "double clicking" on the object title bar. The General tab is where you set break points for use in debugging.

HP VEE

Object Color



HEWLETT PACKARD

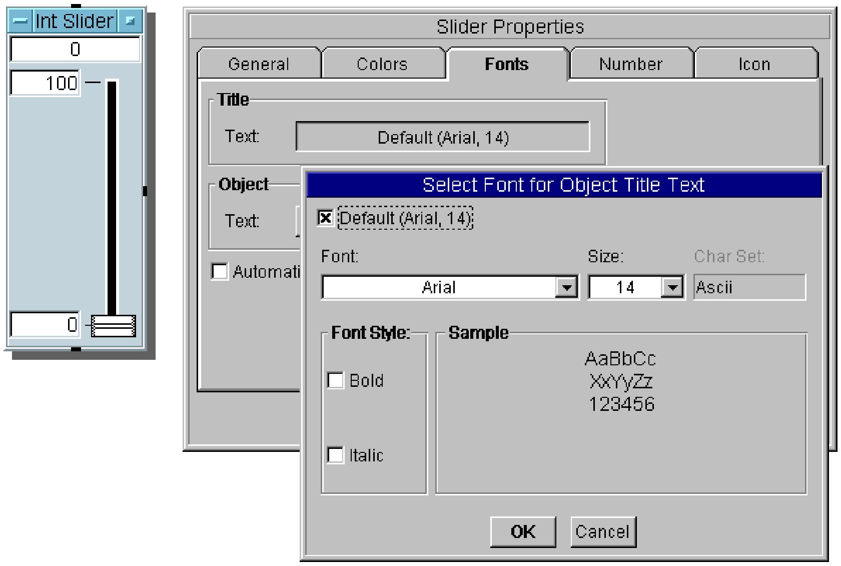
HP VEE: 02DATAOB:0395:13
E2110C+24D

You can use the Colors properties tab to assign a color to each area of an object.

Each related area, such as the title bar, has the color control in a titled view to help organize the attributes.

Object Font

HP VEE



Int Slider: 0, 100, 0

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

HEWLETT PACKARD

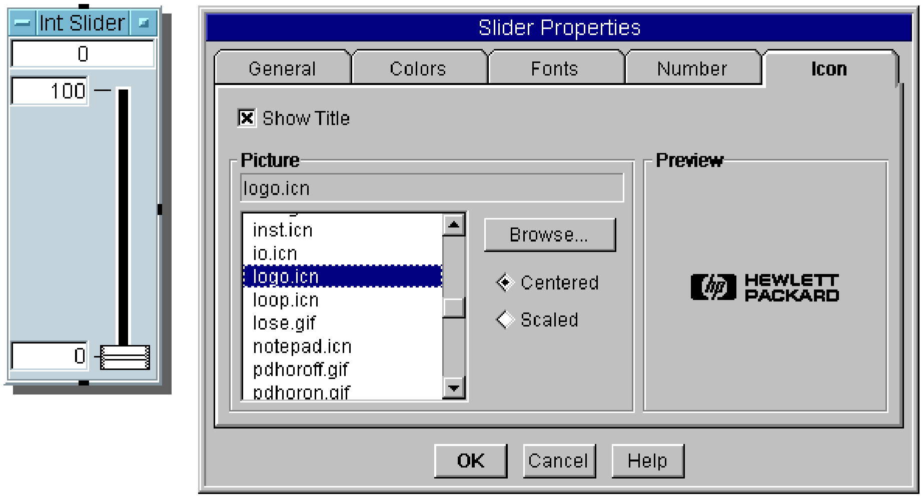
HP VEE: 02DATAOB:0395:14
E2110C+24D

As with the Colors properties tab, the Fonts tab allows you to apply a font characteristic to each area of an object. It will show all fonts available on your system that HP VEE can use.

The Font dialog displays the character set associated with the language you have chosen for your environment. For instance, when using Japanese Windows, the Properties tab displays Kanji in the char set field.

Object Icon

HP VEE


Int Slider

0

100

0

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: 02DATAOB:0395:15
E2110C+24D

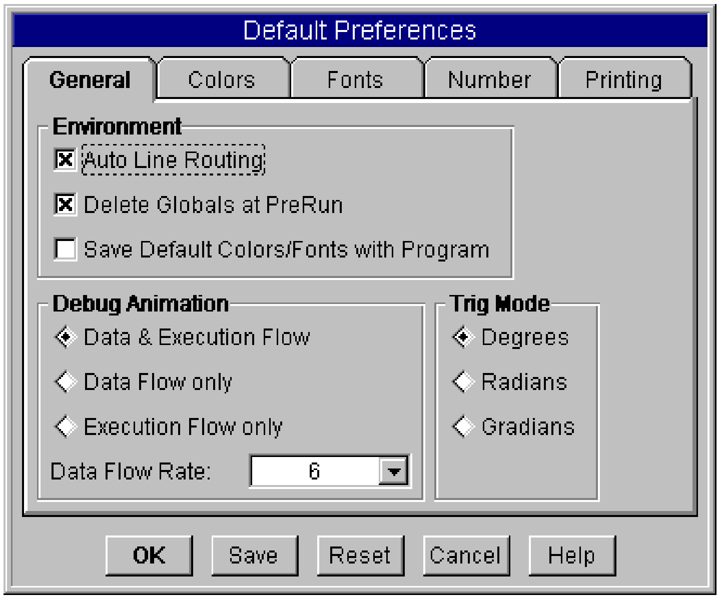
You can assign a graphical element to be displayed in the icon view of any object. The supported graphics file types are:

MS Windows	X Windows
.gif	.gif
.bmp	.bmp
	.xwd (x bitmap file)
.icn	.icn (x icon file)

You can allow the icon to stretch or shrink to fit the size of your iconized object.

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

HP VEE: 02DATAOB:0395:16
E2110C+24D

Program Properties

File ==> Edit Default Preferences...

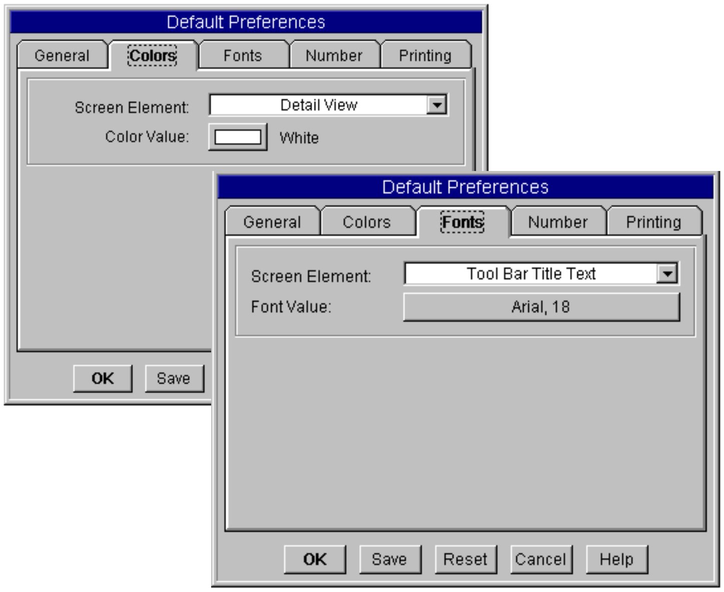
This dialog allows you to set the appearance and/or behavior of your VEE program.

You can make the current settings affect all of your VEE programs by pressing the "Save" button in the dialog. This saves the settings to your .veerc or vee.rc file.

In addition, you can override a program’s color and font attributes by checking the "Save Default Colors/Fonts with Program" box.

HP VEE

Colors and Fonts



Screen Element: Detail View

Color Value: White

Screen Element: Tool Bar Title Text

Font Value: Arial, 18

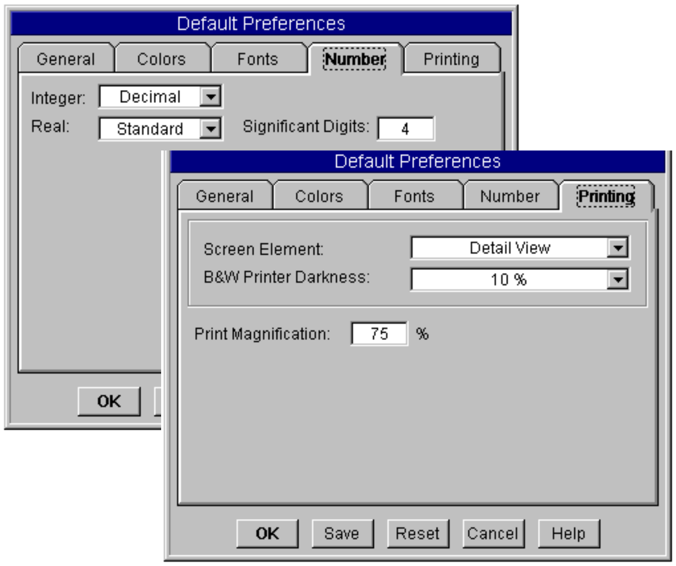
HP VEE: 02DATAOB:0395:17
E2110C+24D

The Fonts tab on the program properties dialog allows you to set the fonts associated with the VEE editor window.

The "Screen Element" field is a drop down list of area whose fonts you may change.

Numbers and Printing

HP VEE



Integer: Decimal

Real: Standard

Significant Digits: 4

Screen Element: Detail View

B&W Printer Darkness: 10 %

Print Magnification: 75 %

HEWLETT PACKARD

HP VEE: 02DATAOB:0395:18
E2110C+24D

The Numbers tab in the program properties dialog allows you to control how numeric data is displayed. It affects things like the AlphaNumeric display. It does NOT affect calculations, which are ALWAYS done at full resolution.

The Printing tab allows you to adjust the appearance of your program hard copies.

Lab 2a

Apple Bagger

Objective: To begin to learn the concept of **data flow** with an interator. In addition, we will continue to learn where objects are located.

Let’s figure out how many apples it takes to fill a **ten** pound basket. Create an HP VEE program that **counts** how many apples it takes to fill the basket. Each apple should **randomly** weigh between **0** and **1** pound.

Suggestions:
This program can be created with 8 or fewer objects. Choose from the following:

Until Break
Random Number
Accumulator
Break
Constant Real
Conditional (A>=B)
Stop
Counter
If/Then/Else

Hints:

As the weight of the apples is accumulated, check to see if the total accumulated weight is greater than 10 pounds. Once the weight is greater than 10 pounds, check to see how many apples were in the bag just before it went over 10 pounds.

Lab 2b

Testing Numbers

Objective: To learn to use **Flow Control** objects in a decision making process.

Step 1
Create a program that allows the user to enter a number between 0 and 100. If the number is greater than or equal to 50, display the number. If it is less than 50, display the message "Sorry".

Suggestions:
This program can be created with 7 or fewer objects. Choose from the following:

Integer
Slider
Real
If/Then/Else
Formula
Gate
Text
Junction
Alphanumeric

Hints:

There are many ways to solve this problem. One way could take advantage of the fact that **A>15** evaluates to a **1** when true and a **0** when false. So you could also say **(A>15)*A** and; for example, when A is 34, the output result would be 34. When A is 3 the output result is 0. This could be useful in an If/Then/Else object.

Step 2
After the program is working with 7 objects or fewer, try the following:

Instead of using the **Flow Control** objects such as **If/Then/Else** or **Gate**, use the **Formula** object with a **Triadic Operator**.

Hints:

The **Triadic Operator** is not an object under a VEE pull-down menu. It is an expression that can be used in objects like the **Formula** object or the **If/Then/Else** object etc **Search for help on** the **Triadic Operator** for more details.

Lab 2c

Random Number Generator

Objective: To learn that objects have optional input terminals that enable a user to programmatically control object variables.

Step 1
Create a random number generator that has user controlled external inputs. Plot the random numbers on a **Stripchart**. The user controlled inputs should be allowed for:

Maximum random number

Minimum random number

Number of random numbers generated

Step 2
Using a VEE object, determine the time it takes to generate and to display the random numbers.

Note: Carefully select what you are timing! Improper connections can lead to erroneous results.

Step 3
Modify the program to **Collect** the random numbers into an array before plotting them. Plot the array of random numbers and also plot the "**Moving Average**" of those same random numbers on the same **Stripchart**.

Hint:

Help on **Collector** and **Moving Average** would be wise.

Lab 2d

String Functions and Global Variables

Objective: To learn about the string functions and the use of **Global** variables.

Step 1
Create a program that expects a users' name to be typed into a Text Constant object or use a **Data=>Dialog Box=>Text Input.**

Assume the name will be space <firstname> space <lastname>

After the user enters their name, have the program strip off the first name and only print the last name.

Note: You can use **Math=>Strings** functions or a **Formula** box.

Step 2
Store the string into a **Global Variable.**

Retrieve the string using a **Formula** box.

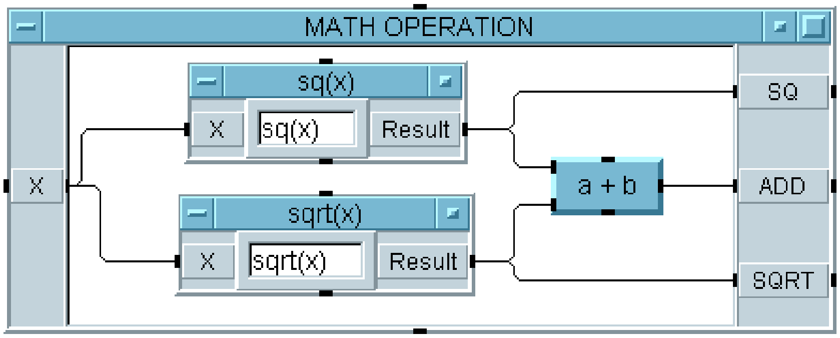
UserObjects

Introduction to HP VEE
V3.1 ©1996

HP VEE

HP VEE UserObjects

- Provides a work area within an object



- A context

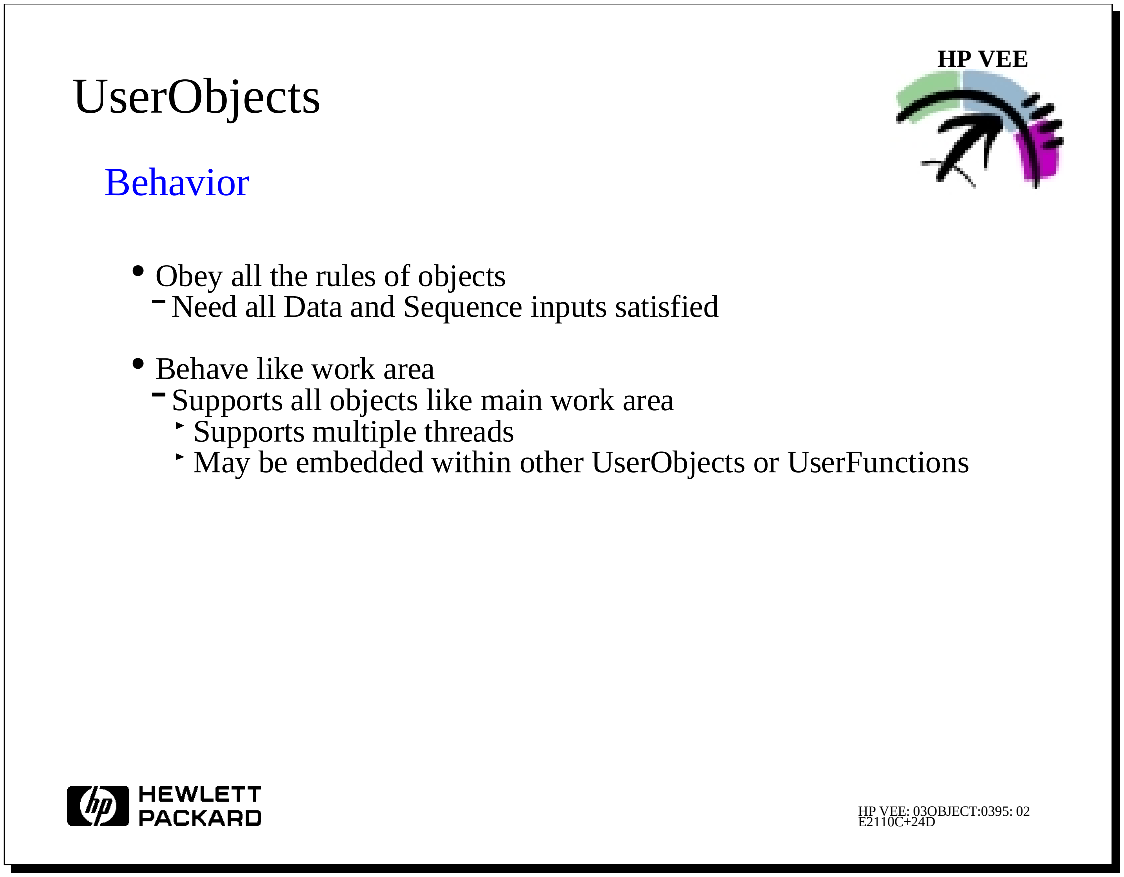
HEWLETT PACKARD

HP VEE: 03OBJECT:0395: 01
E2110C+24D

A UserObject provides you with a work area within a work area. You build programs inside it that are completely independent of the work area outside of it. In fact, you can run a program inside the UserObject without affecting anything else in your work area. They create a new context, or working environment.

These are most commonly used to build your own "custom" object, and can be thought of as an in-line subroutine.

You can assemble a UserObject to perform a common task, then shrink it to an icon for use throughout the current and future VEE programs.



HP VEE

UserObjects

Behavior

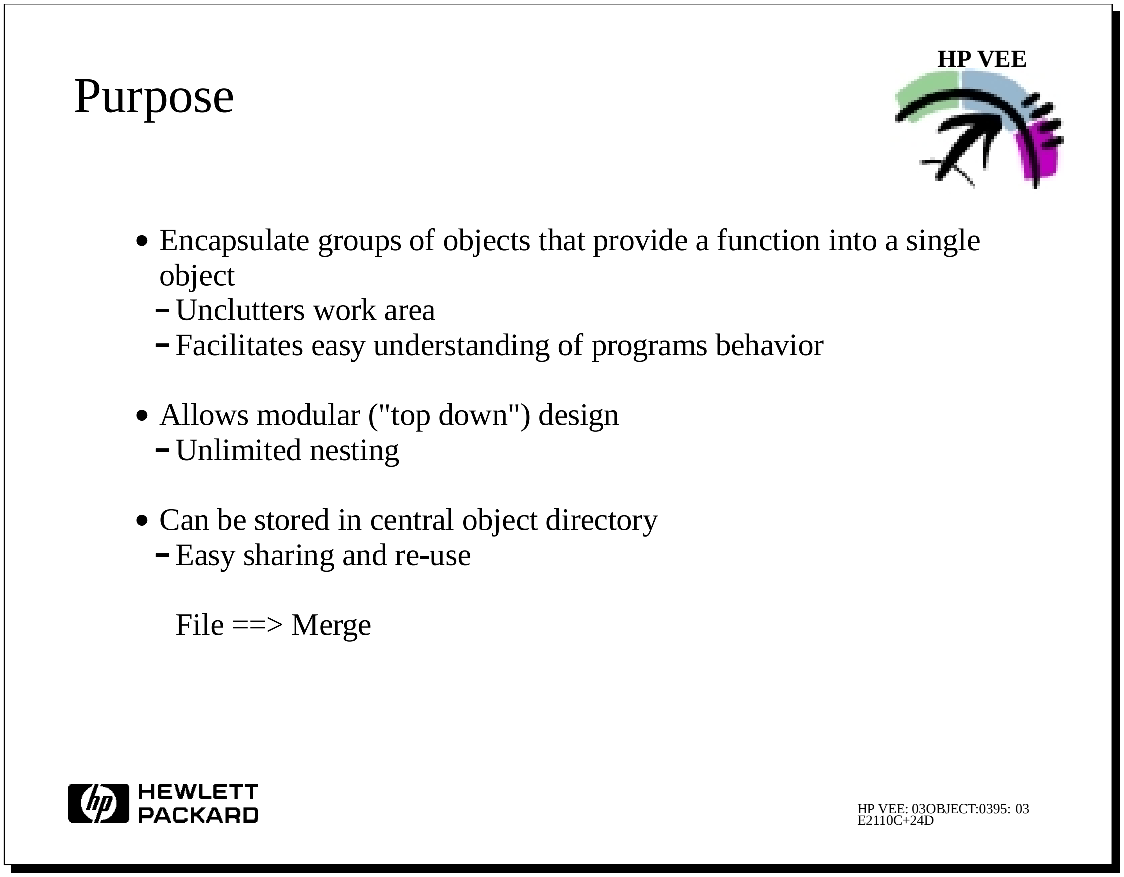
- Obey all the rules of objects
  - Need all Data and Sequence inputs satisfied
- Behave like work area
  - Supports all objects like main work area
    - Supports multiple threads
    - May be embedded within other UserObjects or UserFunctions

HEWLETT PACKARD

HP VEE: 03OBJECT:0395: 02
E2110C+24D

UserObjects obey all the same rules as other objects, namely

They must have all data and sequence inputs satisfied before operating. They behave exactly like the general work area, supporting all HP VEE objects and multiple threads.



Purpose

HP VEE

- 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

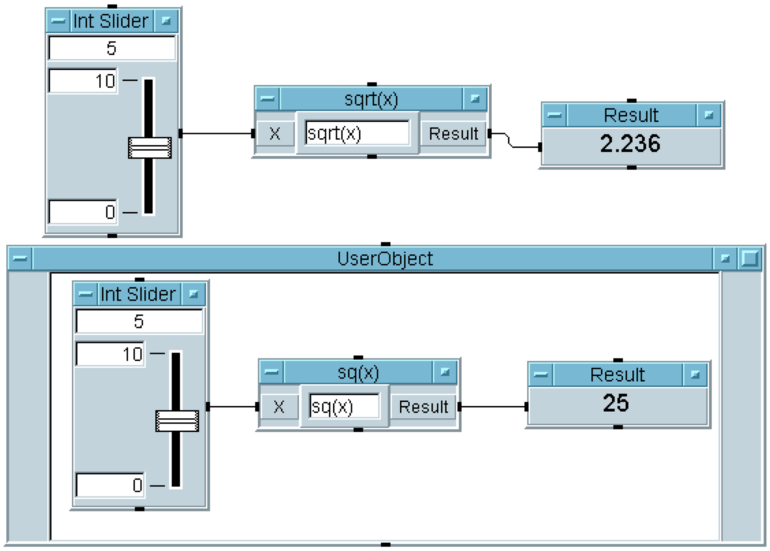
HEWLETT PACKARD

HP VEE: 03OBJECT:0395: 03
E2110C+24D

UserObjects are quite helpful in the development of HP VEE programs. They permit you to encapsulate groups of objects that provide a function into a single object; uncluttering the work area. This also helps to document the program, make understanding the program's behavior easier and facilitates "top-down" design. You can nest an unlimited number of UserObjects. So create your program as a block diagram. Then simply fill in each UserObject with the necessary functions.

Parallel Thread Example

HP VEE



HEWLETT PACKARD

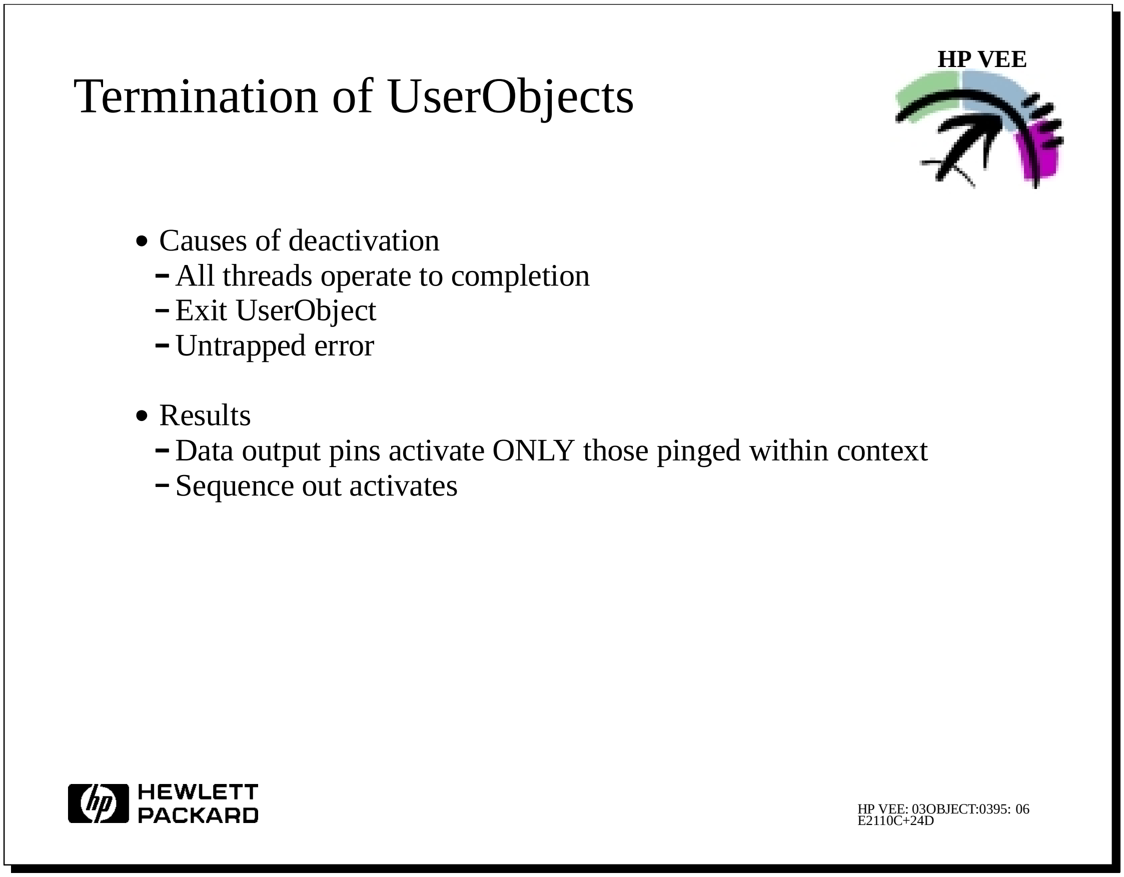
HP VEE: 03OBJECT:0395: 05
E2110C+24D

Here is an example of how the propagation engine shares time between parallel threads, where a UserObject contains one of the threads.

Note that the propagation engine treats each object inside the UserObject as a primitive object and executes them one at a time.

The UserObject operates in the following fashion:

> It activates only when all data and sequence inputs are satisfied, even if one of its multiple internal threads only requires one of the data inputs. All internal threads activate at once.
>
> Only propagates its output pins once per UserObject execution, even if the output pins are pinged multiple times.



HP VEE

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

HEWLETT PACKARD

HP VEE: 03OBJECT:0395: 06
E2110C+24D

A UserObject terminates operation for the following reasons:

ALL threads completed execution

Thread execution encounters an Exit UserObject device

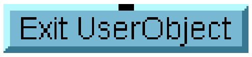
A thread causes an untrapped error.

When the UserObject terminates operation, it activates only its data output pins that received data from within the UserObject. It then activates its sequence out pin.

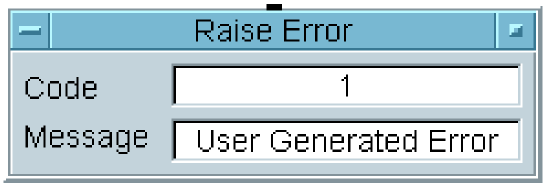
HP VEE

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

HEWLETT PACKARD

HP VEE: 03OBJECT:0395: 07
E2110C+24D

As mentioned before, UserObject terminates operation when it encounters an Exit UserObject. At that point:

All threads in the UserObject terminate operation

Output pins that have received data activate

Sequence out activates

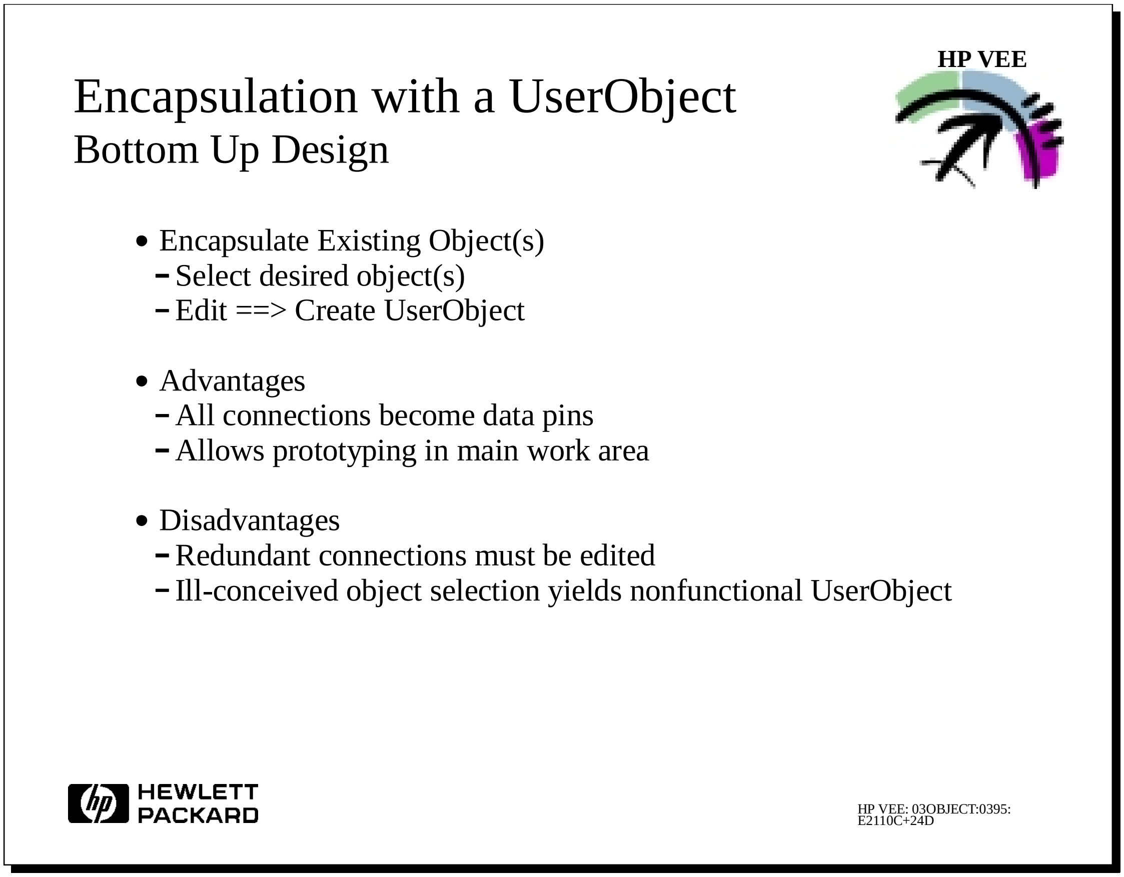
UserObjects will also stop operation when they encounter a "Raise Error". This provides a user-generated error condition. At that time:

All threads in the UserObject terminate operation

NO output pins activate

The Error pin (if one exists on this UserObject) generates the escape code from the "Raise Error" object. If there is no Error pin, an error dialog box appears.

Note that you can allow error messages to propagate up through nested UserObjects.



HP VEE

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

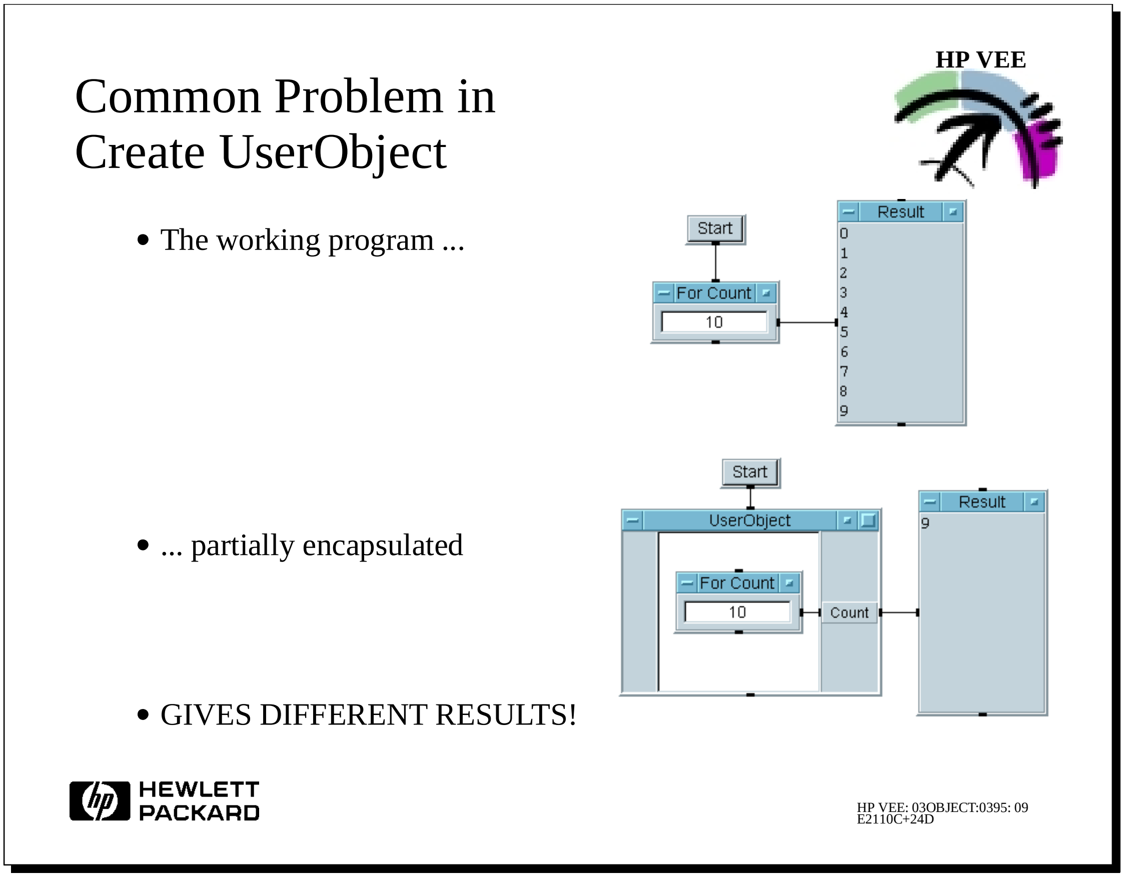
HEWLETT PACKARD

HP VEE: 03OBJECT:0395:
E2110C+24D

There are two methods of creating UserObjects. In the first method, you select your desired objects, create a working function and then encapsulate it in a UserObject.

The advantage to this method is that all connections become data pins automatically. It also allows you to prototype in the main work area.

This method also presents some disadvantages. Encapsulating a function often creates redundant data pin connections which must be deleted. Also, ill-conceived object selection and the misunderstanding of how an object operates often yield a non-functioning UserObject. "It just doesn’t work like it did before." We’ll illustrate this with the following slide.



Common Problem in Create UserObject

HP VEE

- The working program ...
- ... partially encapsulated
- GIVES DIFFERENT RESULTS!

HEWLETT PACKARD

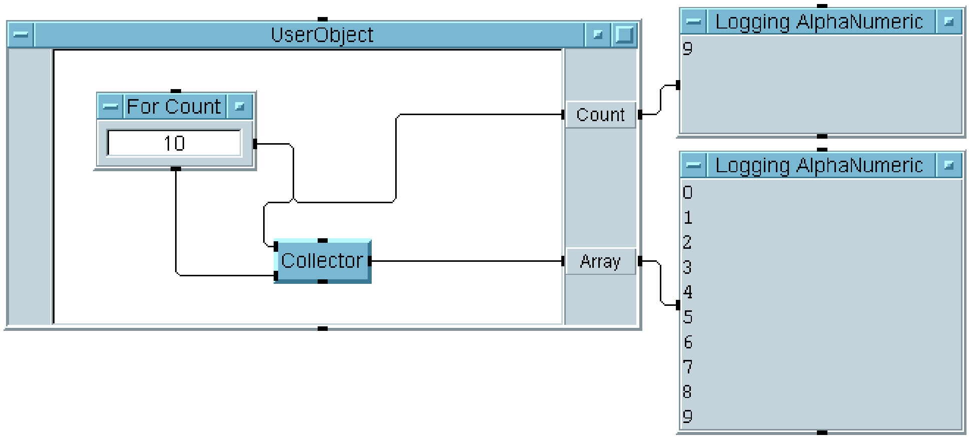
HP VEE: 03OBJECT:0395: 09
E2110C+24D

In the example listed above, the UserObject will only send a single output container, so, it will only output the final value.

Remember that objects only operate once and only provide one data container.

The next slide illustrates how to handle iterative data within a UserObject.

UserObject With Collector

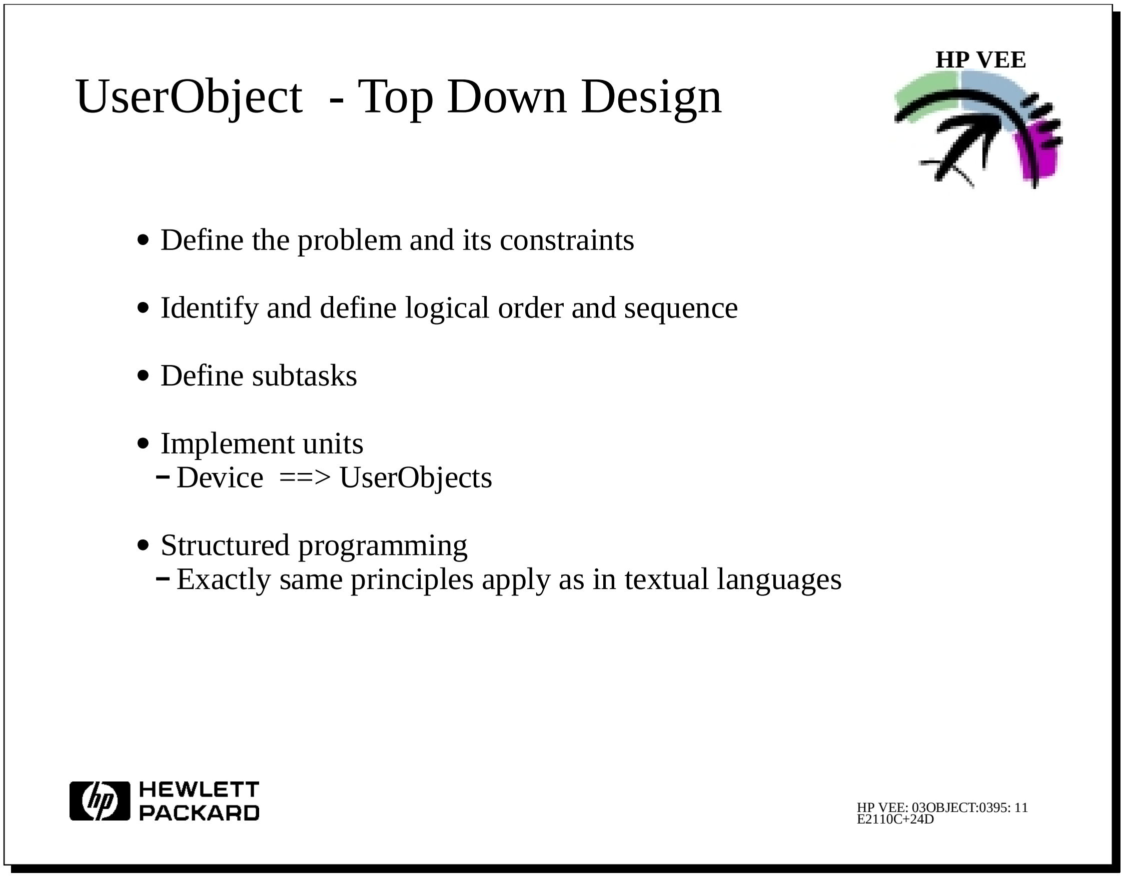


```
9
```

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

HP VEE: 03OBJECT:0395: 10
E2110C+24D

To bundle data within a UserObject, a Collector is used to collect the data to pass out of the UserObject as a single container of Array data.



UserObject - Top Down Design

HP VEE

- 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

HEWLETT PACKARD

HP VEE: 03OBJECT:0395: 11
E2110C+24D

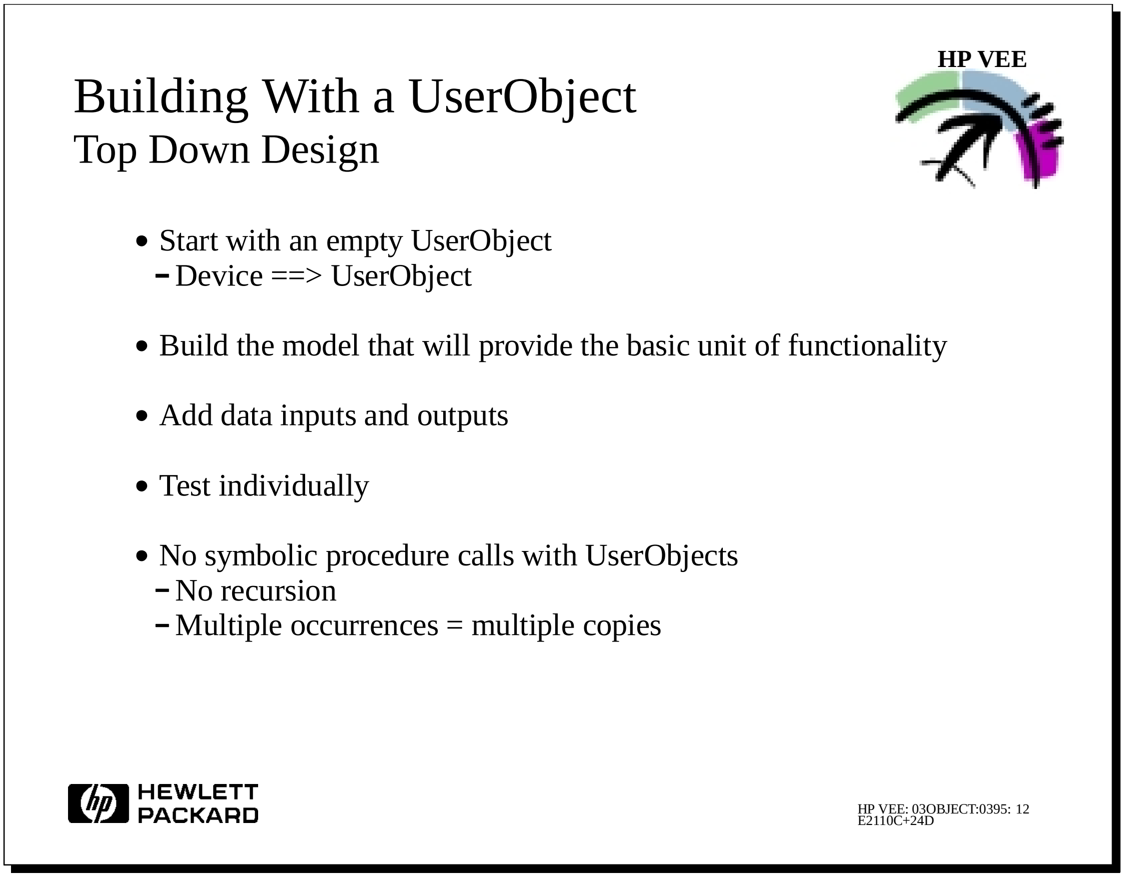
Let’s review the Top-Down design methodology.

Define the problem, including all of its constraints

Identify a logical order and sequence for each of the tasks within the problem

Define each subtask within each task. Note how well this fits into HP VEE’s paradigm of nested UserObjects

Now implement each of these tasks as a UserObject within HP VEE.



HP VEE

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

HEWLETT PACKARD

HP VEE: 03OBJECT:0395: 12
E2110C+24D

Start with empty UserObjects: Use them as stubs or empty boxes

Build the program that provides basic functionality

Add data inputs and outputs

Test each UserObject individually. Do this with a Start object inside the UserObject, run only those devices in it.

Note that HP VEE does not support recursion in UserObjects. When you need multiple occurrences of an object, you must make multiple copies or promote UserObjects to UserFunctions.

Shortcut: Connecting an object outside the UserObject with an Object within, directly connect the data out pin with the data in pin and HP VEE will make the UserObject connection automatically.

Lab 3a

UserObjects and Data Handling

Objective: To learn about data handling to improve speed and to understand the concept of **UserObjects.**

Step 1
For this lab, we will build an HP VEE program several ways and note the differences in their execution speeds.

Create a program that sends the **RANGE (0 to 710, steps of 10**) through a **SINE** and **COSINE** function. Do **not** use a **Device=>Virtual Source=>Function Generator**.

Display the results of the functions in an **X vs Y** display.

Use a **TIMER** to determine how long the program takes to execute.

Step 2
Now, **clone** that entire example and place it below the first one. Change the clone so that the **RANGE** data is **COLLECTED** into an array. Take the collected data into the **SINE** and **COSINE** functions and then plot that data on the **X vs Y** display. Note the new time of execution.

Step 3
Let's take a look at **UserObjects**. Save Steps 1 and 2. Do this in two parts. **SELECT** the objects in thread #1 to **Create UserObject**. This will include the **RANGE**, **SINE** and **COSINE**.
On thread #2, **SELECT** the **RANGE, SINE, COSINE** and **COLLECTOR** in the **UserObject.**

What is the difference in the way each thread operates and why?

Lab 3b

Mask Test with UserObject

Objective: To build and utilize a new **UserObject** that is built from **Device=>UserObject**. All objects except **Noise Control** and **Display** should be in the UserObject.

Your VEE program should plot a noisy sine wave, a mask limit line and mark any point over the limit line with a diamond. The Graph will visually show you the data, but you will need to programmitically check the limits with a Comparator object

Step 1
Create a 50Hz sine wave with a user controllable amount of noise. Use the **Device=>Virtual Source=>Function Generator** and **Noise Generator**

Step 2
Build your mask limit line using data points from **Data=>Constant=>Coord** with these values:

(0, 0.5)
(2.2m, 1.2)
(7.2m, 1.2)
(10.2m, 0.5)
(20m. 0.5)

Hint:

The **Coord** object has some userful **Properties** that will help with making a scalar **Coord** turn into an array **Coord**. This coordinate array is used to plot the limit line and also as the reference values for the comparator.

Step 3
If any point from the noisy sine wave exceeds the limit line, mark that failing point with a red diamond.

Hint: You can change the trace format of the display from lines to dots and the points from dots to diamonds. Look under the **Traces** and **Scales** menu option on the display's **Edit Properties.**

You may also find the **Device=>Comparator** object essential for this lab.
(Don't forget to read **Help on the Comparator**!))

Lab 3c

Random Noise UserObjects

Objective: Create a **UserObject** that generates a random noise waveform and compare it to the Virtual Function Generator's data.

Step 1
Create a **UserObject** that generates a random noise waveform. Provide control outside of the **UserObject** for the following:

Amplitude, Number of Points, Interval (timespan), and DC offset

Step 2
Plot the noisy waveform on a **Waveform(Time)** object and the noise spectrum on a **Magnitude Spectrum** object outside of the **UserObject**.

Note: Do **NOT** use a **Virtual Source** inside the **UserObject! Build Waveform** and **Random Number**, etc. are recommended.

Step 3
Compare your noisy waveform generator with the **Device=>Virtual Source=>Noise Generator** by plotting both traces on the same graphs.

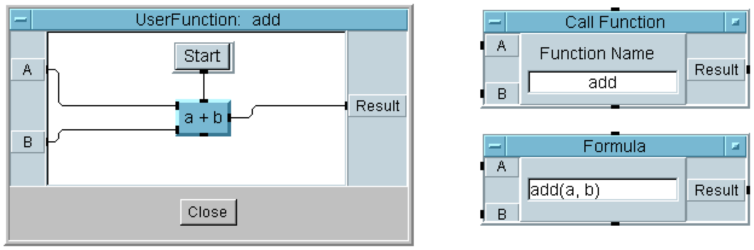
HEWLETT
PACKARD

UserFunctions

HP VEE

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

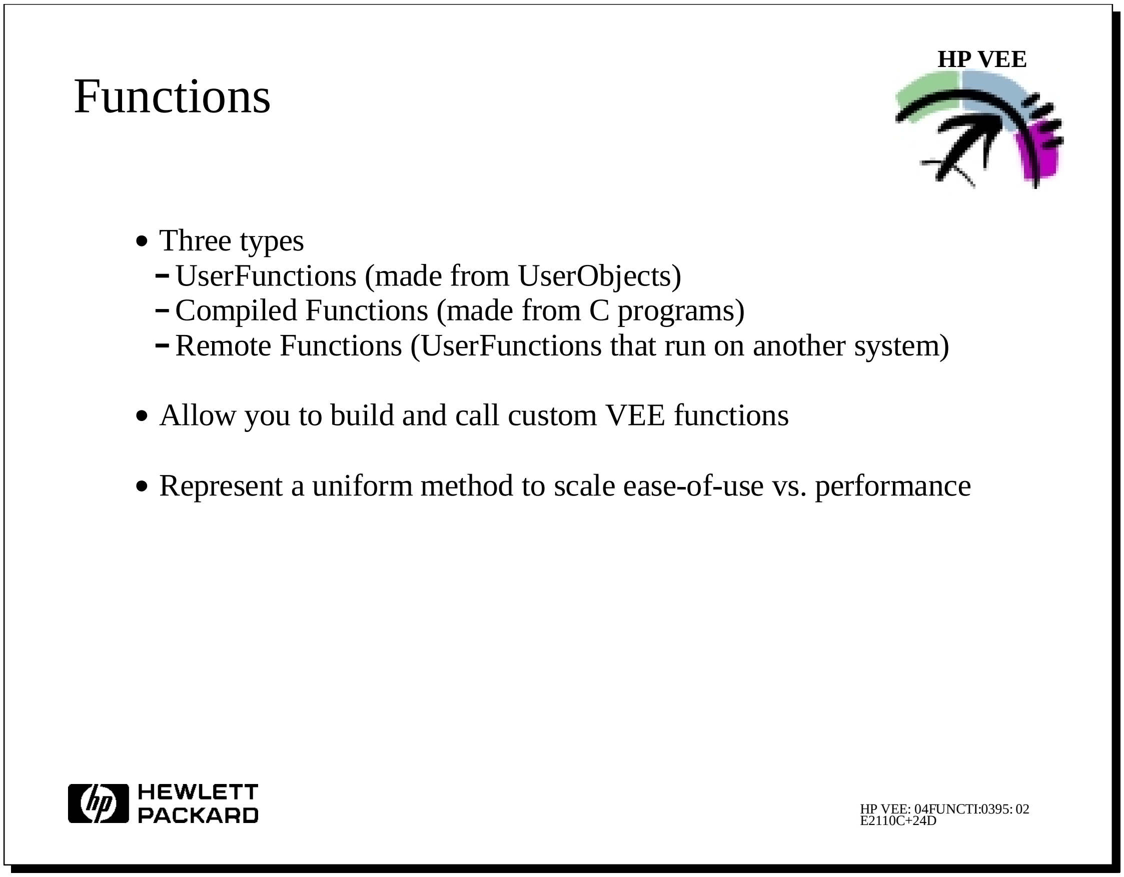
add(a, b)

HEWLETT PACKARD

HP VEE: 04FUNCTI:0395: 01
E2110C+24D

The UserFunction is the next progression above the UserObject. It is made from a UserObject but has several advantages over the UserObject.

The advantages are the UserFunction is referenced in the program from one location and multiple references to the UserFunction uses incrementally smaller amounts of memory.



Functions

HP VEE

- 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

HEWLETT PACKARD

HP VEE: 04FUNCTI:0395: 02
E2110C+24D

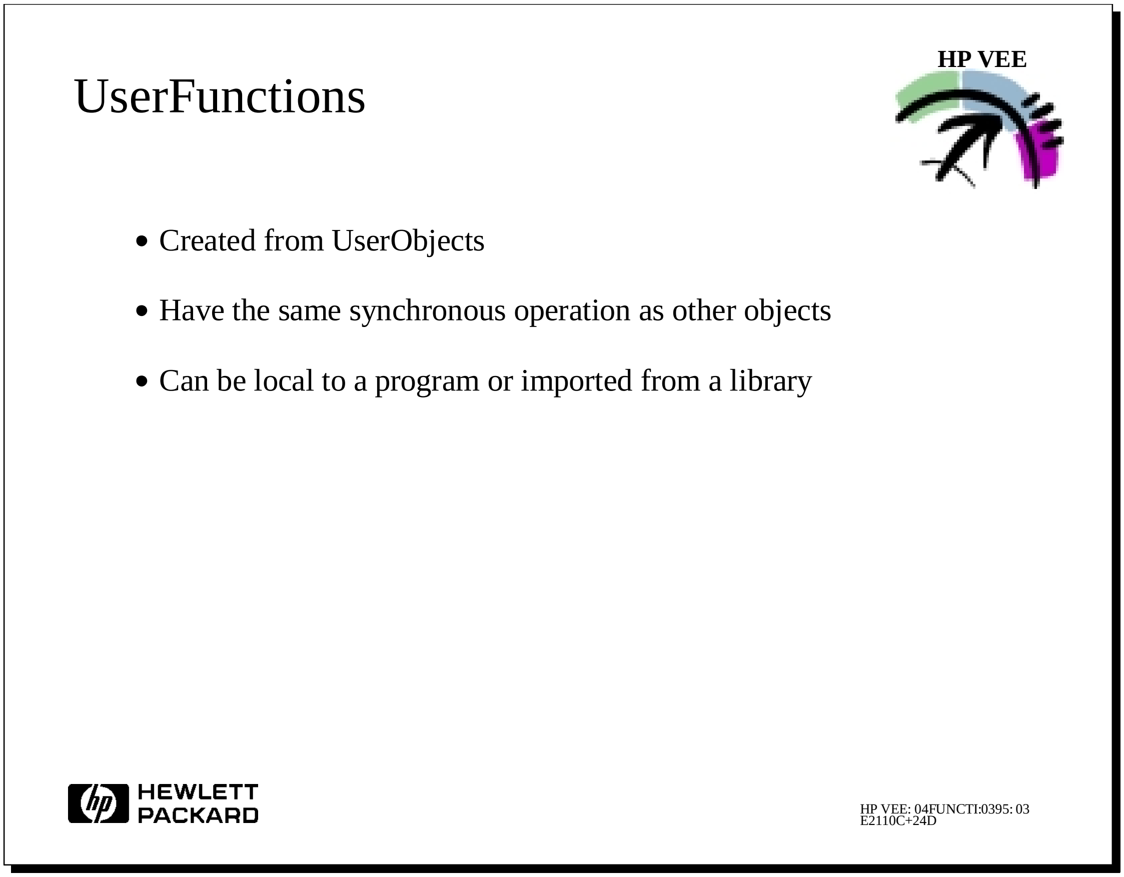
Functions allow the user to define a particular capability or set of capabilities that can be used as a standard part of your HP VEE environment.

All calls to a function reference back to a single defining instance of that function. If you change the single instance of the function, this change is reflected in all instances of the object. Calls to the function have separate data references, so even though only one defining, executable part of a function exists, each call object acts on whatever data is supplied to its data terminals.

Functions can be created as one of three types:

UserFunctions, Compiled Functions, or Remote Functions. The type of function you choose depends on whether you want to write the functions in HP VEE for ease of creation, C for speed, or if you want the function to execute on a different system. If you decide to change the type of function you call, HP VEE only requires you to change one object... the mechanism to call the types of functions is the same.

Compiled and Remote Functions will be discussed in a later section.



HP VEE

UserFunctions

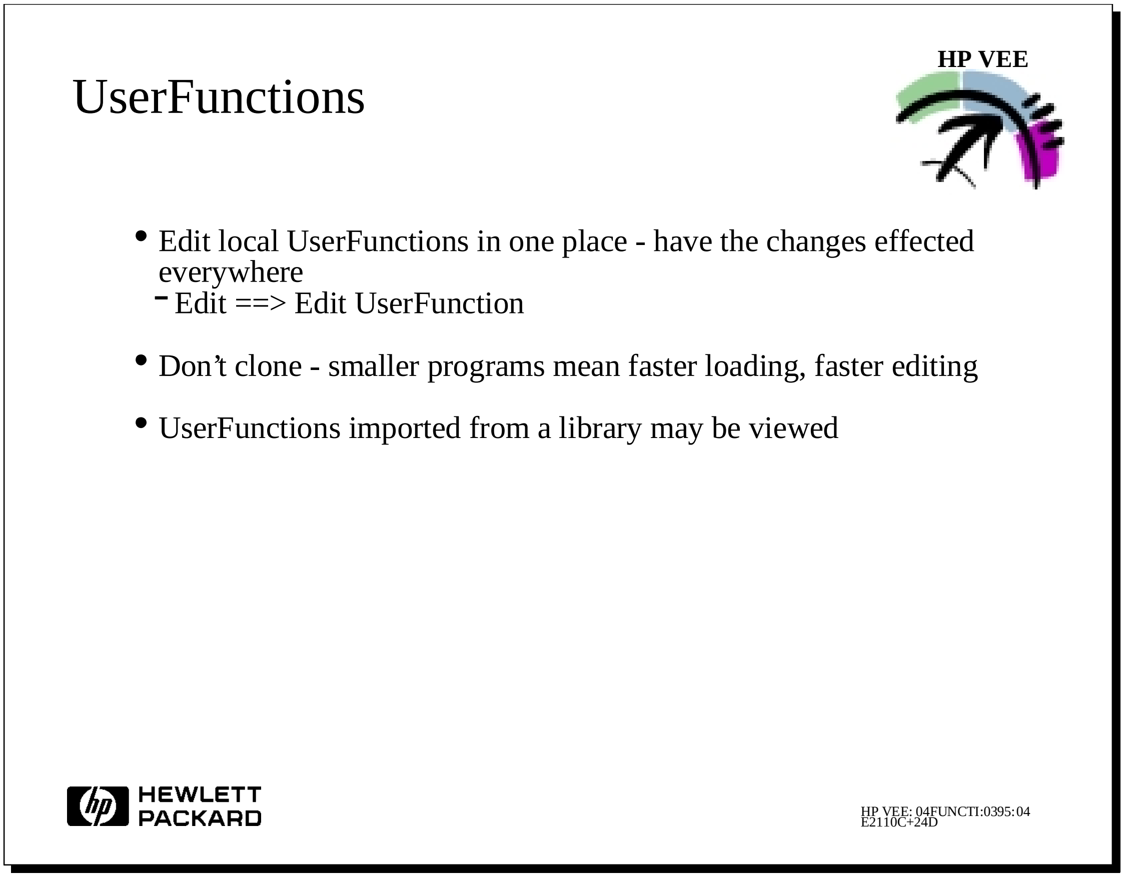
- Created from UserObjects
- Have the same synchronous operation as other objects
- Can be local to a program or imported from a library

HEWLETT PACKARD

HP VEE: 04FUNCTI:0395: 03
E2110C+24D

Functions can be defined to be local within a program or can optionally be stored for later use as a library module. This allows access from other programs and other workstations and serves as the basis for a user-defined library of functions.

In relation to other objects, UserFunctions synchronously operate as a primitive object in that it will operate to completion once it is activated. This is since the UserFunction is referenced through a primitive object. Device ==> Function ==> Call



HP VEE

UserFunctions

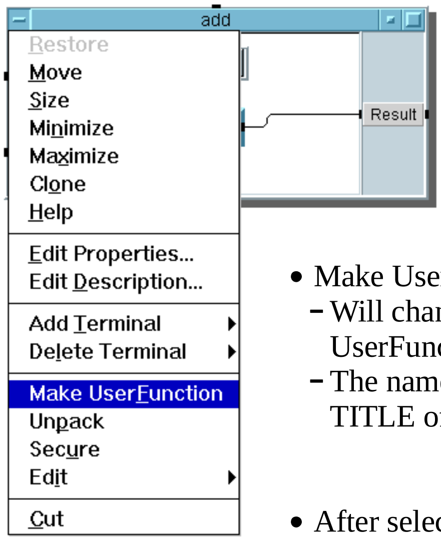
- Edit local UserFunctions in one place - have the changes effected everywhere
  - Edit ==> Edit UserFunction
- Don't clone - smaller programs mean faster loading, faster editing
- UserFunctions imported from a library may be viewed

HP VEE: 04FUNCTI:0395:04
E2110C+24D

Once a UserFunction is created, it is stored in memory. We will see how to call the UserFunction later in this section. Unlike UserObjects, they are not copied but only referenced.

HP VEE

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.

HEWLETT PACKARD

HP VEE: 04FUNCTI:0395: 05
E2110C+24D

To create a UserFunction, first create a UserObject. Each UserObject includes a menu choice to Make UserFunction. (The name of the function is decided by the title of the UserObject). When you make the UserFunction, you will see that the UserObject is replaced by a Call Function object that has the same input and output data terminals, with all the lines still attached. The "look" of the object will change.

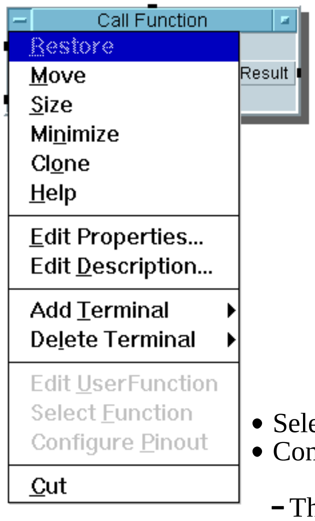
Once a UserFunction is made, it is stored in memory

UserFunctions are saved as part of the program. Once the UserFunction has been saved in a program, it can be imported as a UserFunction into other programs

HP VEE

Calling UserFunctions

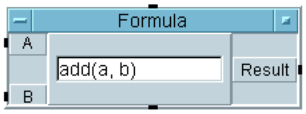
Device ==> Function ==> Call



- Select Function...
- Configure pin-outs

  – The function name can be specified as a control pin

or from ANY expression



add(a, b)

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

HP VEE: 04FUNCTI:0395: 06
E2110C+24D

Device ==> Function ==> Call

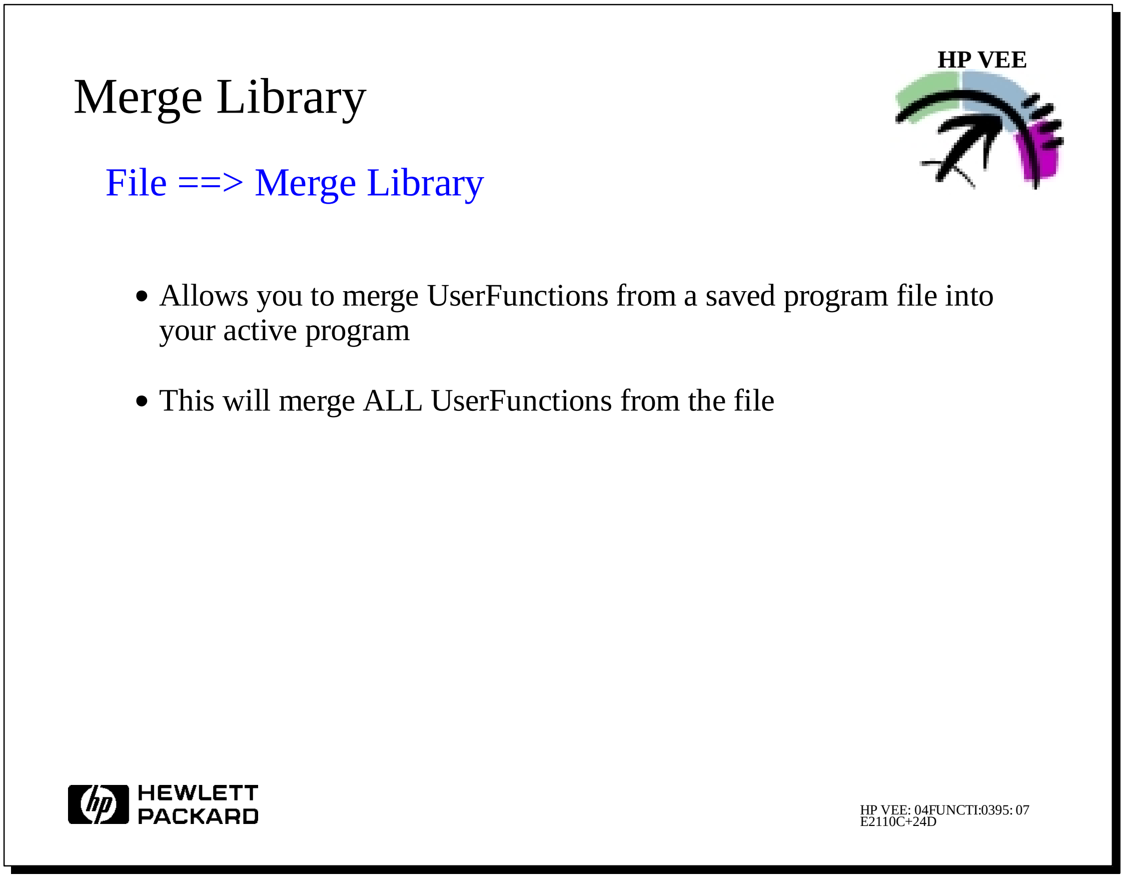
You can call this UserFunction from a Call object or from a Formula box or expression. When you access the Call Function object menu, you will see a choice labeled "Select Function". If you choose this, you will get a dialog box that shows which functions are available to be called. Remember that the Call object adheres to all the same rules as any other object in HP VEE. To function, all of its data in pins must be supplied with new data.

After you have selected a function, you will see a choice on the object menu labeled Configure Pinouts. This is useful when you have changed something in the defining object that requires a different number of pins than you had originally. Configure Pinouts will put the right number of pins on your Call Function object.

To call a UserFunction from a Formula or expression, use the name of the function, followed by a parameter list. Each parameter listed after the function will relate to an input to the function. The parameter list Left-to-Right should equate to the function's Input Terminals, Top-to-Bottom. As in the example, add(a,b)

When called from expressions, the UserFunction will propagate data to the TOP output terminal only.

Note!!: If you want to use a formula or expression to call a UserFunction that does not contain any input terminals, you MUST include empty parens () after the function name. Otherwise, HP VEE will think you are referencing a global variable or input terminal.



Merge Library

HP VEE

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

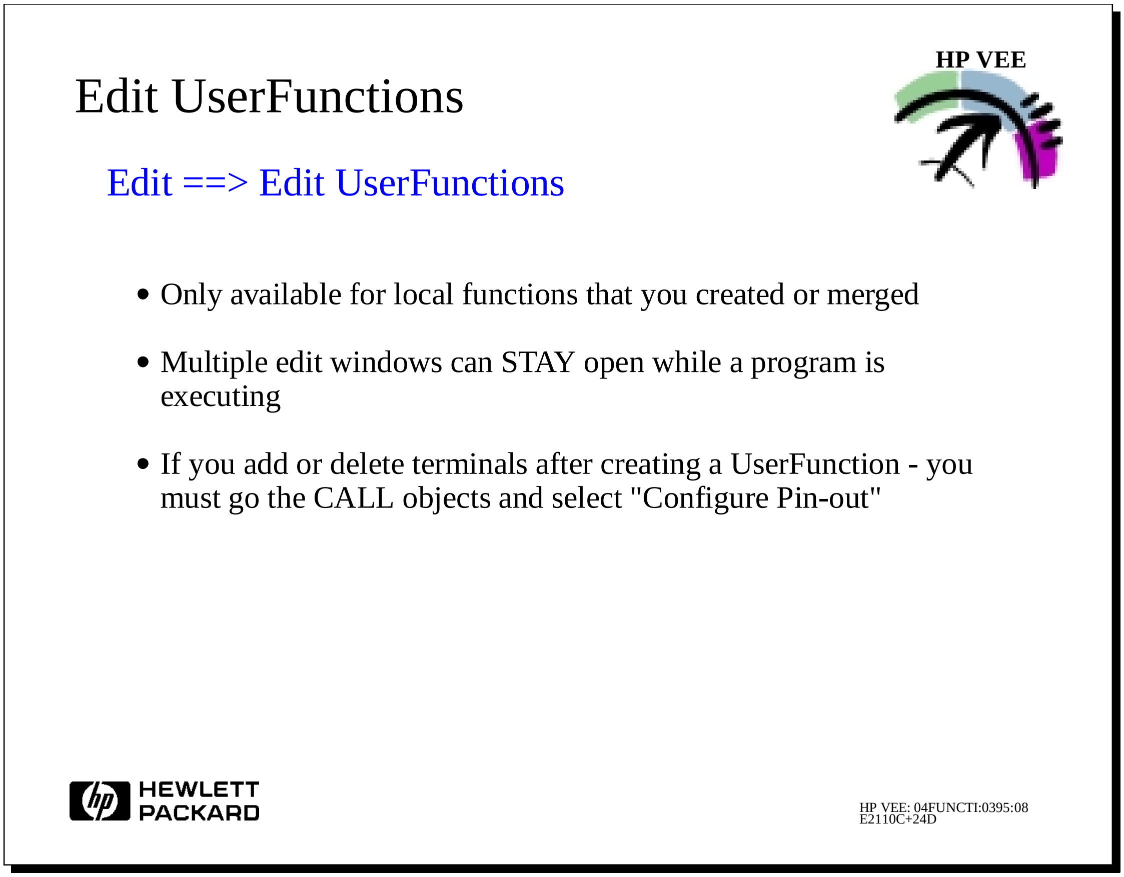
HEWLETT PACKARD

HP VEE: 04FUNCTI:0395: 07
E2110C+24D

When you merge a library, all UserFunctions from the file specified are copied into your program, and are available as if you had created them locally.

The term ’Library’ just means a HPVEE program file that contains the functions you want to merge.

Merged UserFunctions can be edited/changed, but the changes become part of the encapsulating VEE program. In other words, the UserFunction that was referenced is now locally defined instead of referencing the Merge Library UserFunction. The Library is not affected; changes are not made to it.



HP VEE

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"

HEWLETT PACKARD

HP VEE: 04FUNCTI:0395:08
E2110C+24D

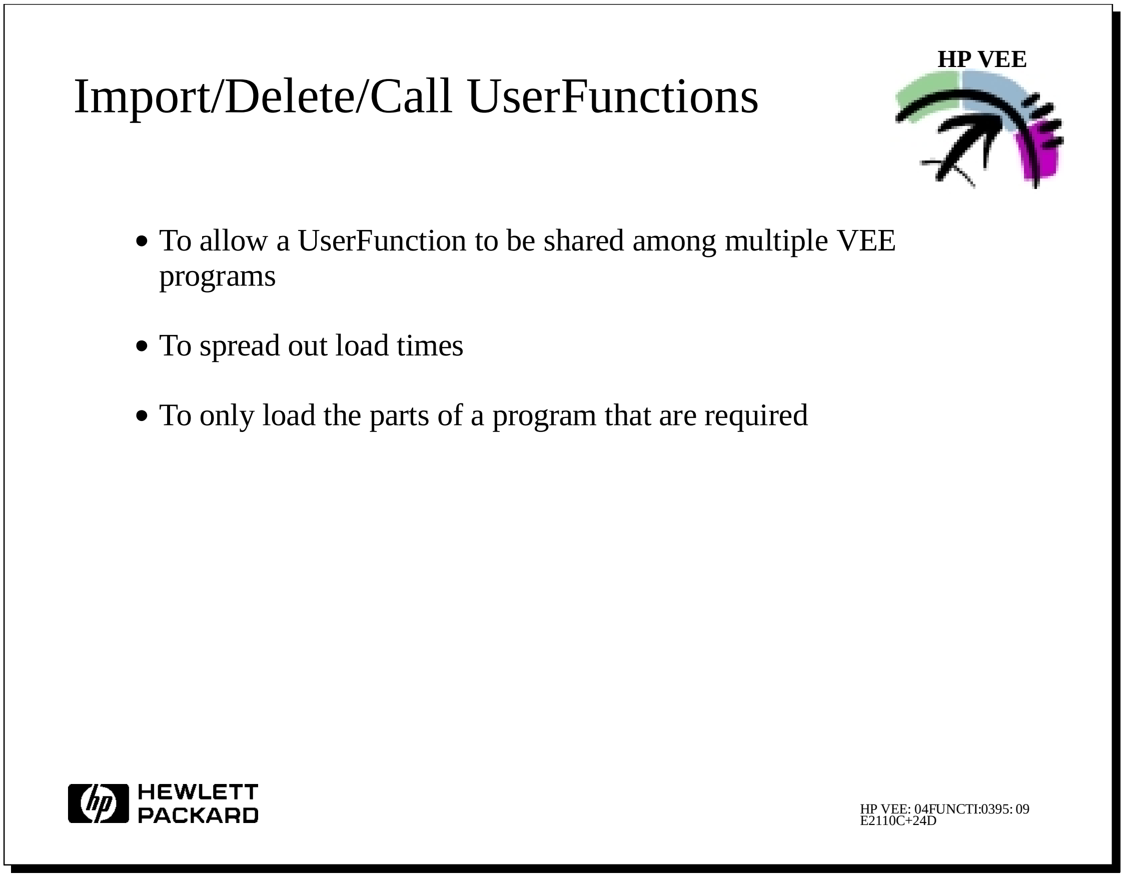
Edit ==> Edit UserFunction

All local UserFunctions can be edited by selecting Edit UserFunction and the name of the Function. When you are finished editing, press DONE. The changes will be reflected for all calls in your program to that specific UserFunction.

This Edit window can stay open while your program is executing. This allows you to use HP VEE's debugging features (e.g. break points, animate, etc.). Also, you can open multiple edit windows for multiple User Functions at the same time.

When the Edit window is open, it has an object menu that is specific to that function. From this menu, users can delete the UserFunction from the program, make it back into a UserObject, change to Detail or Panel View, or several other choices.

To change the Title or Show Panel on Exec setting of the UserFunction, you must select the Change Properties choice from the Edit window's object menu.



HP VEE

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

HEWLETT PACKARD

HP VEE: 04FUNCTI:0395: 09
E2110C+24D

Device ==> Function ==> Import

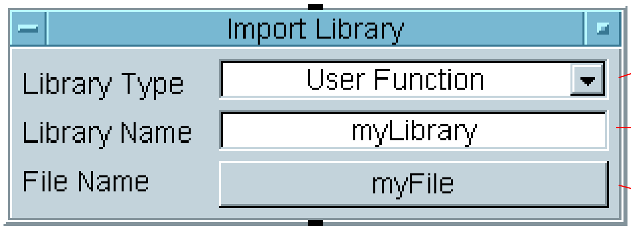
Instead of having all of your UserFunctions loaded and stored with every HP VEE program, you can selectively Import only the functions required. This is great if you want to have multiple HP VEE programs all access the same function, instead of having to keep a copy of the function in every program.

Also, you can now selectively load the parts of a particular program that you need. This will keep you from having to always have all functions loaded at once. An example would be if you have a PASS/FAIL decision. Then only the UserFunctions needed in either the PASS or the FAIL branch are imported.

HP VEE

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.

HEWLETT PACKARD

HP VEE: 04FUNCTI:0395:10
E2110C+24D

To save your UserFunction for use in another program requires no special operations. Simply saving your program will allow other programs to access your functions. One thing to remember is that if you import a function into another program, all functions defined in the source program automatically get included. For this reason, you may want to save your functions in some logical grouping of several at a time in a few files. In this way, you can build a set of libraries that have a single common source (that you could manage with a source code control system) grouped logically by some user-defined criteria.

To access a function from another file, use the Device -> Function -> Import object. This object specifies the type of function you want, such as a UserFunction. It also has the name of the library you will reference if you decide to programmatically delete these functions. The third field is a file selector box you use to tell this object which file to look in to get your function. It is literally the file on the system.

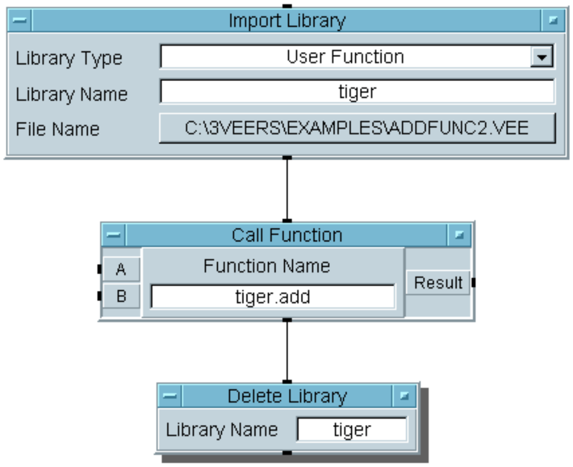
Notice the selections labeled Load Lib and Delete Lib in the Import Library function’s object menu. These are immediate action selections that allow you to have interactive control over the addition and deletion of libraries from your program. This is a very handy feature when you are developing your program. If you first select Load Lib, you can then use the CALL object Select Function to choose the function to call. If you do not use Load Lib on the Import object, you must remember the name of the function and the proper pin-out.

The library name and file name can be specified as control pins on the Import object.

HP VEE

Import/Delete/Call UserFunctions

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



Import Library

Library Type: User Function

Library Name: tiger

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

Call Function

A

B

Function Name: tiger.add

Result

Delete Library

Library Name: tiger

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

HEWLETT PACKARD

HP VEE: 04FUNCTI:0395: 11
E2110C+24D

An example of the Import Library would be to configure the object with Library Type and File Name of the UserFunctions that are needed. Use the edit menu "Load Lib" to immediately load the UserFunctions into memory. Now, the Call Function's Select Function is enabled. This will show you the names of the UserFunctions loaded. If Select Function and/or Edit ==> UserFunction is grayed-out, no UserFunctions were loaded.

Lab 4a

Using UserFunctions

Objective: Learn more about **UserObjects** and then how to make them into **UserFunctions**. In addition, we will learn about calling **UserFunctions**.

Step 1
Create a **UserObject** named **NoiseGen** that accepts an **Amplitude** value (0-1) from a slider and returns a noisy waveform.

Do NOT use **Virtual Source**, **For Count, For Range, Collector**
Do use **Data=>Allocate Array=>Real** and **Advmath=>Probability=>Randomize**

Step 1a
Turn the **UserObject** NoiseGen into a **UserFunction** named NoiseGen.
You may want to build a simple program to be certain this function works correctly.

Step 2
As part of the same program, create a **UserFunction** named **AddNoise** that calls the first **UserFunction**, **NoiseGen** and **adds** the noisy waveform from **NoiseGen** with data from an input pin on the **UserFunction** named **AddNoise.**

The **AddNoise** function should have 2 inputs, 1 for the **NoiseGen** amplitude and 1 for the sine wave (use the **Device=>Virtual Source=>Function Generator** [100Hz] for the sine wave). It should have 1 output pin for the result.

Using the **UserFunctions** that you have made, build a simple program with a **Slider** to control the amplitude of the **NoiseGen** waveform . Plot the resultant waveform.

Step 3
Adding to the same program, call the **AddNoise** function again, this time from a formula box taking the **Absolute Value** of the result.
Plot this **Absolute Value** waveform on the same display as in Step 2.
Next **Edit** the **AddNoise UserFunction** .
Turn ON **Show Data Flow.**
Run the program and notice how the debugging capabilities work on an open **UserFunction.**

Step 4
Now change your program so that the slider sets a **Global Variable** called **Amplitude.** Have the **NoiseGen** function use that **Global** (so **NoiseGen** will no longer require an input pin).

Save this file as UFLAB.

Lab 4b

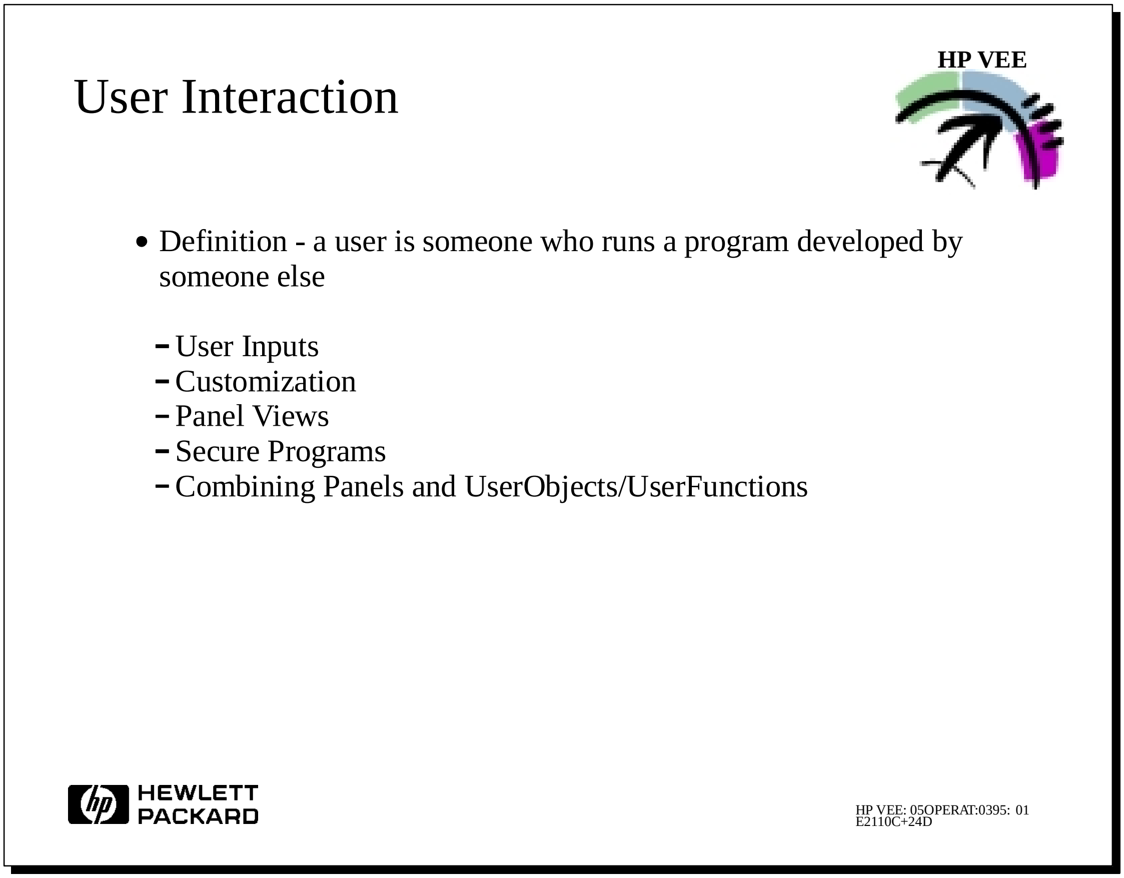
Importing and Deleting UserFunctions

Objective: Learn to import and delete UserFunctions from libraries.

Step 1
Build a simple program to import the **UserFunctions** from the **File** UFLAB. Call the **UserFunction** that adds the noise, and then **Delete** the **UserFunctions** programmatically.

Step 2
Try using the **Select Functions** choice from the **Call** box's object menu.

Operator Interface



HP VEE

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

HEWLETT PACKARD

HP VEE: 05OPERAT:0395: 01
E2110C+24D

Let’s look now at some features that HPVEE provides that help you interact with people using your program.

We’ll discuss:

User Inputs - How the user enters data

Customization - How you can customize the manner in which users interact with your program

Panel Views - A custom interface for your users

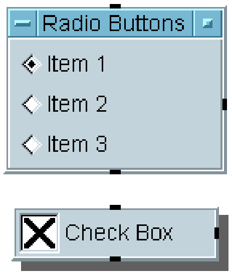
Secure Programs - You can secure your program so that others cannot alter it

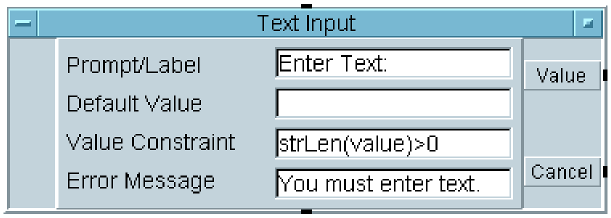
Combine Panels and UserObjects - You can attach custom Panel Views to UserObjects and UserFunctions

HP VEE

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 program



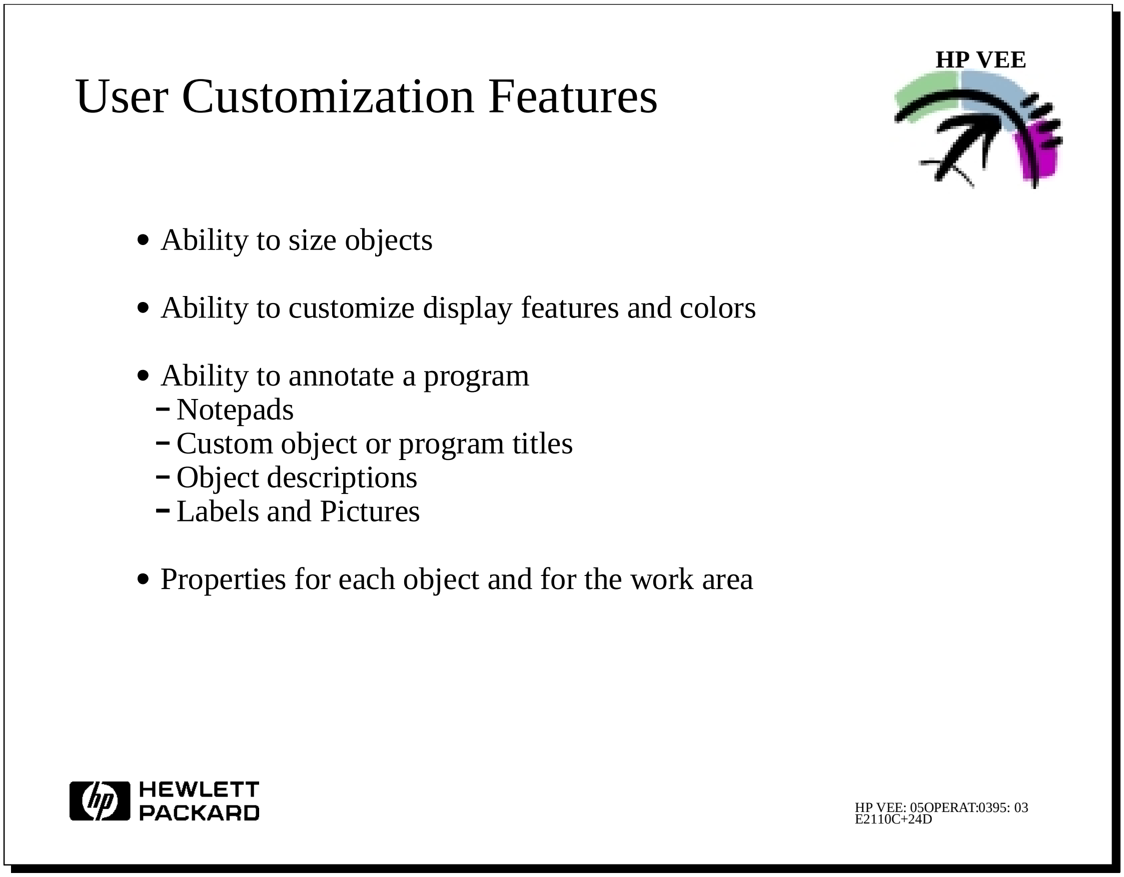

Text Input

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

HEWLETT PACKARD

HP VEE: 05OPERAT:0395: 02
E2110C+24D

The user enters data into a program by using the Set Control, Toggle Control, Slider, Constants, and Dialog Boxes. These objects permit you to prompt the user for inputs. These objects also allow Auto Execute, which automatically propagates the data to other objects. Note that you don’t need to stop and restart the program to enter data. It can be entered while the program is running. Some of the objects will pause the program waiting on the user input.



HP VEE

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

HEWLETT PACKARD

HP VEE: 05OPERAT:0395: 03
E2110C+24D

You can customize your program in many ways such as:

Changing the size of objects to denote importance or to make them easier to use

Changing display features and colors with object properties and work area properties

Annotating your program with user instructions or documentation for maintenance purposes.

You can make annotations by using:

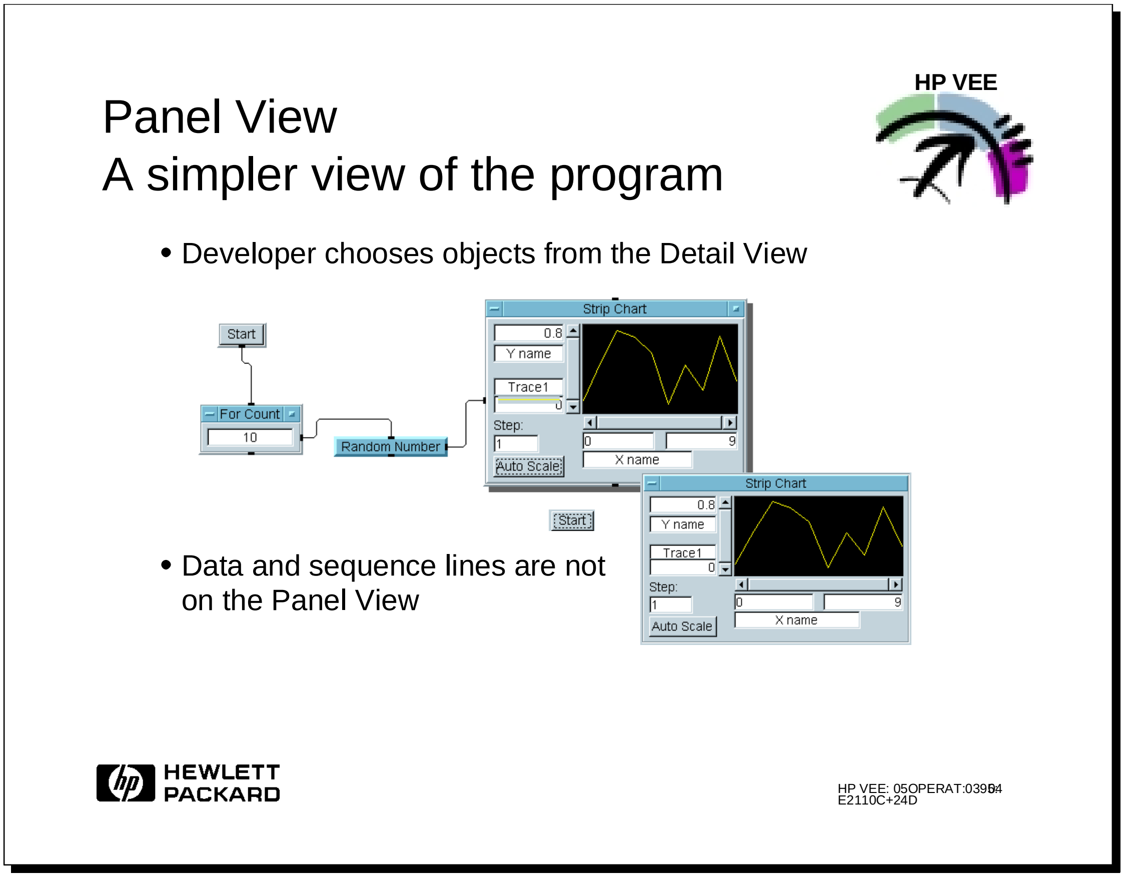
Notepad objects to write notes to yourself in the work area

Editing object titles for further clarity of function or purpose

Writing in the Show Description area of each object

Changing or adding Bitmaps to the Icon View of the object

Labels and Pictures document the program



HP VEE

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

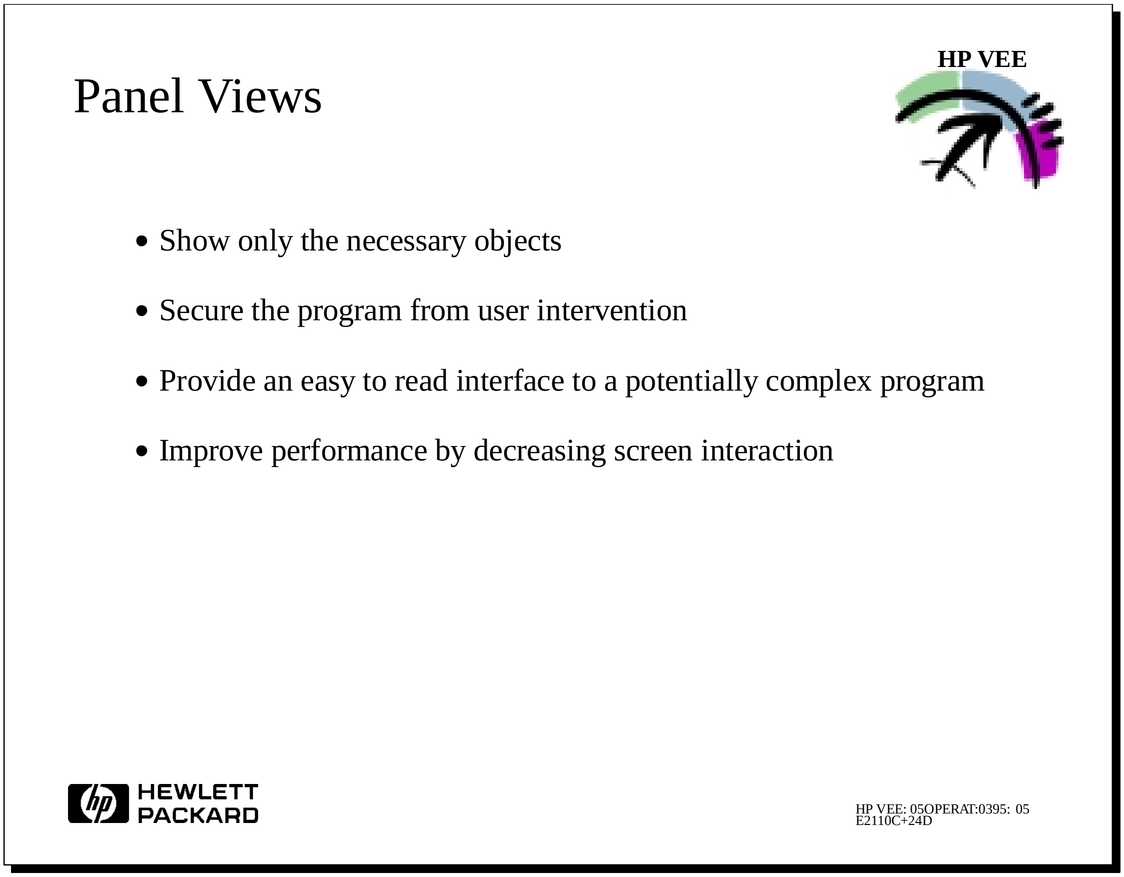
HEWLETT PACKARD

HP VEE: 05OPERAT:03984
E2110C+24D

The Panel View of a program provides another means of customizing your program. You choose which objects appear on the panel. The Panel View does not show the interconnecting lines between objects. By using a Panel View, you simplify and clarify the program seen by the user.

The Detail View is where the development of a program is done. As seen in the other labs, a Panel View is optional.

Detail View <==> Development
Panel View <==> Operator/User Interface



HP VEE

Panel Views

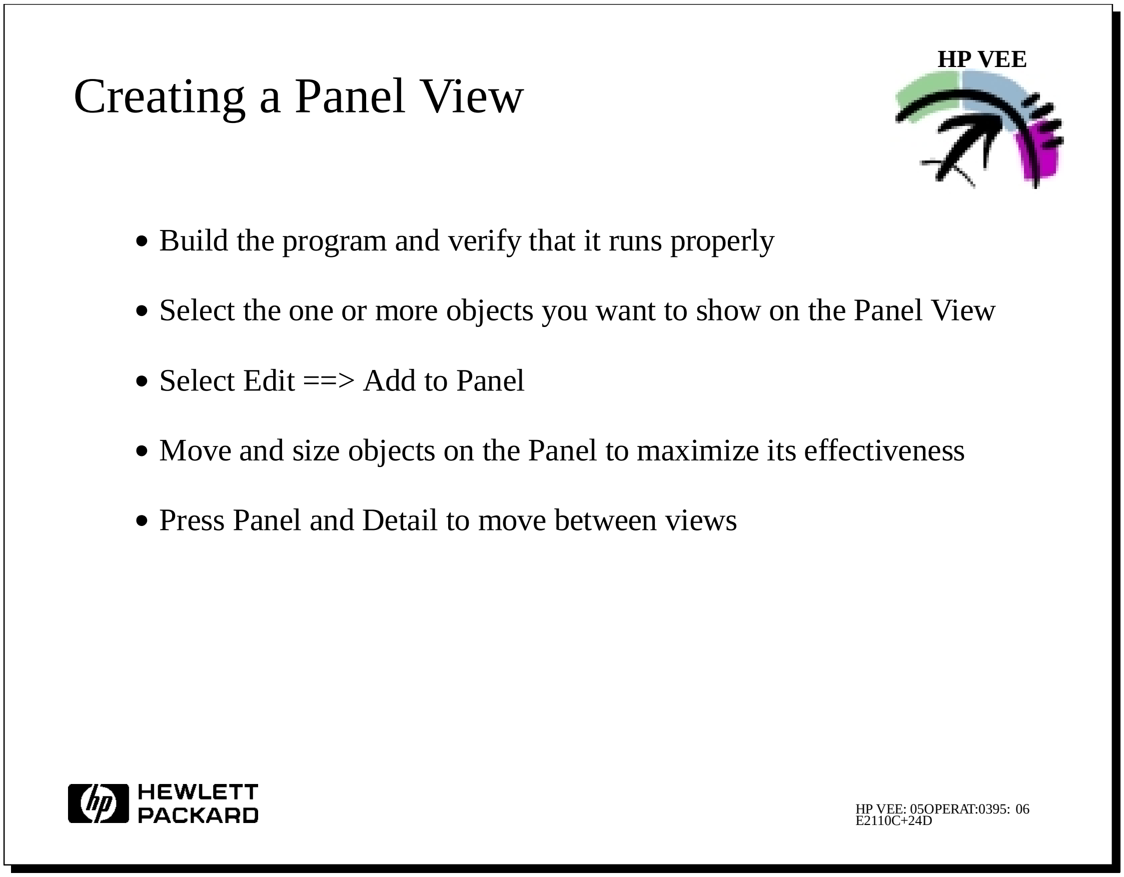
- Show only the necessary objects
- Secure the program from user intervention
- Provide an easy to read interface to a potentially complex program
- Improve performance by decreasing screen interaction

HEWLETT PACKARD

HP VEE: 05OPERAT:0395: 05
E2110C+24D

With Panel Views you display only the objects necessary to interact with your program. You can also secure your program with only the Panel, thus preventing the user from altering the program. Again, the Panel View provides an easy-to-read interface for complex programs.

One of the major performance enhancers is to minimize the computer screen interaction. Updating display objects once by use of a Collector object will usually increase performance. In addition, minimize the use of textual display objects.



HP VEE

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

HEWLETT PACKARD

HP VEE: 05OPERAT:0395: 06
E2110C+24D

So how do you create a Panel View?

1. Create the program and verify that it works as desired.

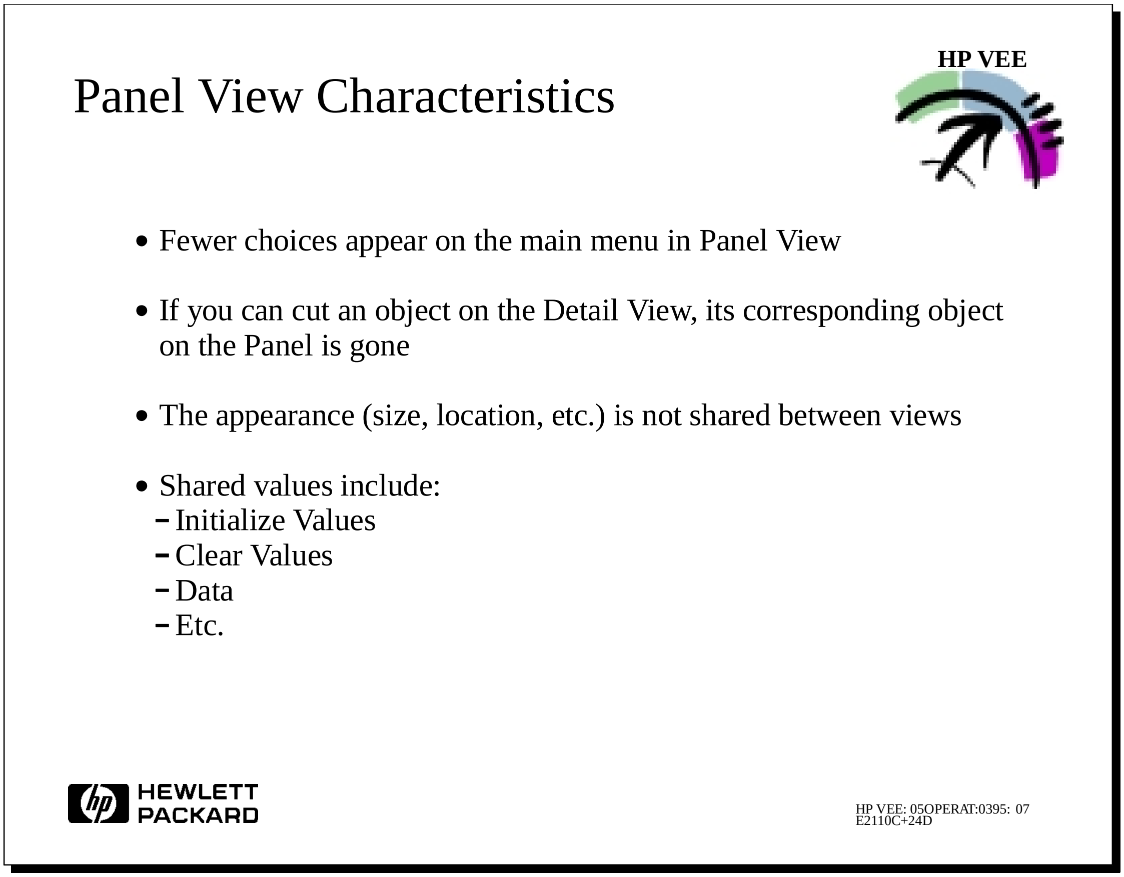
2. Select the objects that you want to appear on the Panel View. These should be objects that the user interacts with or that display information for the user. A shortcut to select objects is Control LB on an object.

3. Select Add to Panel from the Edit menu. Edit ==> Add to Panel

4. Move and size the objects on the Panel View to maximize its effectiveness for the user.

5. Use Panel and Detail buttons on the title bar to move between views.

Remember that changes to the Detail View DO NOT affect the Panel View, except when a Detail View object is deleted. This will automatically delete the Panel View object.



Panel View Characteristics

HP VEE

- 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.

HEWLETT PACKARD

HP VEE: 05OPERAT:0395: 07
E2110C+24D

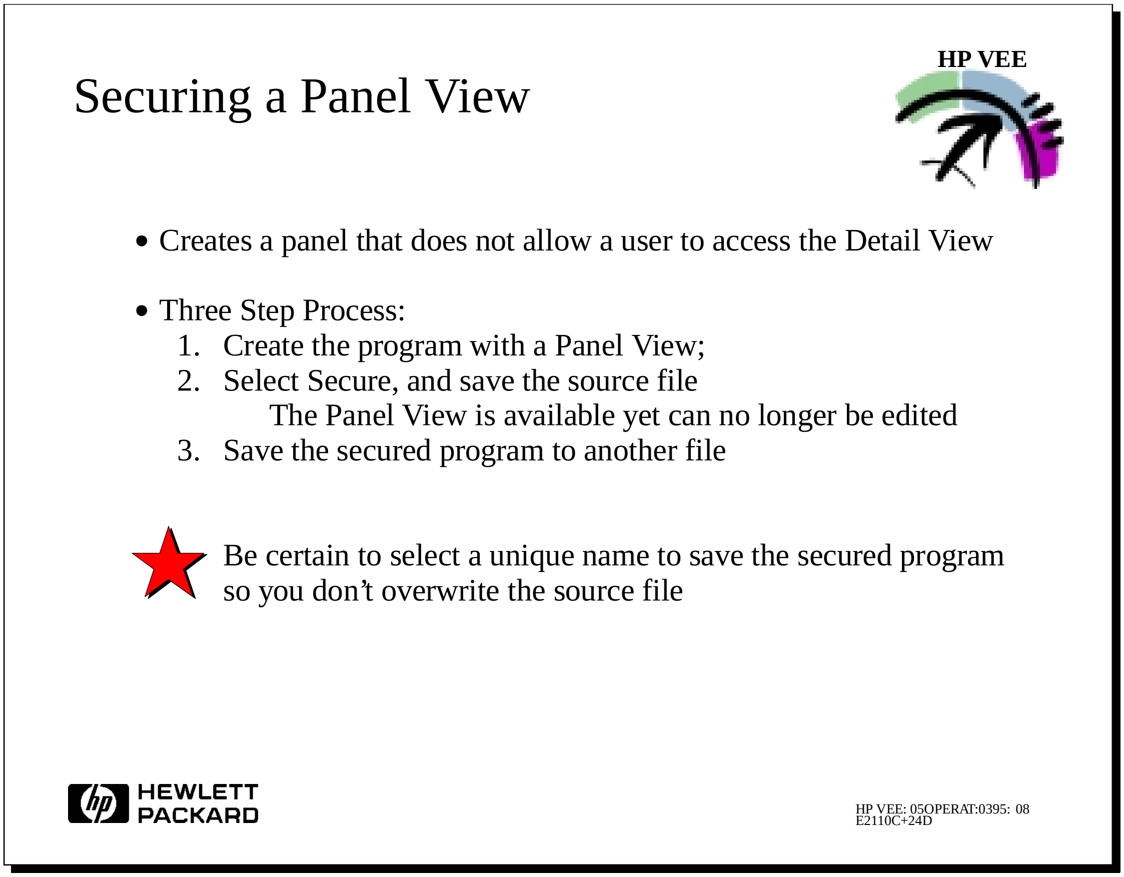
Note that fewer choices appear on the main menu bar in the Panel View. This prohibits the user from altering the operation of the program.

When you delete an object from the detail view, its corresponding object on the Panel View is also deleted.

HP VEE does not share appearance characteristics between the panel and detail views. You can change the size or location in one without affecting the other. However, it does share the following characteristics:

Initial Values
Clear Values
Data
Etc.

Data means the actual content of the program. The program can have different Alphanumeric number formats between the Panel and Detail Views.



HP VEE

Securing a Panel View

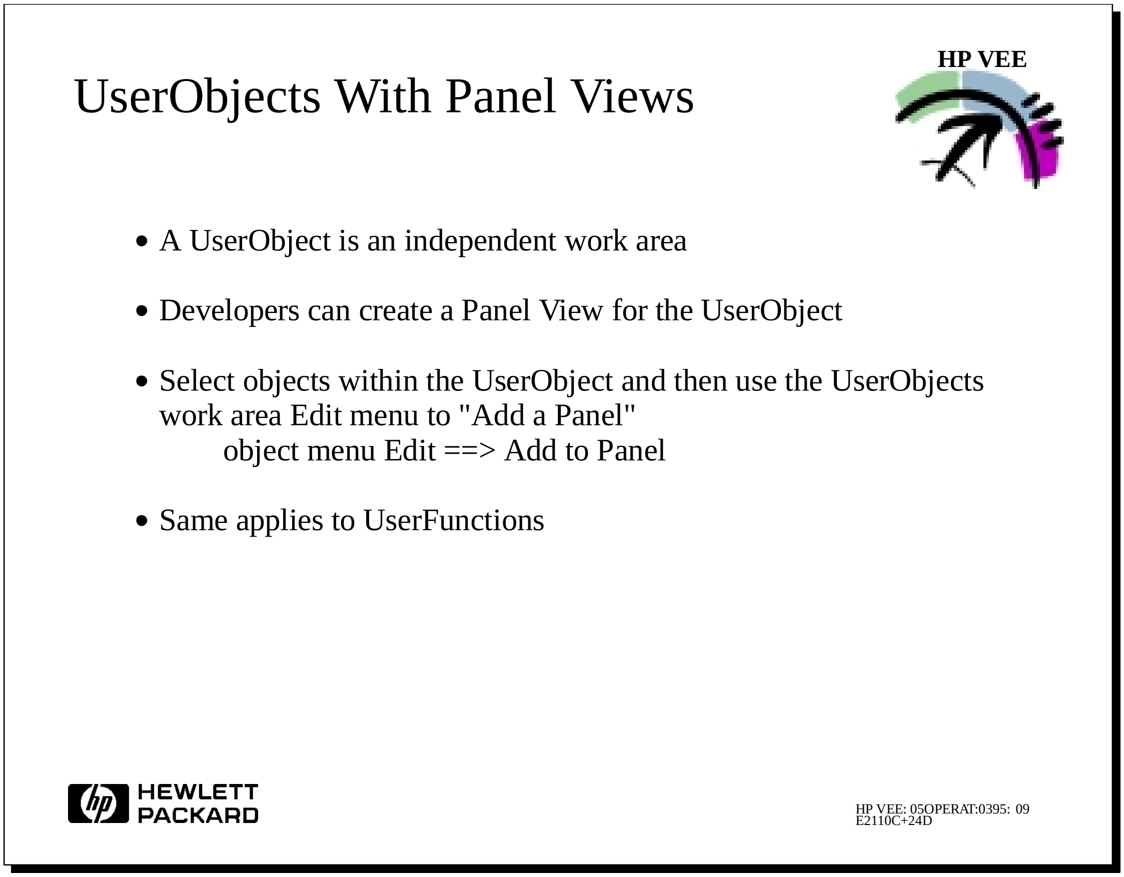
- Creates a panel that does not allow a user to access the Detail View
- Three Step Process:
  1. Create the program with a Panel View;
  2. Select Secure, and save the source file
     The Panel View is available yet can no longer be edited
  3. Save the secured program to another file

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

HEWLETT PACKARD

HP VEE: 05OPERAT:0395: 08
E2110C+24D

Securing a panel prevents the user from seeing the detail view.



HP VEE

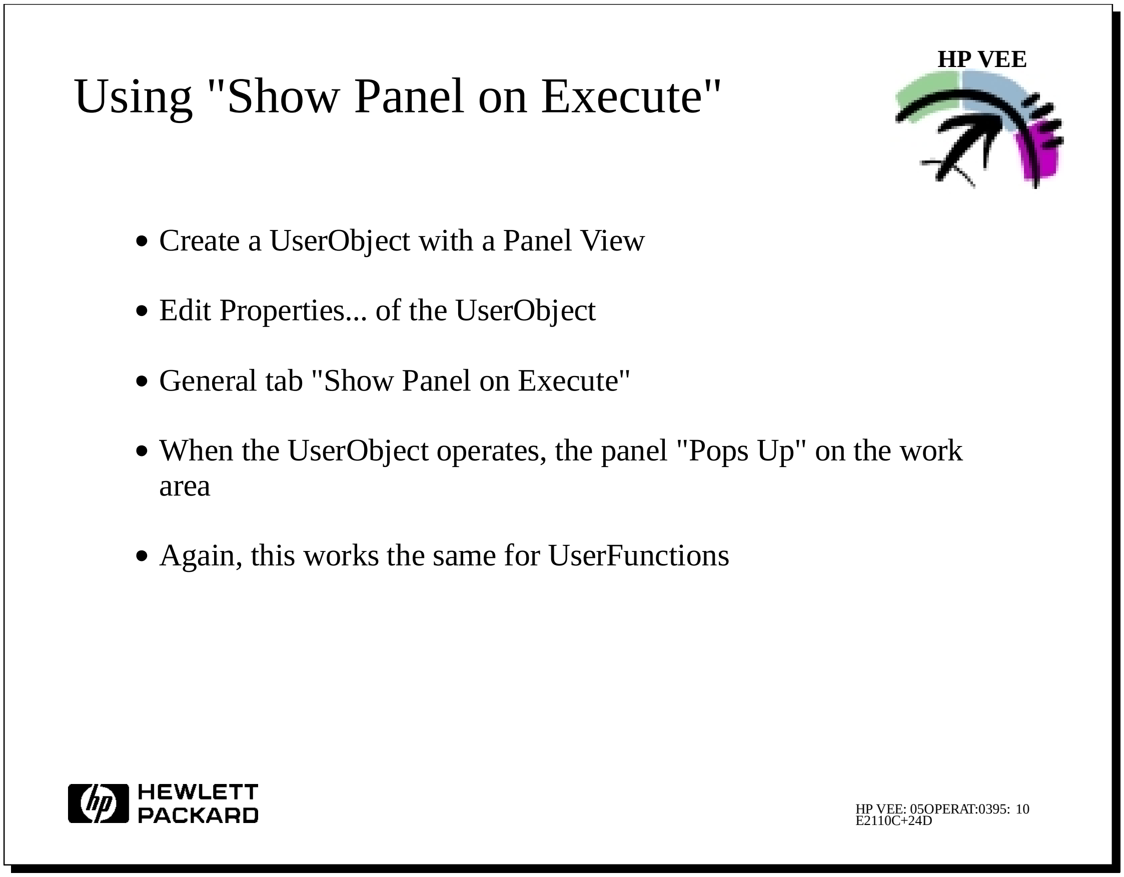
UserObjects With Panel Views

- 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

HEWLETT PACKARD

HP VEE: 05OPERAT:0395: 09
E2110C+24D

Remember that a UserObject is an independent work area within the main work area or another object. Each UserObject allows you to create a Panel View associated with that individual object. As before, select the objects that you want to appear on the Panel View, then select Add to Panel from the UserObject’sEdit menu or the pop-up Edit menu inside the UserObject work area. Add to Panel is context sensitive.



HP VEE

Using "Show Panel on Execute"

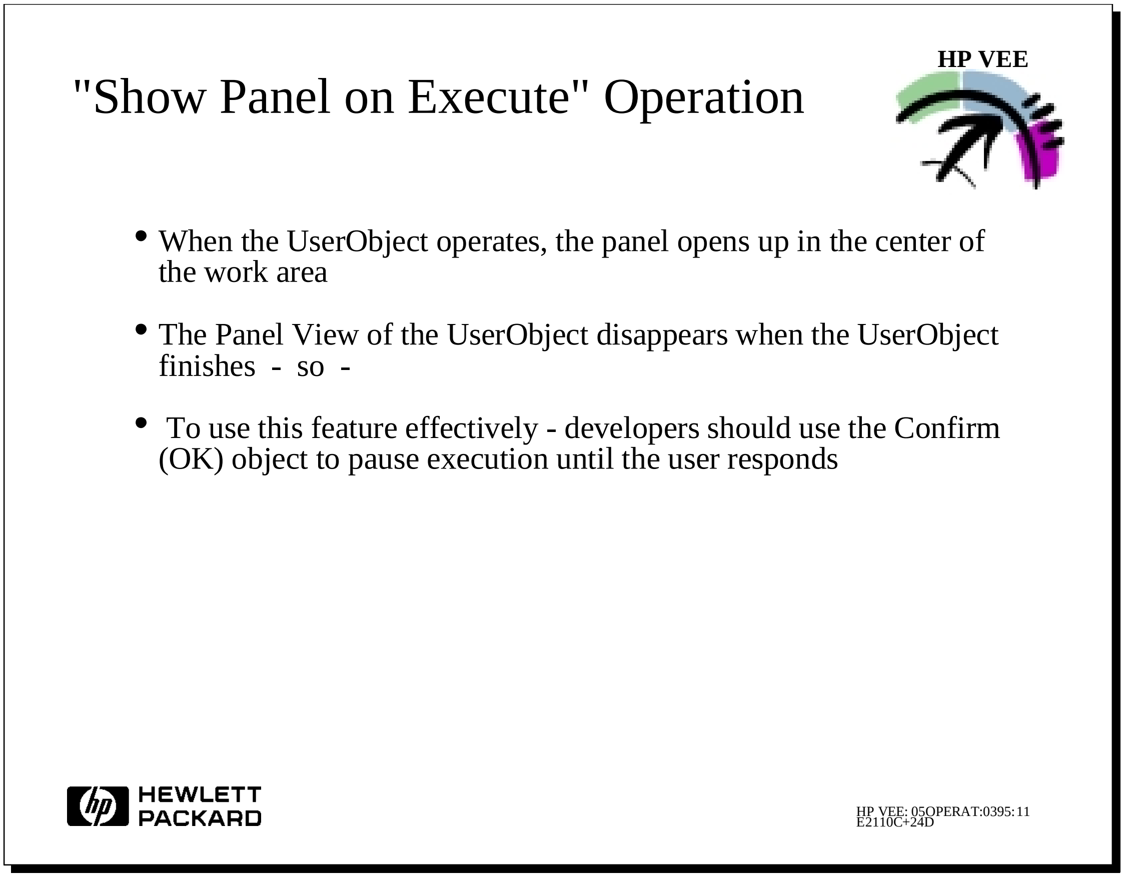
- Create a UserObject with a Panel View
- Edit Properties... of the UserObject
- General tab "Show Panel on Execute"
- When the UserObject operates, the panel "Pops Up" on the work area
- Again, this works the same for UserFunctions

HEWLETT PACKARD

HP VEE: 05OPERAT:0395: 10
E2110C+24D

The UserObjects also implement a unique feature with respect to the Panel View. You can display the panel ONLY while the UserObject is operating. To do this, select Show Panel on Execute from the UserObject'sEdit Properties... Now the Panel View associated with that UserObject appears in the work area when the UserObject is operating. The UserObject will remain on the work area for as long as primitive objects are active. Check on Exit UserObject.

If a UserObject is on an iterating subthread, it will repeatedly pop up and go away.



"Show Panel on Execute" Operation

HP VEE

- 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

HEWLETT PACKARD

HP VEE: 05OPERAT:0395:11
E2110C+24D

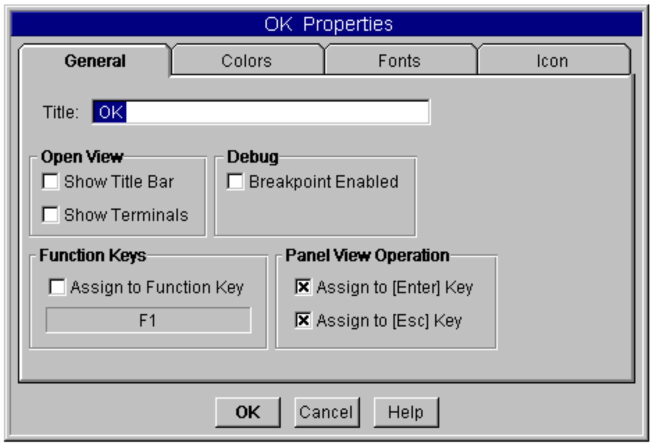
When the UserObject operates, the Panel View appears in the work area. It disappears when the UserObject finishes. Therefore, to use this feature effectively you should add a Confirm (OK) object to the Panel View to pause execution until the user responds.

Note: If the program is propagating out the wrong data, check the ordering of the OK object relative to the data object.

HP VEE

OK Button - Keyboard Actions

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



OK Properties

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

HP VEE: 05OPERAT:0395: 12
E2110C+24D

On the Panel View, the OK button can be assigned to the Enter and ESC keys; this can facilitate an operator using the keyboard instead of the mouse.

The OK button can also be assigned to Function Keys (F1 - F12). This assignment will operate in Detail or Panel Views.

Lab 5

Creating Operator Panels and Pop-Ups

Objective: To learn to develop an Operator Panel and to create Pop-Up Panels

Step 1
Create a **UserObject** to interact with an operator. You will need two data inputs, A and B. After data is entered, if A equals B (A==B), send A to an output display. If A and B are not equal (A!=B), prompt the operator to select either A or B while the information is being displayed in a menu. If the operator is not fast enough, after five (5) seconds, generate a **User Defined Error** message.

Hint: Each panel that pops-up needs to be a separate **UserObject.**

Remember **SHOW PANEL ON EXECUTE** for pop-up action.

Step 2
Change the **UserObjects** into **UserFunctions**.

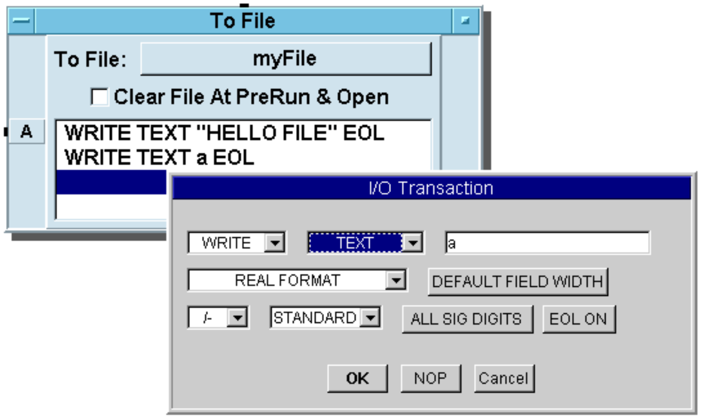
Transactions

Introduction to HP VEE
V3.1 ©1996

Transactions

HP VEE

- 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

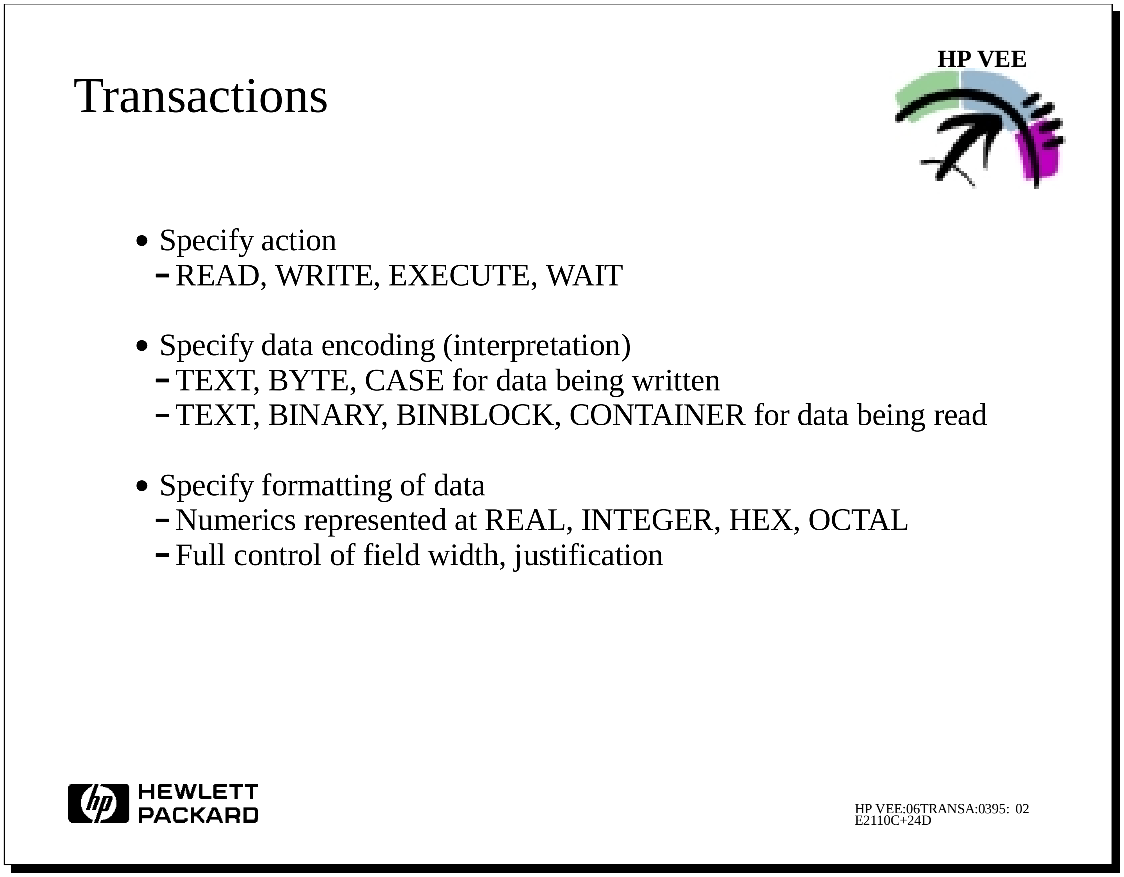
HEWLETT PACKARD

HP VEE:06TRANSA:0395: 01
E2110C+24D

All of the communications paths between HP VEE and other resources go through transaction objects. For example, this slide shows a I/O ==> To ==> File transaction.

Note that each object may have multiple transactions contained within it, i.e. multiple reads and writes.

Also note that each transaction may handle multiple data items. For example, you can read in a single coordinate pair, or an array of coordinate pairs as a single transaction.



HP VEE

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

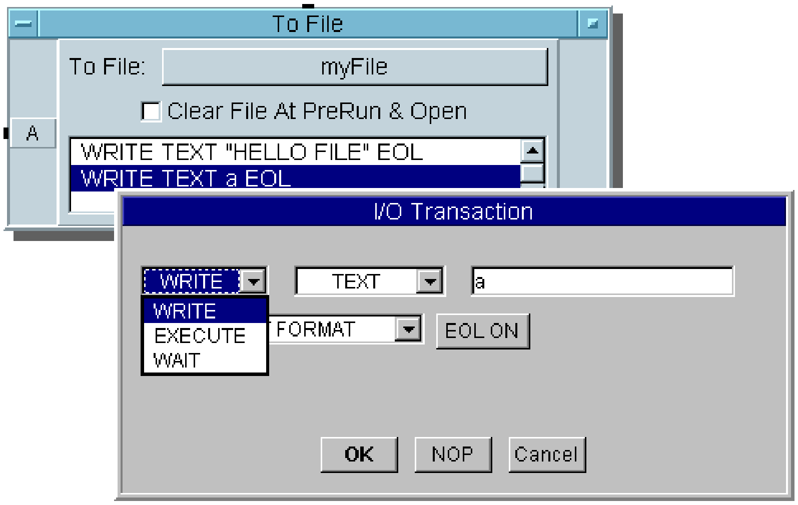
HEWLETT PACKARD

HP VEE:06TRANSA:0395: 02
E2110C+24D

Within each Transaction, items may be specified such as Action, Data Encoding, and Data Formatting. There are a great number of permutations to these items. The following section is not an exhaustive list.

Actions

HP VEE



To File: myFile

☐ Clear File At PreRun & Open

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

HEWLETT PACKARD

HP VEE:06TRANSA:0395: 03
E2110C+24D

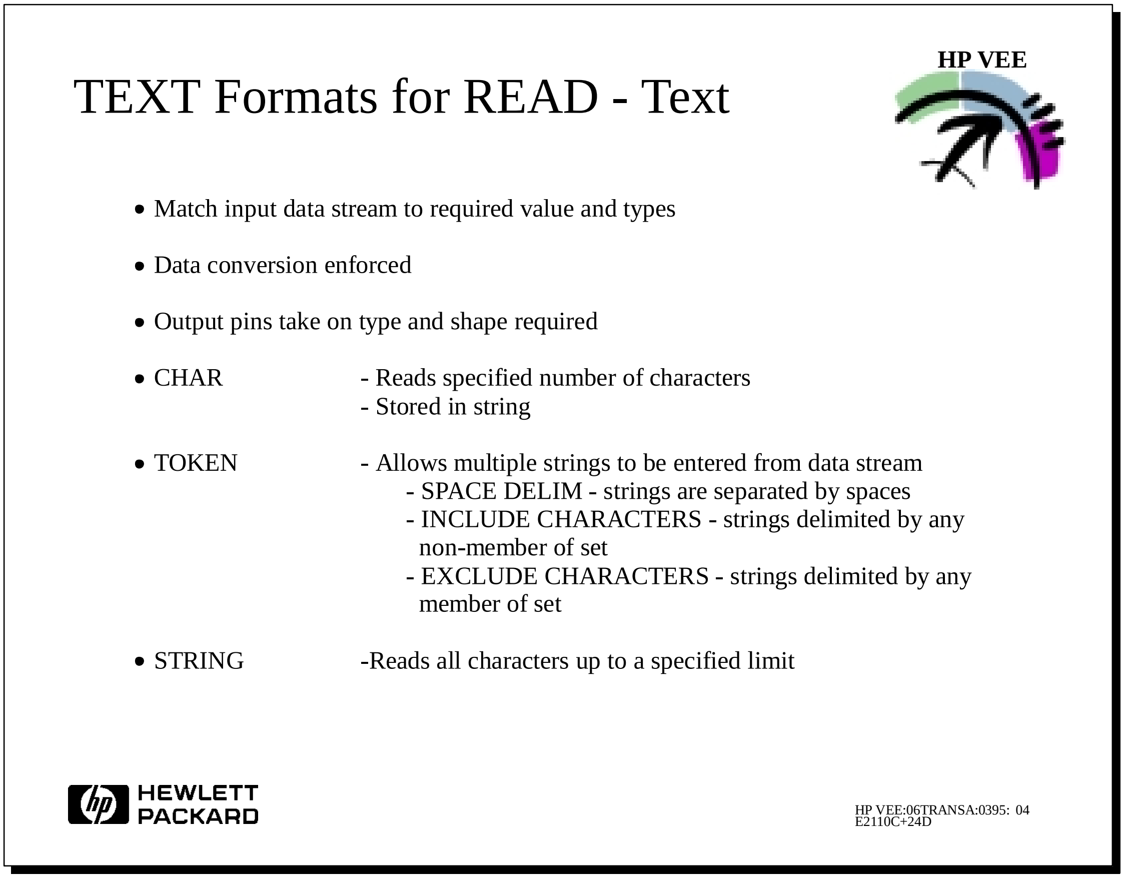
You can specify the following actions in a transaction box:

READ - read data into HP VEE from another resource

WRITE - write data out of HP VEE to another resource

EXECUTE - performs a device-dependent operation on the resource. Such as REWIND a File to the beginning

WAIT - waits the specified number of seconds before performing the next transaction



HP VEE

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

HEWLETT PACKARD

HP VEE:06TRANSA:0395: 04
E2110C+24D

You also specify TEXT formats for READ actions. HP VEE then matches the input data stream to the required values and types. It enforces the data conversion specified. The object's output pin then takes on the type and shape of the data.

CHAR **-** This format reads the specified number of characters and stores it as a string.

TOKEN **-** This format allows you to read several strings from a data stream.

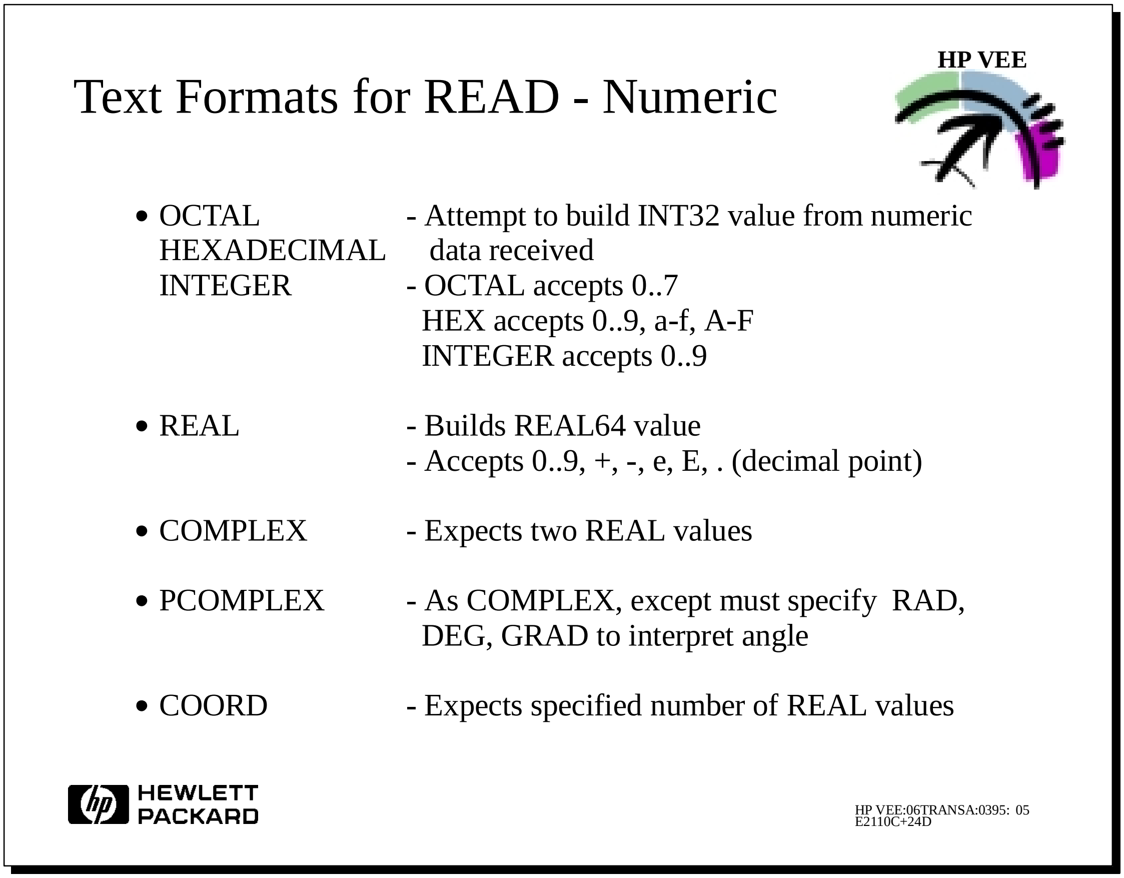
SPACE DELIM specifies that a space separates each string.

INCLUDE CHARACTERS specifies that any character not specified in the set separates each string.

EXCLUDE CHARACTERS specifies that any character in the specified set separates each string.

Useful for parsing data

STRING **-** This format reads all characters up to the number specified.



HP VEE

Text Formats for READ - Numeric

- OCTAL
  HEXADECIMAL
  INTEGER
  - Attempt to build INT32 value from numeric data received
  - 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

HEWLETT PACKARD

HP VEE:06TRANSA:0395: 05
E2110C+24D

The text formats for numeric READ actions are:

OCTAL - build an Int32 from received numeric data

HEXADECIMAL - build an Int32 from received numeric data

INTEGER - build an Int32 from received numeric data

REAL - build a Real (64-bit) from received numeric data

COMPLEX - build two Reals (64-bit) from received numeric data

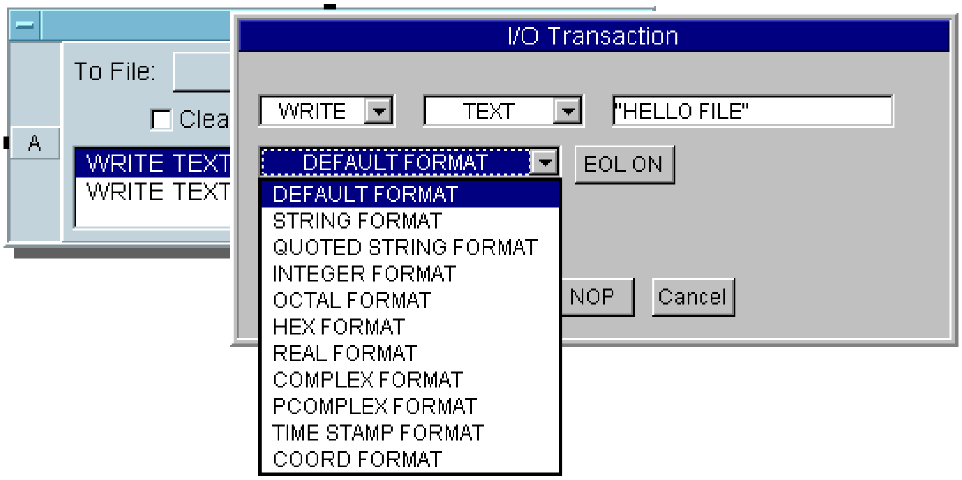
PCOMPLEX - build two Reals (64-bit) from received numeric data. Remember to specify RAD, DEG, or GRAD to interpret the angle correctly.

COORD - build the specified number of Reals (64-bit) from received numeric data.

HP VEE provides you objects to communicate data to the "outside world." This enables you to share information with other programs, or simply archive data generated by your program.

Text Formats for WRITE

HP VEE



HEWLETT PACKARD

HP VEE:06TRANSA:0395: 06
E2110C+24D

These are the TEXT formats available for a WRITE action in a transaction box. You use them to "beautify" your output data. Note that very little type checking or conversion takes place. It simply formats your data.

DEFAULT - all data is in a free-field notation. It includes all characters of string data or all significant digits of numeric data.

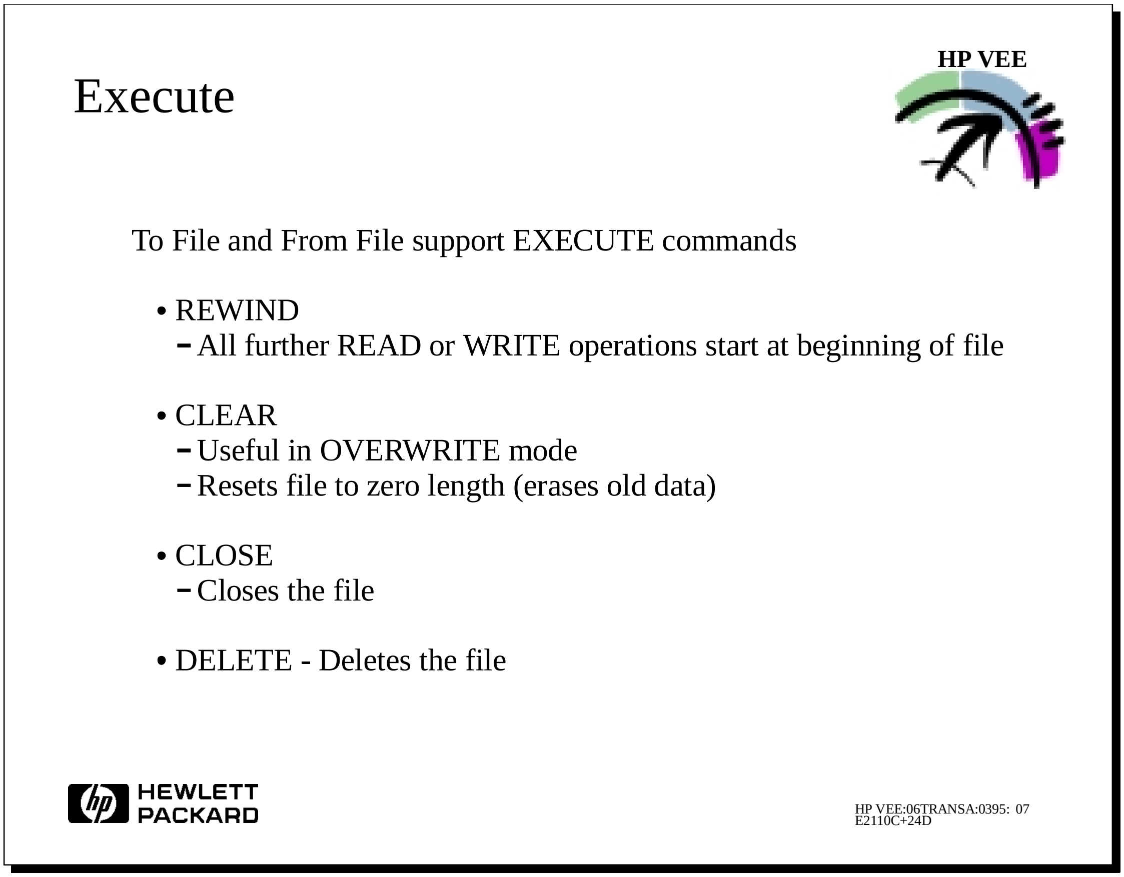
STRING - all data is in a free-field notation as in the DEFAULT format. However, with this format you can control the field width and justification (left to right).

QUOTED STRING - this format is the same as STRING format, except that all data is enclosed in double STRING quotes. It also handles data with embedded quotes.

REAL - all data is in a free-field notation as the above formats. You can control the sign prefix and number of significant digits. You can also designate FIXED, STANDARD, or SCIENTIFIC notation formats for the data.

COMPLEX - this format appears the same as two REALS separated by a comma.

PCOMPLEX - this format appears the same as COMPLEX, except that the angle value is preceded by a @.



HP VEE

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

HEWLETT PACKARD

HP VEE:06TRANSA:0395: 07
E2110C+24D

REWIND **-** this will move the file pointer back to the beginning of the file. If you want to APPEND data to a file, do not REWIND

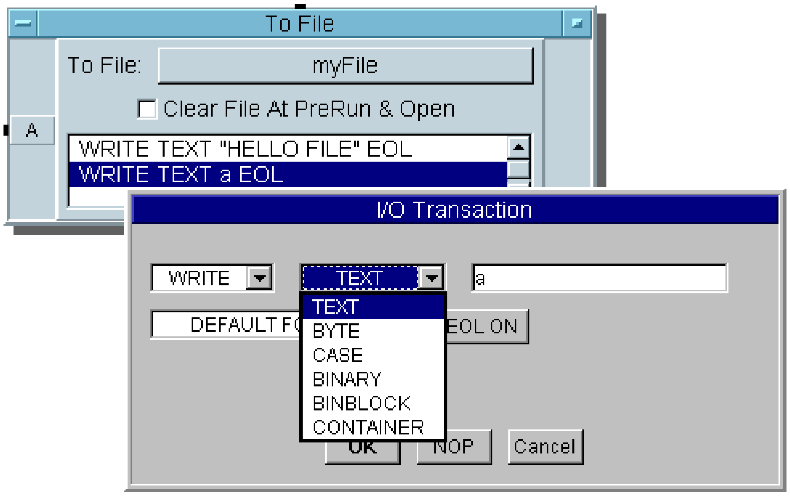
CLEAR **-** this will set the file length to zero, essentially erasing the file contents but not the file existents

CLOSE **-** closes the file from any reading or writing

DELETE **-** removes the file from the system. The file no longer exists

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:06TRANSA:0395: 08
E2110C+24D

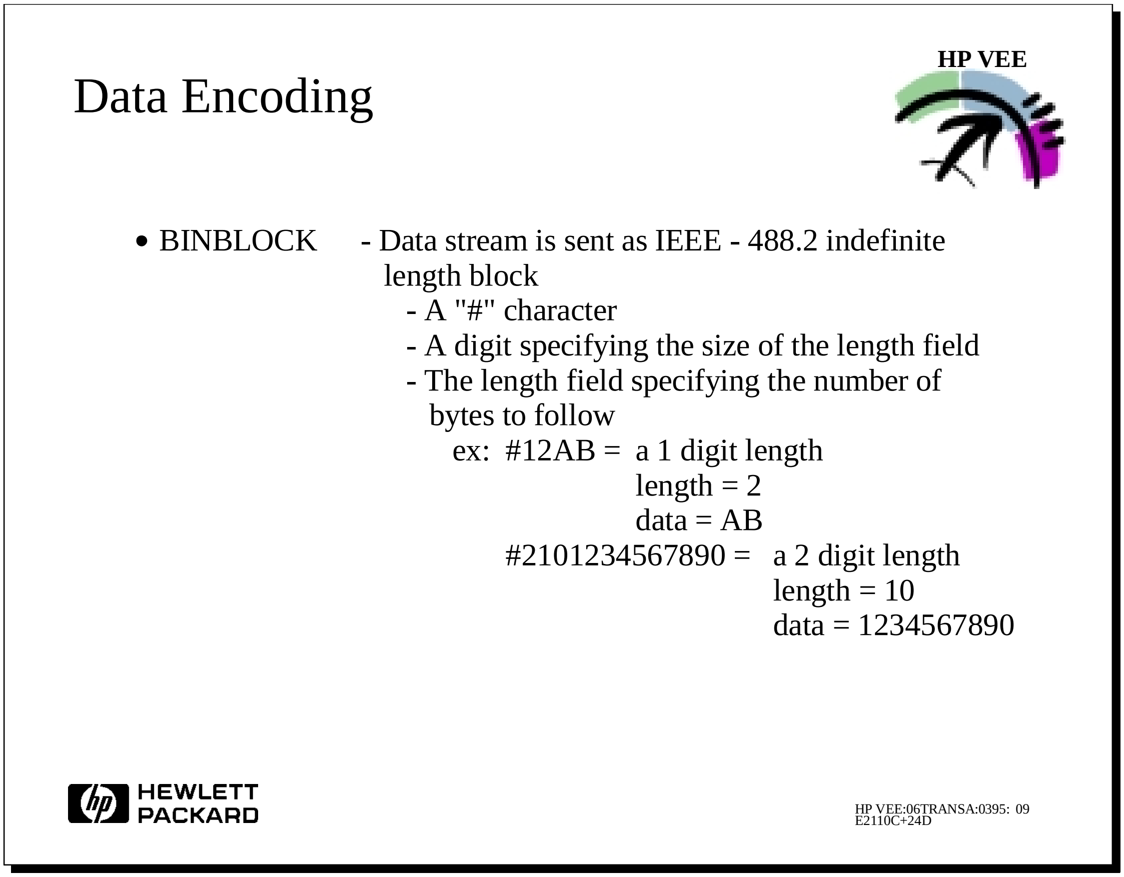
You can specify the following data encodings in a transaction box:

TEXT - a data stream of ASCII characters. Data types are constructed character by character. For example, the data stream 1.23456 EOL becomes the REAL value 1.23456

BINARY - a data stream of bytes which match the HP VEE internal representation. For example, a REAL value will have the IEEE 754 64-bit data format, which uses 8 bytes

BYTE - a data stream consisting of one byte per variable

CASE - a data stream that behaves like an enumerated type. For example, the statement CASE x OF ZERO, ONE, TWO will select the string TWO if x equals 2. Note that this works only for WRITE transactions



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

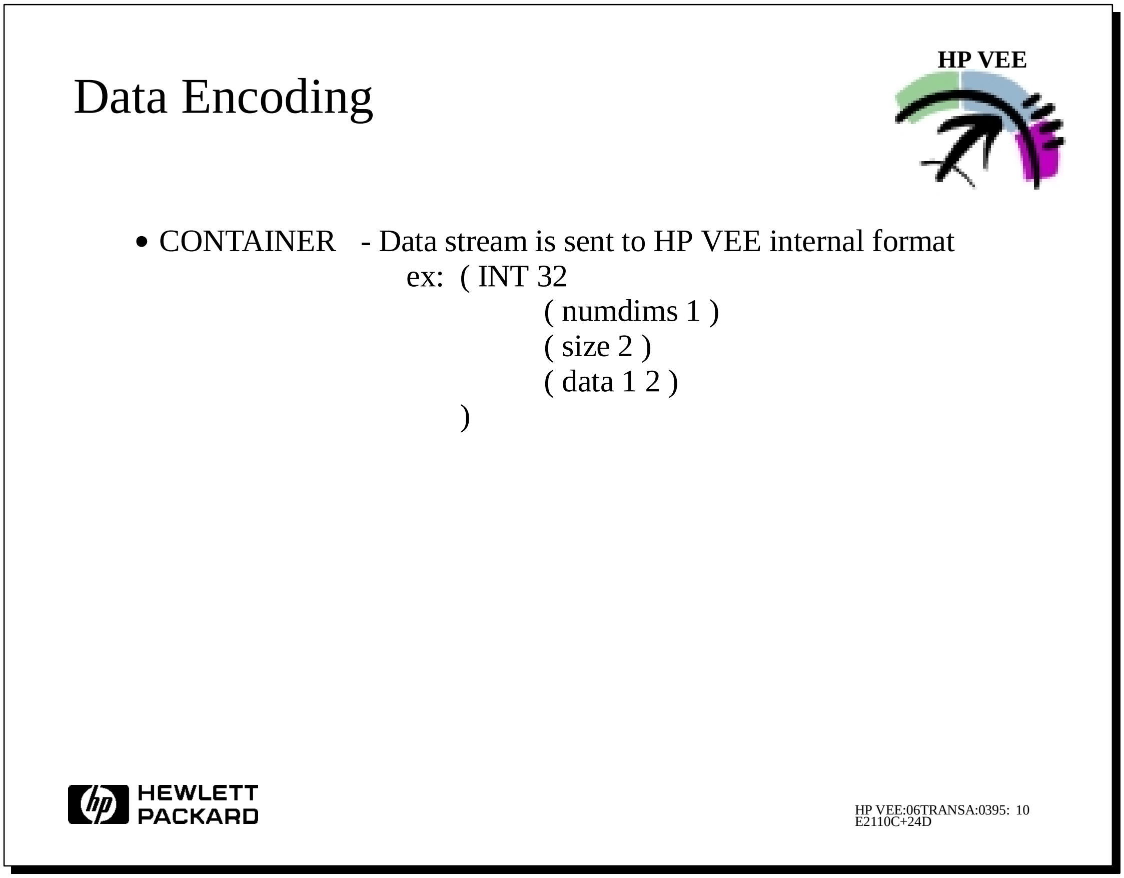
```
ex:  #12AB =  a 1 digit length
              length = 2
              data = AB
     #2101234567890 =   a 2 digit length
                        length = 10
                        data = 1234567890
```

HEWLETT PACKARD

HP VEE:06TRANSA:0395: 09
E2110C+24D

BINBLOCK -- A data stream sent as an IEEE-488.2 indefinite length block.

Useful when talking to instruments (that conform to IEEE-488.2) that capture waveforms, like scopes or digitizers as well as other instruments that would transfer blocks of binary data.



Data Encoding

HP VEE

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

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

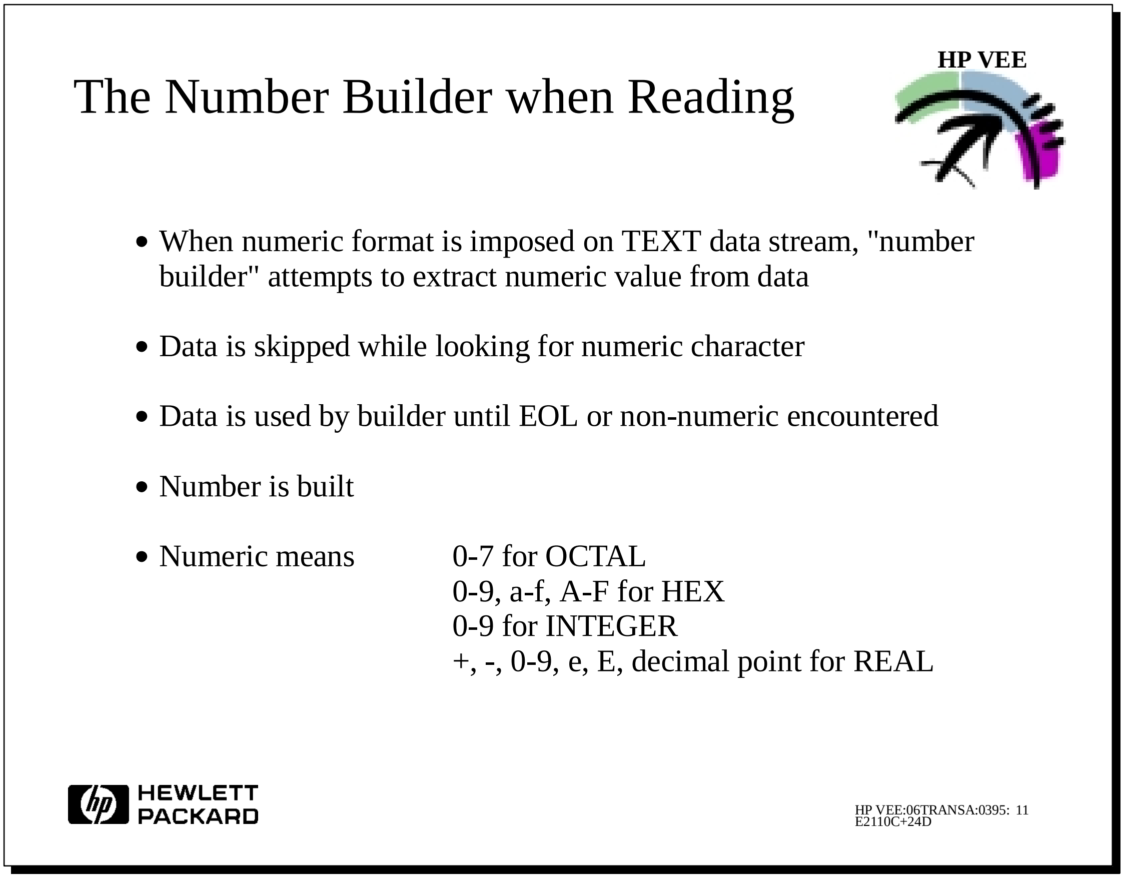
HEWLETT PACKARD

HP VEE:06TRANSA:0395: 10
E2110C+24D

CONTAINER -- A data stream in HP VEE descriptive format.

For example: (INT 32 (numdims 1) (size 2) (data 1 2))

You might want to use this format so you don’t need to worry about the encoding format.



HP VEE

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

HEWLETT PACKARD

HP VEE:06TRANSA:0395: 11
E2110C+24D

HP VEE uses a Number Builder to extract a numeric value from textual data when you specify a numeric format for READ action data. The number builder works in the following manner.

1. Skips data until it encounters a numeric character
2. Builds value from data until it encounters an EOL or non numeric character
3. Builds the final number

Valid numeric characters are:

OCTAL: 0-7
HEX: 0-9, a-f, A-F
INTEGER: 0-9
REAL: +,-,0-9,e,E,. (decimal point)

Lab 6

Moving Data To and From Files

Objective: Learn how to move data in and out of files.

Step 1
Create an HP VEE program to:
write the time of day to a file.
Generate 100 random numbers and also write them to the same file.
Calculate the **Mean** and **Standard Deviation** of the 100 numbers and write those two numbers to the file using the format of :

Mean: xxxxxx
Std Dev: yyyyyy

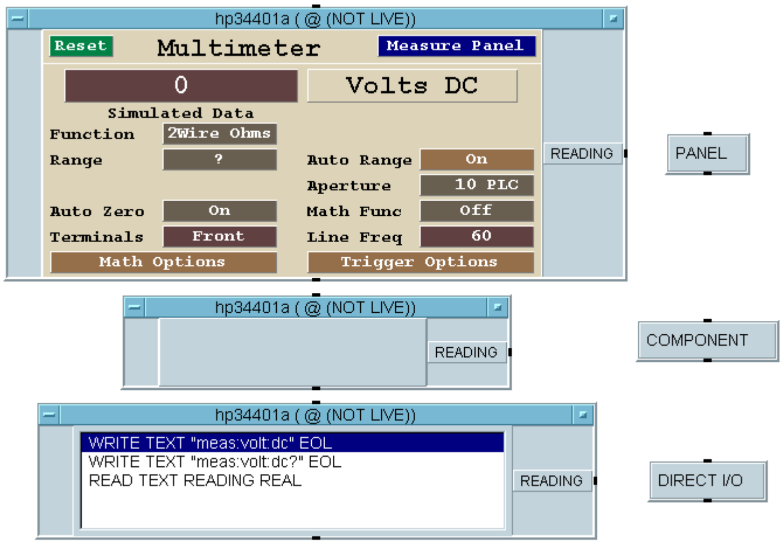
Hints: Don’t forget to write out the "Mean" and "Std Dev." words too! Take a look at the Real Format type in the File I/O Transaction Line, it has "fixed digits" capabilities....

Step 2
Create a different HP VEE program to read just the mean and standard deviation from the file. Remember that this means that you will need to skip over the other entries to get to the desired data.

Instruments

HP VEE

HP VEE Instrument I/O



Reset | Multimeter | Measure Panel

0 | Volts DC

Simulated Data

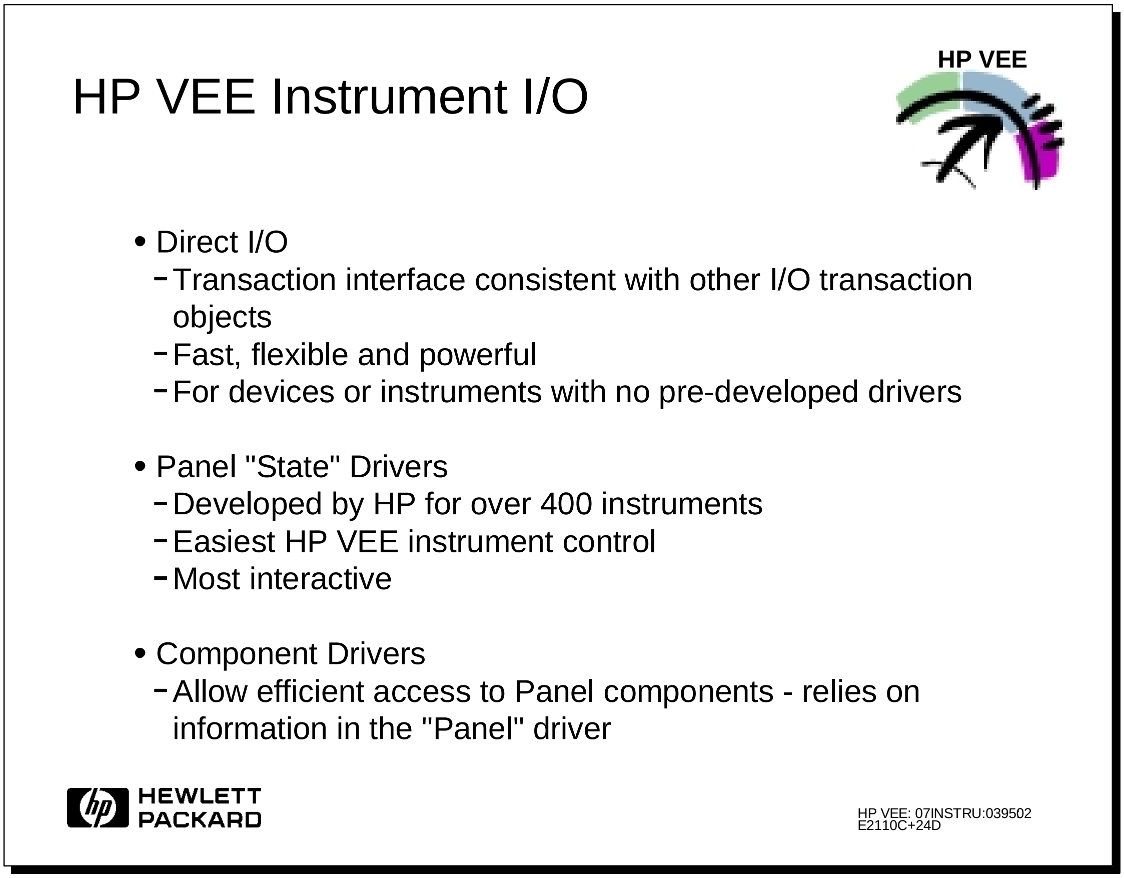
Function: 2Wire Ohms
Range: ?
Auto Range: On
Aperture: 10 PLC
Auto Zero: On
Math Func: Off
Terminals: Front
Line Freq: 60
Math Options | Trigger Options

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

HEWLETT PACKARD

HP VEE: 07INSTRU:0395: 01
E2110C+24D

These are the objects that HP VEE uses to communicate to any device that is attached to any of the supported interfaces. An example might be an instrument, a temperature chamber, or bar code readers.



HP VEE Instrument I/O

HP VEE

- 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

HEWLETT PACKARD

HP VEE: 07INSTRU:039502
E2110C+24D

HP VEE communicates with instruments in three ways.

For the very fastest interaction with instruments, or for those devices which do not have a driver, HP VEE provides Direct I/O. This object is the fastest and most flexible but requires the user to provide the actual instrument commands. The two methods below do not require this. Direct I/O uses the same transaction interface that we've previously learned about.

HP VEE uses HP Instrument drivers. We call them State drivers and they provide easy interactive instrument control. They are called state drivers because they keep track of the instrument's state or function settings. State drivers have a "Component" to store the state of an instrument setting giving you a way to transition to Component drivers.

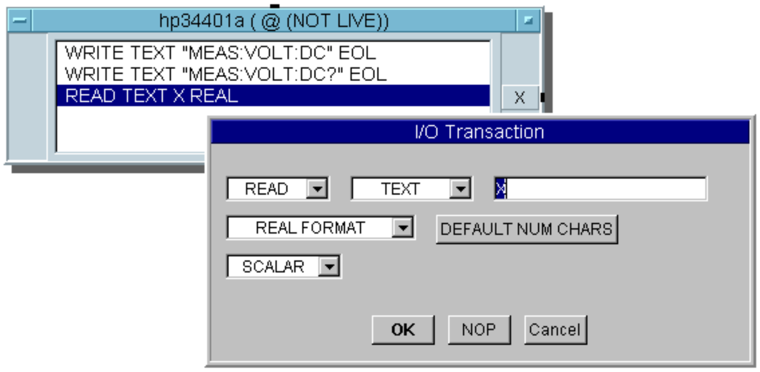
HP VEE also uses the instrument drivers in a fashion called Component drivers. These same drivers can also communicate more efficiently by not keeping track of the instrument state and not displaying the instrument information.

Help ==> Instruments will give you a list of all installed drivers.

HP VEE

Direct I/O

- 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

HP VEE: 07INSTRU:0395: 03
E2110C+24D

What if HP VEE doesn’t have an instrument driver for your favorite instrument? HP VEE can still communicate with it using Direct I/O objects. These objects use the transaction box construct that we’ve seen before.

Direct I/O provides the following actions:

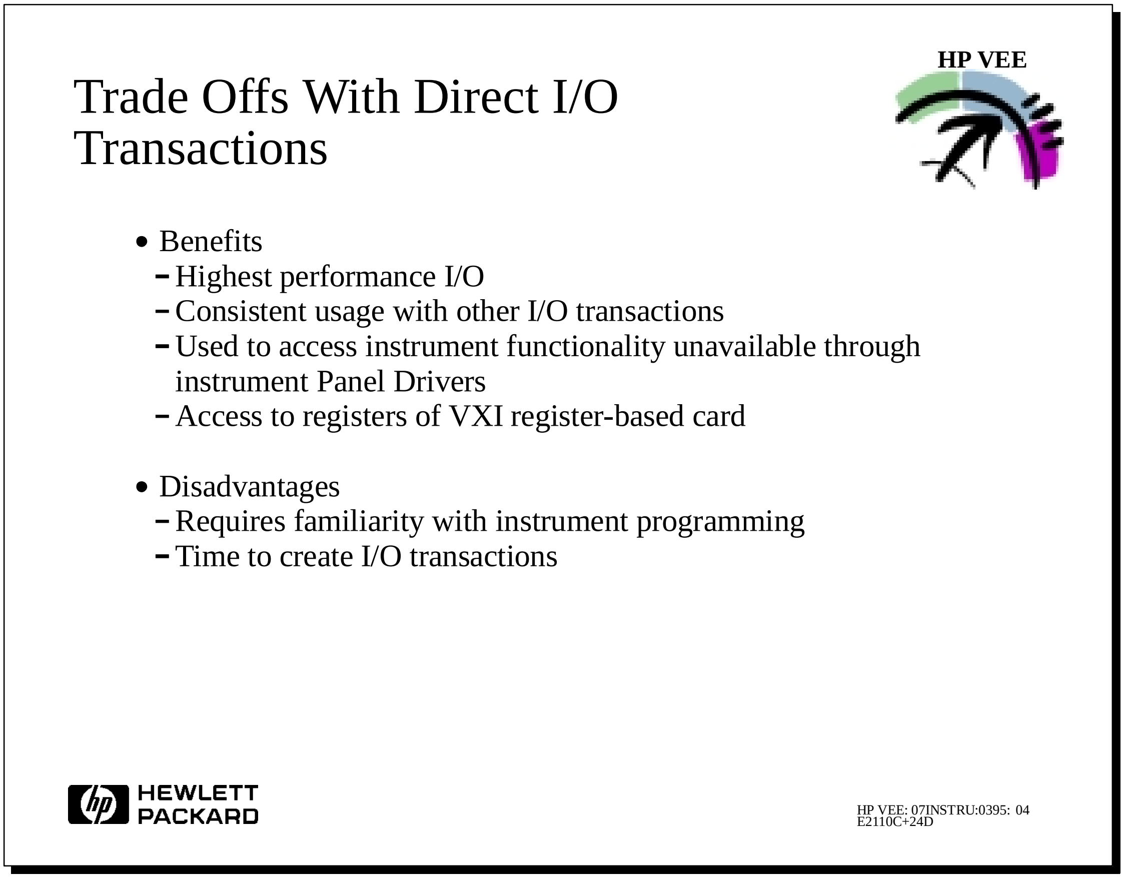
READ and WRITE data to your instrument in all formats

EXECUTE controls the instrument interface and your device

WAIT until you’re ready for the next command or until a Reading is ready/measurement has completed.

Direct I/O is similar to:

Rocky Mountain Basic	OUTPUT/ENTER
Quick BASIC	IOOUTPUT/IOENTER



HP VEE

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

HEWLETT PACKARD

HP VEE: 07INSTRU:0395: 04
E2110C+24D

What are the tradeoffs of using Direct I/O?

Direct I/O provides the following benefits:

Highest performance I/O versus Panel Drivers

Consistent interface with other I/O transactions

Access to instrument functionality unavailable through instrument drivers or for instruments without drivers

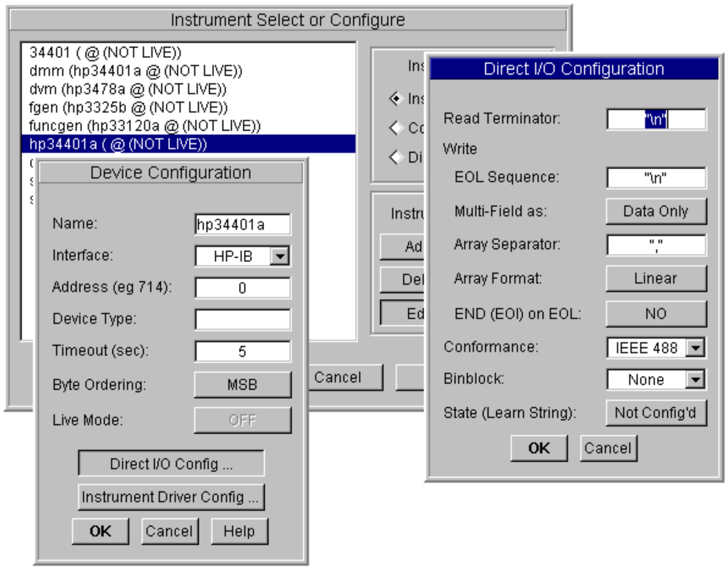
However, it also has the following disadvantages:

You must be familiar with the instrument programming commands or have access to documentation

Time to gather instrument commands. Similar to text based programming

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

HP VEE: 07INSTRU:0395: 05
E2110C+24D

Each instrument has separate configuration for Driver and Direct I/O.

Direct I/O configuration specifies:

Terminators and EOL sequences

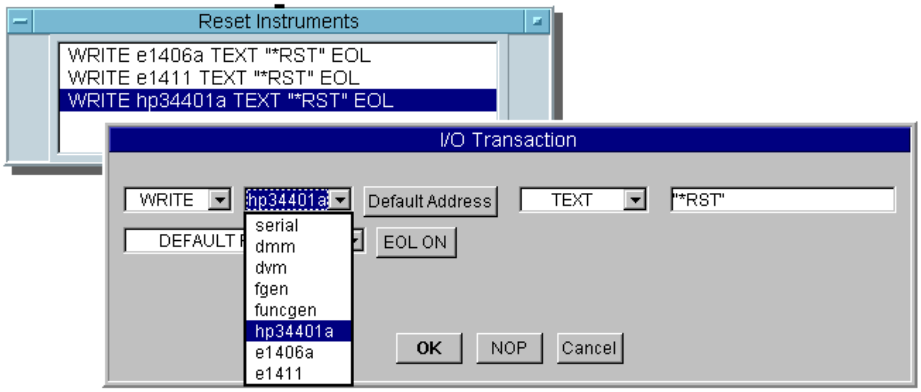
Formatting of array data

Conformance to IEEE 488 or 488.2

Information needed to save and restore instrument learn string

HP VEE

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

HEWLETT PACKARD

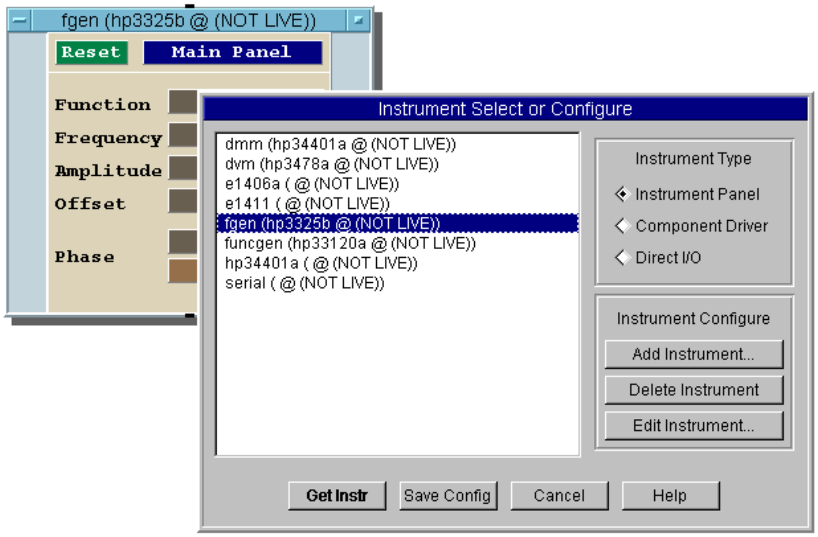
HP VEE: 07INSTRU:0395: 06
E2110C+24D

The MultiDevice Direct I/O is used to communicate to different instruments or devices through one MultiDevice Direct I/O object. A typical use would be to have a single object perform a similar task to several instruments, such as RESET or SETUP.

I/O ==> Advanced I/O ==> MultiDevice Direct I/O

Instrument Select or Configure

HP VEE



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

HEWLETT PACKARD

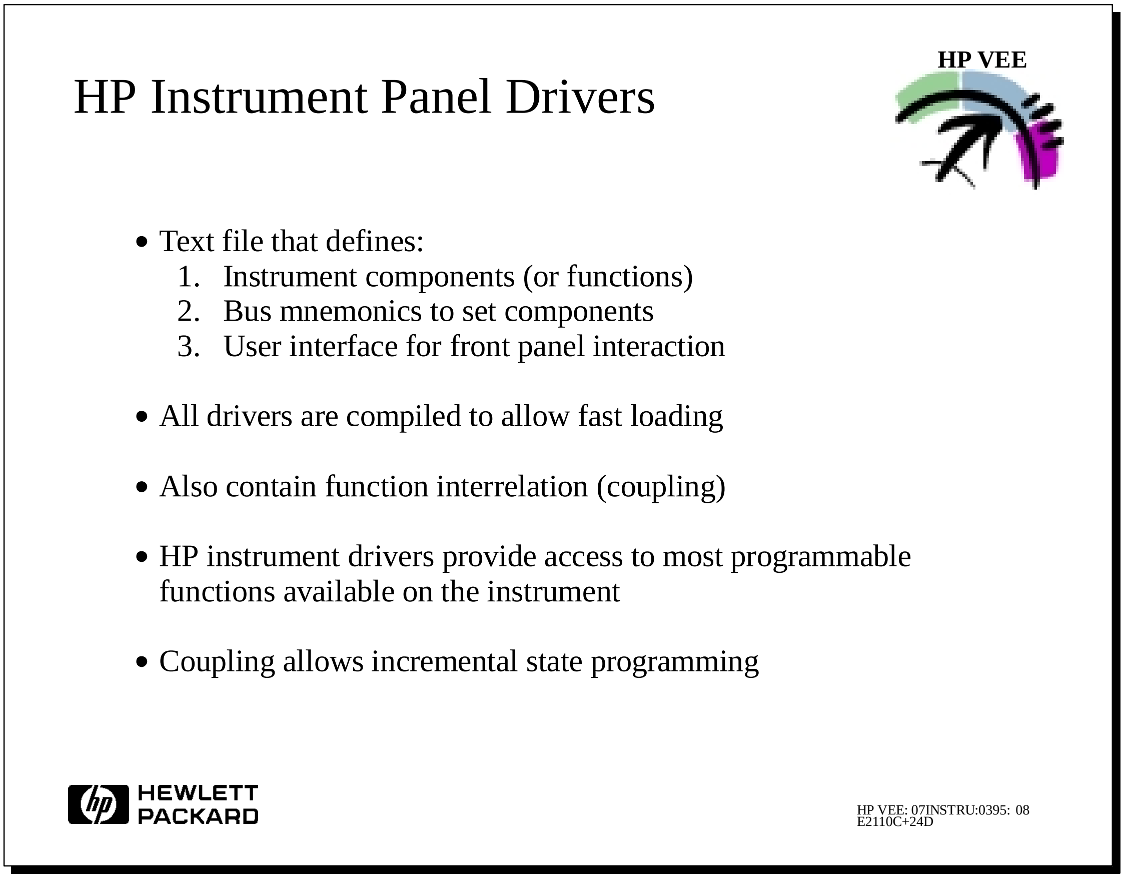
HP VEE: 07INSTRU:0395: 07
E2110C+24D

I/O ==> Instrument ...

Add Instrument ...
This option will add a new instrument to the selection. It will prompt with a default Device Configuration dialog box.

Delete Instrument ...
This option will delete the hi-lighted Instrument.

Edit Instrument ...
This option will show the Device Configuration dialog box for the hi-lighted Instrument.



HP VEE

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

HEWLETT PACKARD

HP VEE: 07INSTRU:0395: 08
E2110C+24D

These instrument drivers are text files that define:

Instrument components or functions

Instrument commands or mnemonics that control the instrument functions

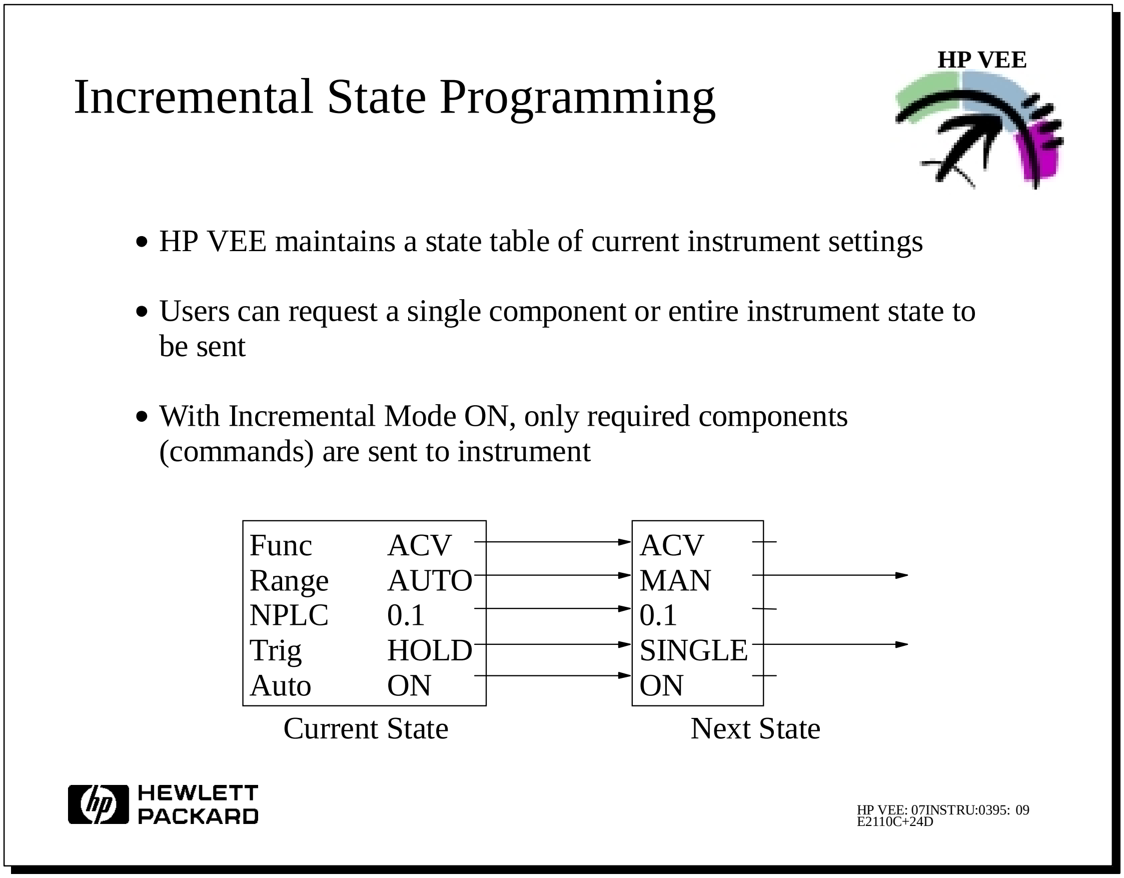
User interface for front panel interaction

The drivers also contain information on how various instrument functions interact. This interaction is called coupling.

HP instrument drivers provide access to most of the programmable instrument functions available in the device. These drivers also permit you to use incremental state programming since it tracks the state of the instrument.

hpidc **-** HP Instrument Driver Compiler **-** Operating System specific.

Note: You can also use the driver writing tools provided for HP ITG to write new drivers.



HP VEE

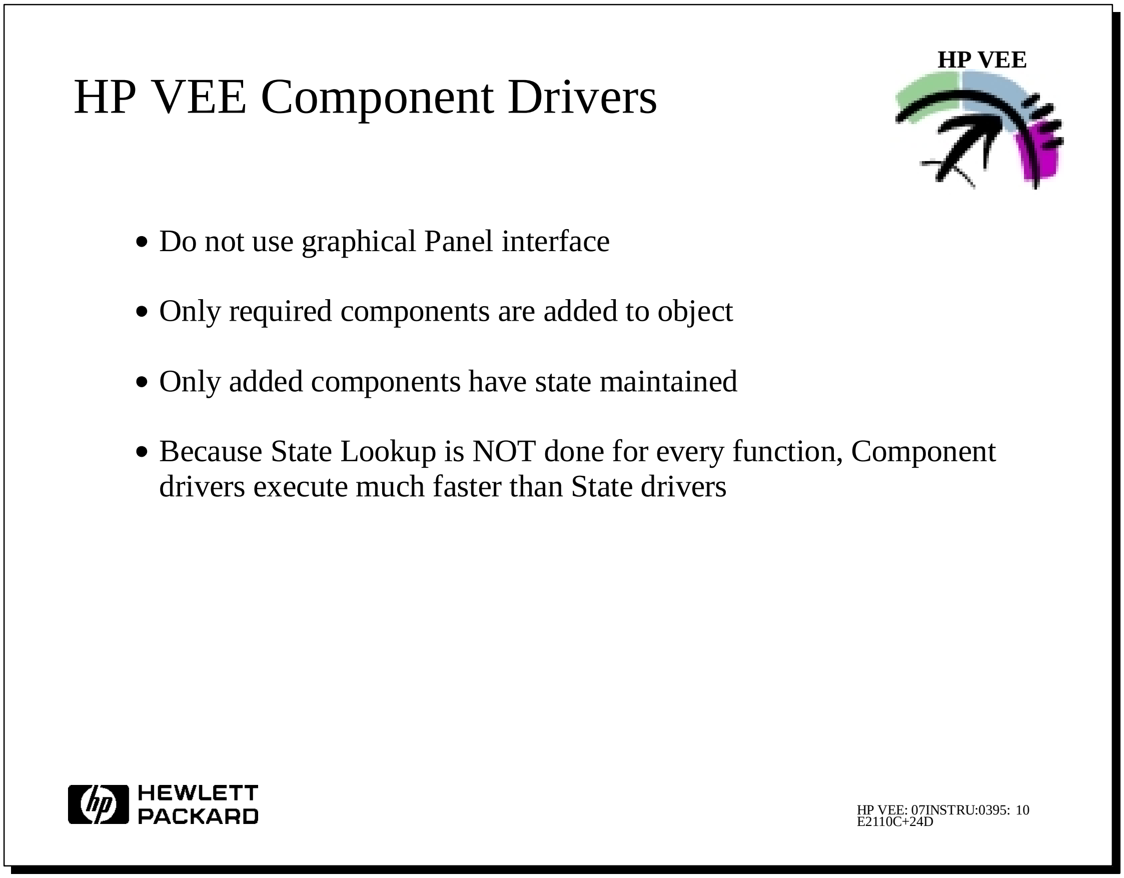
Incremental State Programming

- 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

Current State		Next State
Func	ACV	ACV
Range	AUTO	MAN
NPLC	0.1	0.1
Trig	HOLD	SINGLE
Auto	ON	ON

HP VEE: 07INSTRU:0395: 09
E2110C+24D

With incremental state programming, HP VEE maintains a state table of current settings for the instrument. You can then request HP VEE send an individual component or function change or an entirely new instrument state. When you have Incremental Mode ON, HP VEE sends only those components (commands) required to update the state of the instrument.



HP VEE

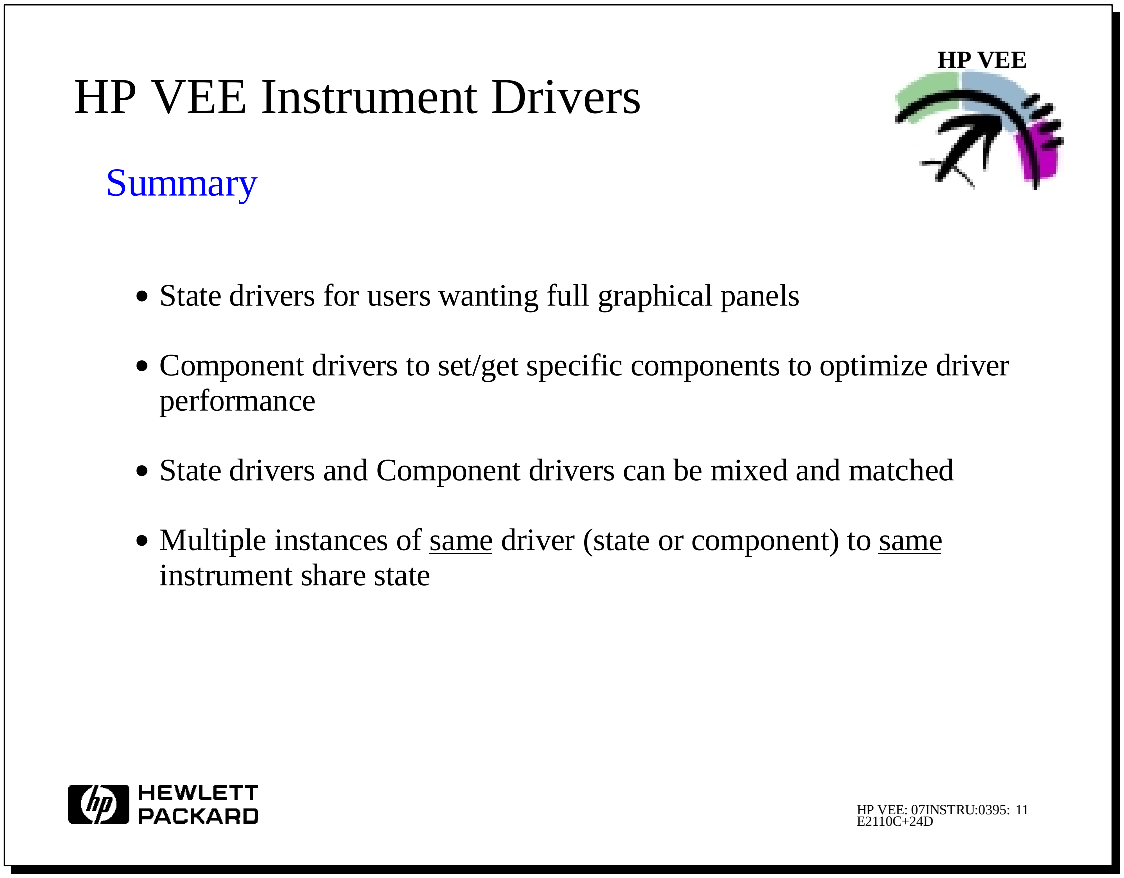
HP VEE Component Drivers

- 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

HEWLETT PACKARD

HP VEE: 07INSTRU:0395: 10
E2110C+24D

With HP VEE Component drivers, you only add the functions you need to the object. You simply add an input or output terminal, choosing from the list of available functions. This driver only tracks the state of the functions you add. Since Component drivers do not look up the entire state of the instrument they execute faster than State drivers.



HP VEE

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

HEWLETT PACKARD

HP VEE: 07INSTRU:0395: 11
E2110C+24D

In summary:

State drivers provide a full graphical interface for the instrument

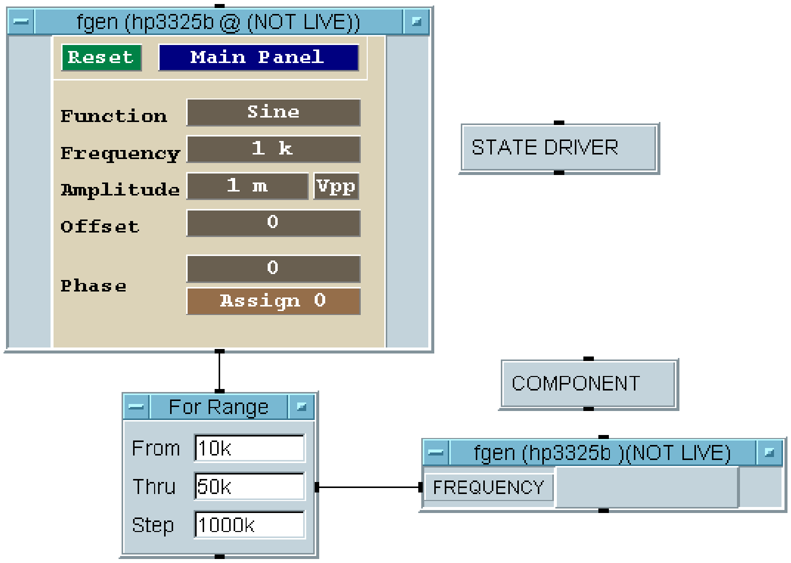
Component drivers set and get specific instrument components to optimize performance

You can use State drivers and Component drivers in the same program. See the next slide for an example of this

Multiple instances of the same driver (either state or component) to the same instrument share the same state table. Same instrument refers to Device Configuration Name

HP VEE

State and Component Drivers



fgen (hp3325b @ (NOT LIVE))

Reset | Main Panel

Function: Sine
Frequency: 1 k
Amplitude: 1 m Vpp
Offset: 0
Phase: 0
Assign 0

STATE DRIVER

For Range

From: 10k
Thru: 50k
Step: 1000k

COMPONENT

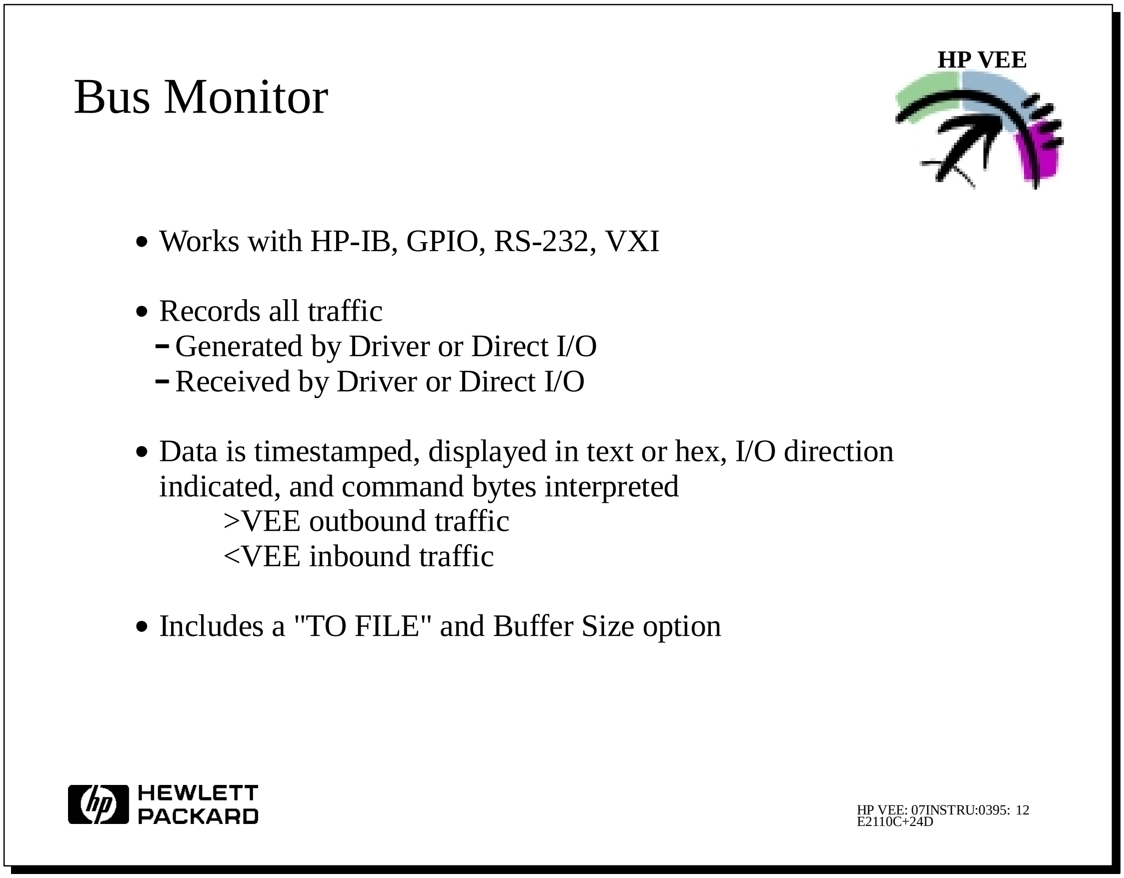
fgen (hp3325b)(NOT LIVE)

FREQUENCY

HEWLETT PACKARD

HP VEE: 07INSTRU:0395: 11a
E2110C+24D

This is a sample of using a State driver and Component driver in the same program. The State driver is used to setup the instrument into a known configuration and then the Component driver is used to repeatedly send new values to the instrument.



HP VEE

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

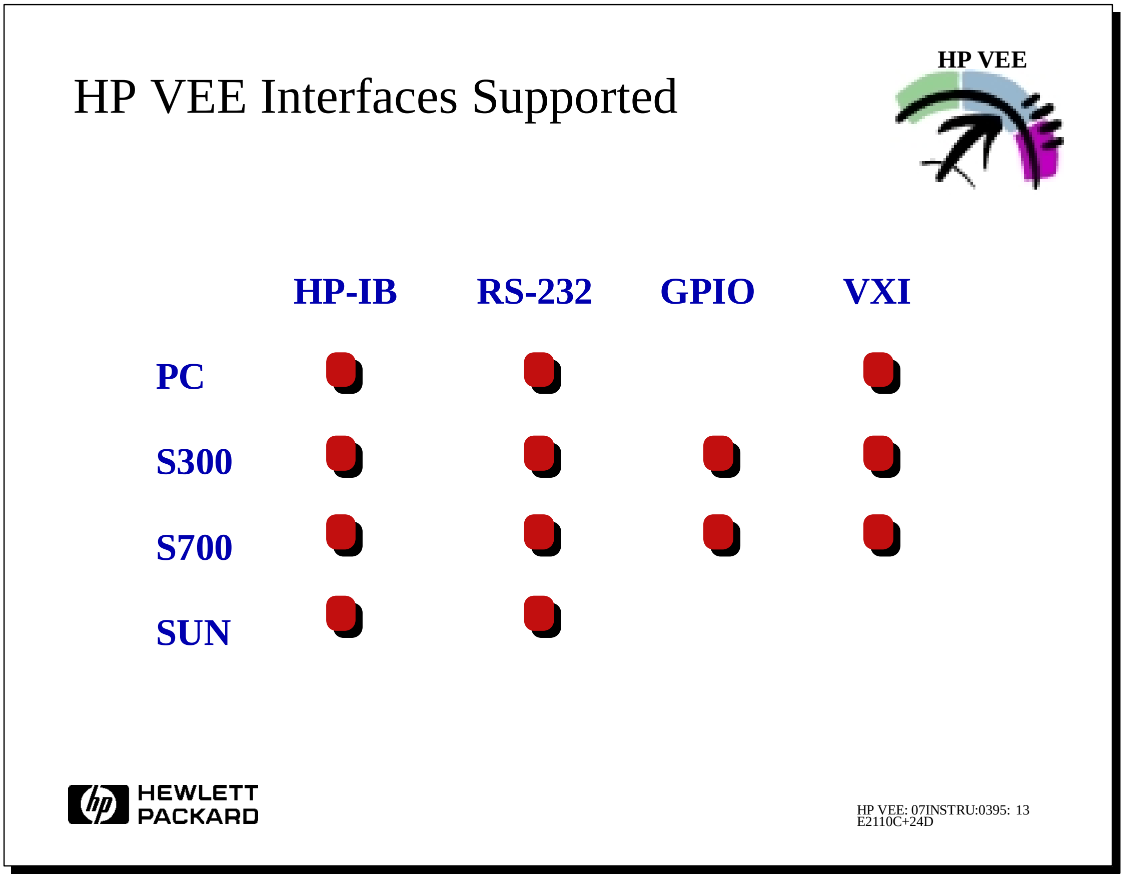
  <VEE inbound traffic
- Includes a "TO FILE" and Buffer Size option

HEWLETT PACKARD

HP VEE: 07INSTRU:0395: 12
E2110C+24D

To help you debug your instrument control models, HP VEE provides the Bus Monitor object. The Bus Monitor works with all the interfaces configured in your computer, recording traffic generated by instrument drivers or Direct I/O, or received by the controller. The monitor timestamps the data, displays it as hex digits, indicating the I/O direction. It also interprets command bytes for you.

NOTE: The Bus Monitor only records I/O traffic related to HP VEE. It does not monitor or record traffic generated or received by other devices or programs, such as an external BASIC program.



HP VEE Interfaces Supported

HP VEE

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

HEWLETT PACKARD

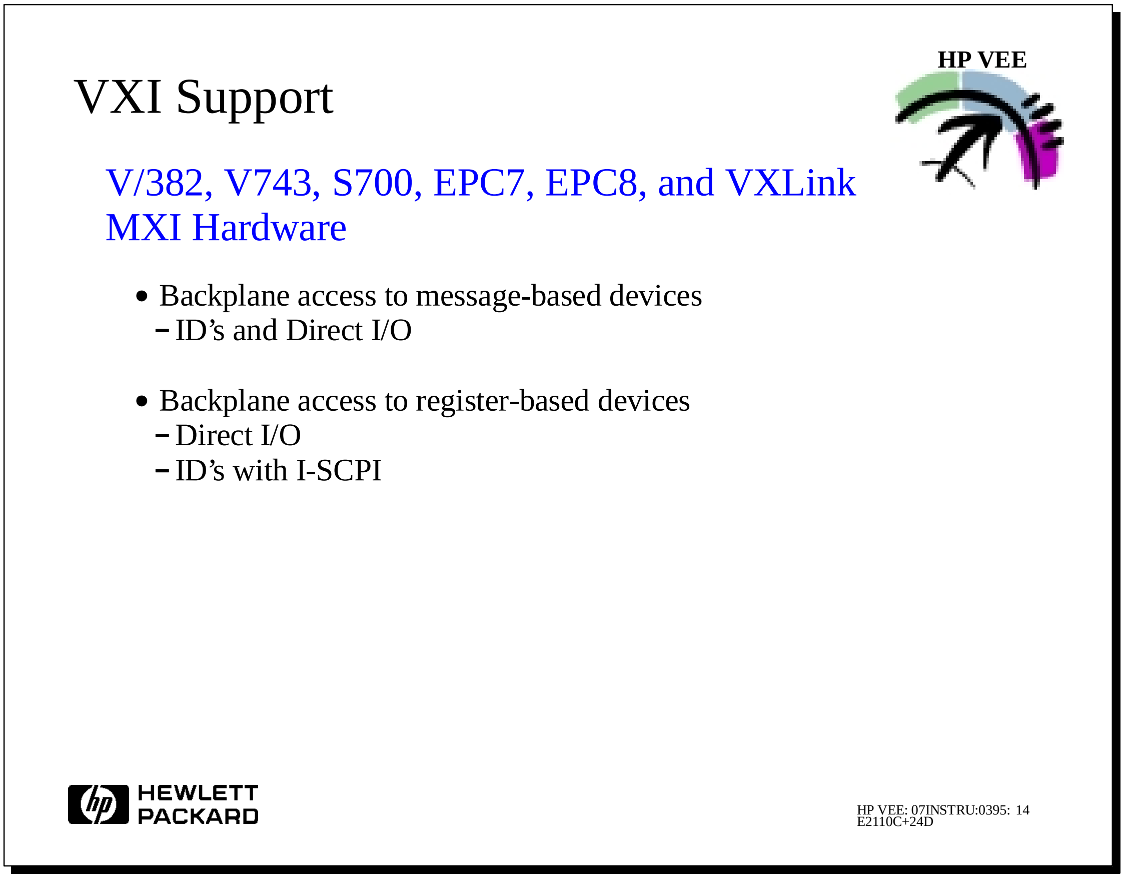
HP VEE: 07INSTRU:0395: 13
E2110C+24D

HP-IB Also known as IEEE-488 and GPIB support IEEE-488 and 488.2 type instruments. Configurable item.

GPIO General Purpose I/O (Parallel)

RS-232 RS-232 Serial interface

VXI VXI is referenced as an interface rather than an instrument. S300 would be V/382. S700 would be MXI or V/743. PC includes EPC7, EPC8, or VXLink.



HP VEE

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

HEWLETT PACKARD

HP VEE: 07INSTRU:0395: 14
E2110C+24D

Message-based Access

HP VEE will support message-based back plane access to message based devices. This access is accomplished through the V/382 resource manger and SICL/PIL. This capability is available from both instrument drivers and Direct I/O transactions.

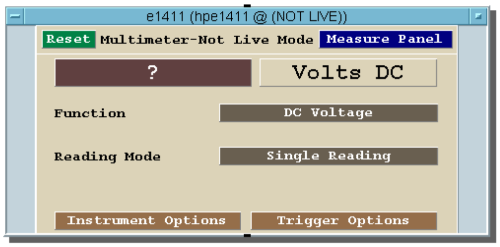
Register-based Access

I-SCPI will give access to register-based devices that have an I-SCPI driver. I-SCPI is done with either a Shared Library or DLL.

Register-based access to register devices is accomplished through the use of a Direct I/O object. The supported transactions are read and write register and read and write memory.

HP VEE

VXI Register-Based Access



e1411 (hpe1411 @ (NOT LIVE))

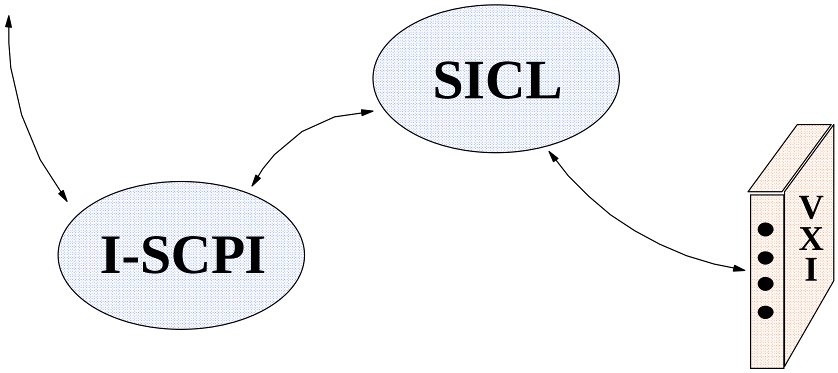
Reset Multimeter-Not Live Mode Measure Panel

? Volts DC

Function: DC Voltage

Reading Mode: Single Reading

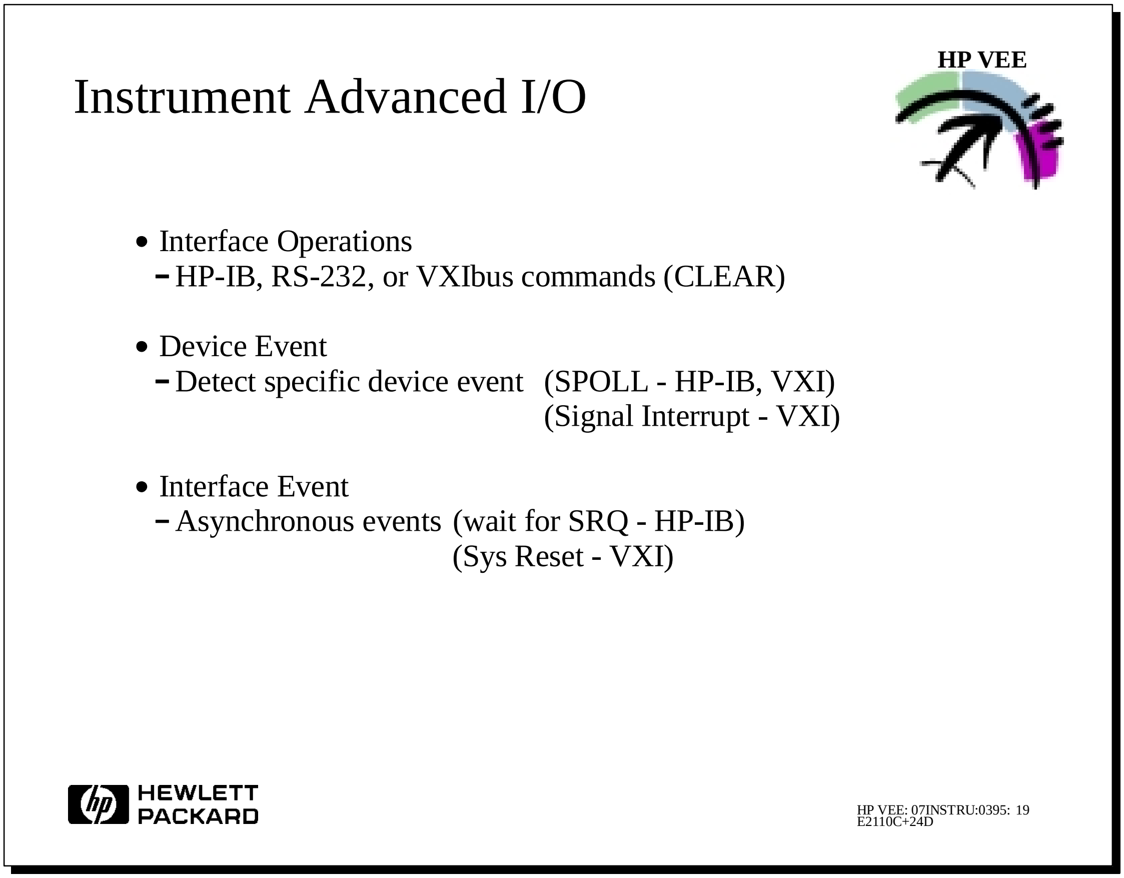
Instrument Options | Trigger Options



HEWLETT PACKARD

HP VEE: 07INSTRU:0395: 14a
E2110C+24D

In an embedded environment, you can use State drivers or Direct I/O to access register-based VXI cards with I-SCPI. In this scenario, you would send the instrument command strings such as "MEAS:VOLT:DC", which I-SCPI will turn into the appropriate register peeks and pokes. The intent is to allow you to realize the speed advantage that Register-based VXI provides, while sending the instrument human-understandable instructions. I-SCPI is a library that will parse the commands for the VXI card.



HP VEE

Instrument Advanced I/O

- Interface Operations
  - HP-IB, RS-232, or VXIbus commands (CLEAR)
- Device Event
  - Detect specific device event (SPOLL - HP-IB, VXI)
(Signal Interrupt - VXI)
- Interface Event
  - Asynchronous events (wait for SRQ - HP-IB)
(Sys Reset - VXI)

HEWLETT PACKARD

HP VEE: 07INSTRU:0395: 19
E2110C+24D

With the addition of the VXI bus to the supported I/O matrix, comes the need to add a few new transactions to what used to be Advanced HP-IB Operations. Those actions are now located under the menu sequence I/O -> Advanced I/O. The actions have been grouped by their scope of operation (either interface wide or device specific) and the nature of the service they provide (either performing defined operations or detecting events).

Interface Operations perform a specific, interface wide action. They can be used with any of the interfaces currently defined in your configuration. The two actions that specifically address VXI are CLEAR and TRIGGER.

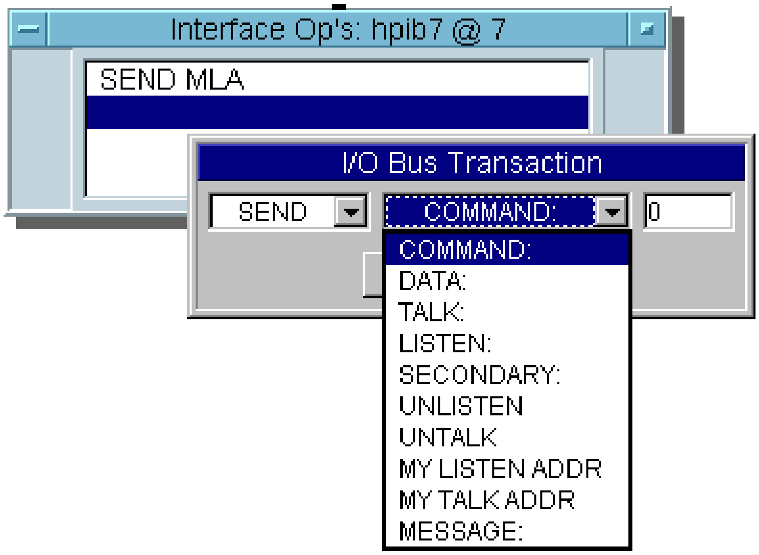
Interface events are objects that capture asynchronous interface events for VXI and HP-IB. These objects can be configured to detect various interface events. The objects may wait for the event, giving up execution to other parallel threads, or simply return a Boolean or other indicator of the interface's state.

If the Interface Event object has been configured to wait for the event, then upon receipt of the event, any parallel thread hosted by this object will have priority over any other parallel threads and execute to completion. If the Interface Object has been configured to simply return an indicator of the device's state, any thread it is a part of will have normal priority and execute in parallel with other threads.

Device Event transactions let you detect events associated with the device, as opposed to events which affect the interface as a whole. The Event can be SPOLL, SPOLL on SRQ, or Signal Interrupt.

HP VEE

Using Interface Operations



Interface Op's: hpib7 @ 7

SEND MLA

I/O Bus Transaction

SEND · COMMAND: · 0

HP VEE also provides Advanced Interface Operations.

EXECUTE - Sends non-addressed (global) bus commands to all devices.
- ABORT, CLEAR, TRIGGER, REMOTE, LOCAL, LOCAL LOCKOUT, LOCK INTERFACE, UNLOCK INTERFACE

SEND - Sends custom command or data transactions

COMMAND - Sends data with ATN line TRUE (HP-IB command)

MESSAGE - IEEE-488 defined mnemonic commands sent with ATN line TRUE DCL, TCT, etc.

DATA - Data sent with ATN line FALSE (HP-IB data)

You can also use the commands:

TALK, LISTEN, UNLISTEN, UNTALK, MTA, MLA, SECONDARY

Lab 7a

Using MultiDevice Direct I/O with instruments

Objective: To learn to use Direct I/O to communicate with instruments.

Using the **I/O=>Advanced I/O=>MultiDevice Direct I/O** object, set up the transactions to:

Step 1
Reset the HP34401A and the HP33120A by sending:
***RST** to each of them.

Step 2
Query the instruments for their identification by writing:
***IDN?** to each of them.

Read back the IDN strings from each instrument and send them to an **AlphaNumeric** display

Step 3
Send a message to the HP34401A Front Panel Display.
The I/O Transaction format for sending a message is: "Display: Text <quoted string>"

Lab 7b

Component Driver

Objective: To learn how the **Component Driver** concept can be used in instrument I/O.

Load the **compdrv.vee** program and **RUN** it. Investigate how it is constructed.

Lab 7c

Using HP Instrument Drivers

Objective: To learn how **State Drivers** are used in instrument I/O.

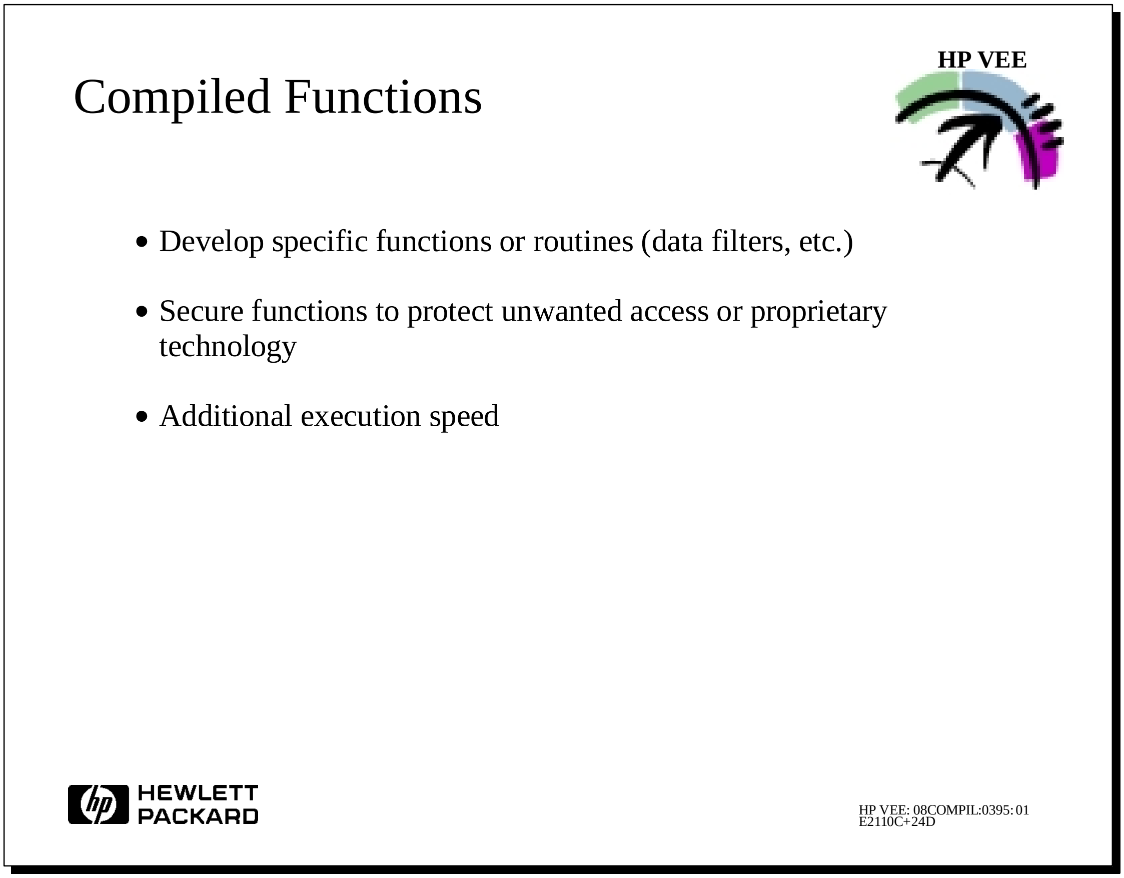
Step 1
Using the HP34401A and the HP33120A, measure and plot the frequency response of the low pass filter over the range of 100 Hz to 10 MHz. Use **Instrument Panels** for these instruments.

Hints: Remember that the Function Generator is putting out a Sine wave! DC voltage measurements won’t work to measure the frequency response of the filter. Also, don’t bother with the swept sine functions of the HP33120A, you will want to step the frequency and measure the response and step the frequency again ...also try Log stepping the frequency. Check out the Log plotting of the **X vs Y** Plot too!

Step 2
Repeat the lab using **Component Drivers** and note if there are any speed improvements.

Step 3
Now do the setup with **Instrument Panels** and **Component Drivers** for repetitive functions.

Compiled and Remote Functions



Compiled Functions

HP VEE

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

HEWLETT PACKARD

HP VEE: 08COMPIL:0395: 01
E2110C+24D

Compiled Functions

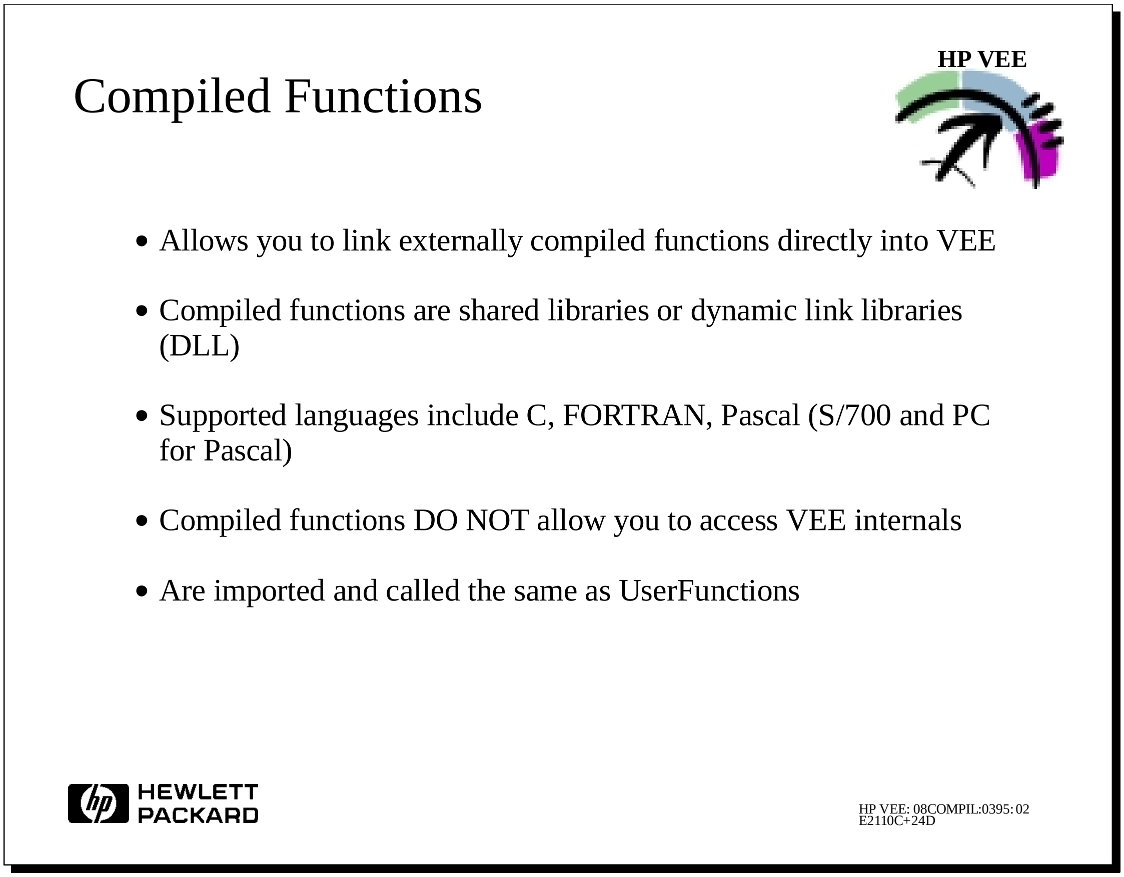
Compiled Functions allow you to integrate externally compiled functions directly into HP VEE-Test. You can think of them in much the same way as you thought of CSUB'sin HP BASIC. Compiled Functions allow you to extend HP VEE'scapabilities by integrating routines you have written in C, C++, FORTRAN, and Pascal (Pascal calls are only supported on S/700 systems and PCs).

Some reasons you might want to use Compiled Functions would include but are not limited to:

Allowing you to develop your own data filters and directly integrate them

Secure the functions in cases where proprietary routines or unwanted access are concerns

Additional execution speed



Compiled Functions

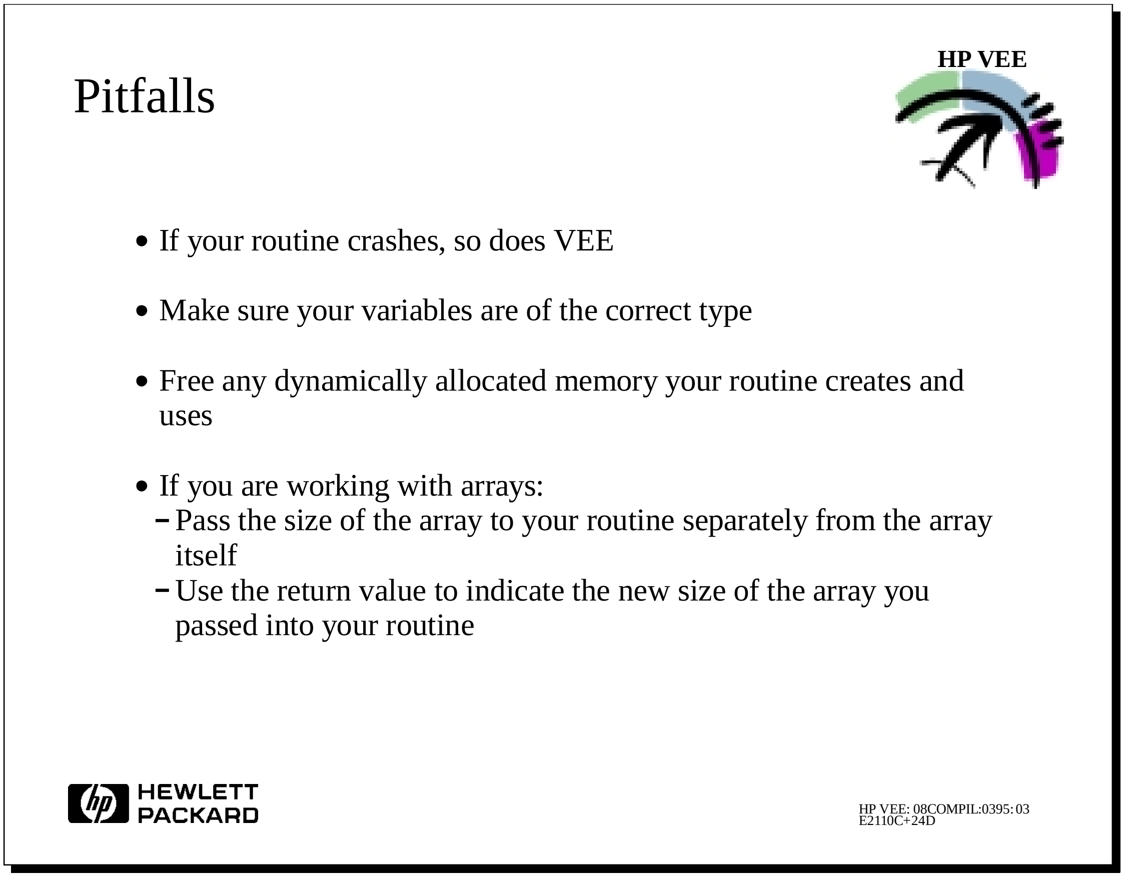
HP VEE

- 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

HEWLETT PACKARD

HP VEE: 08COMPIL:0395: 02
E2110C+24D

You can use any facilities available from the operating system to include RPC procedures, math routines, instrument I/O, etc. You specifically cannot access any of HP VEE's internals. Compiled Functions are functions that you write without access to any of the routines HP VEE uses to build objects or perform calculations.



Pitfalls

HP VEE

- 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

HEWLETT PACKARD

HP VEE: 08COMPIL:0395: 03
E2110C+24D

With the enhanced capabilities of HP VEE come potential pitfalls. Because your routine is now a part of HP VEE itself, if any part of your routine has a mistake, that mistake propagates back to HP VEE.

It is entirely possible to crash HP VEE if your routine crashes. Be careful to free any dynamically allocated memory that your routine needs.

Always constrain the data in terminals of the Compiled Function object to accept only the data type and shape that your routine expects. If your routine accepts arrays as part of your application, make sure that your routine has a pointer valid for the type of data you are examining, and make sure that your routine checks the size of the array it is working on. The best way to do this is to pass in, from HP VEE, the size of the array you are using as an input separate from the array itself.

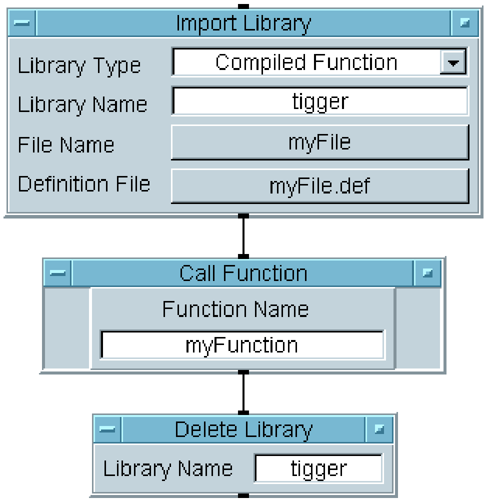
When you use a Compiled Function the best design goal is to keep the routine’s purpose highly focused at a specific task. Use Compiled Functions only when the capability or performance you need is not available via an Execute Program or when you need performance not available from HP VEE or through a User Function.

Compiled Functions

HP VEE

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

HEWLETT PACKARD

HP VEE: 08COMPIL:0395:04
E2110C+24D

To use a Compiled Function, you follow a similar process to the one used for User Functions.

Device ==> Function ==> Import

This is the same Import object as in a User Function.

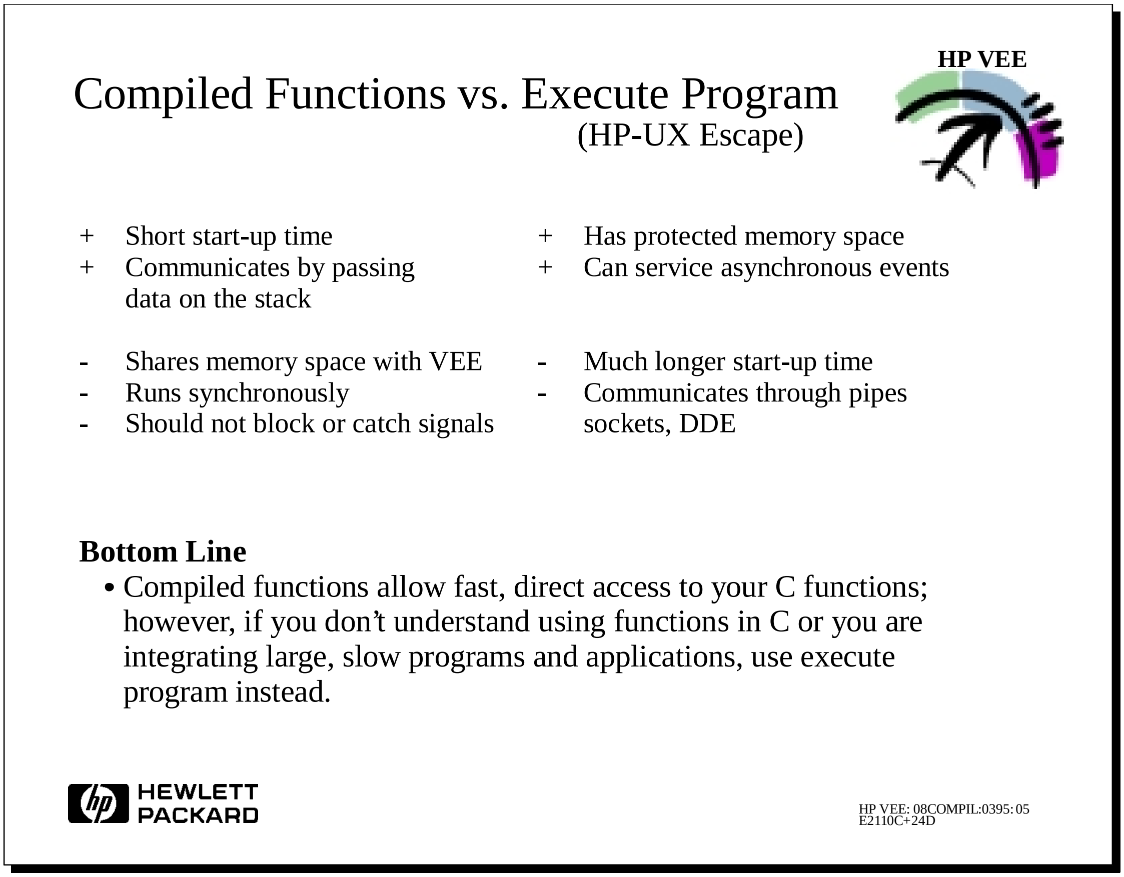
When you choose "Compiled Function" from the "Library Type" selection box, you will see an additional field at the bottom of the box. That field is a selection box that allows you to choose the file that contains the definition of your compiled function. We'll discuss the structure of the definition file later in this section.

Once you have imported the library, HP VEE treats the compiled function exactly the same as a User Function that has been imported from disk. You can call the Compiled function from a Call object or a formula box (or expression field).

Once again, use the Load Lib choice from the Import object's menu to manually load the compiled function into HP VEE. After Load Lib is selected, you can then choose Select Function from the Call object's menu. This will give a choice of all available functions.

Another interesting selection in the Call object's menu is Configure Pinouts. Selecting this adds the appropriate number and type of input and output terminals to your object. Just as with User Functions, pinouts are also configured automatically when you select a Function from the "Select Function" choice. A call to a compiled function gets the information about the appropriate pinouts from a function definition file that you create.

If you are a C programmer, using a mix of Compiled Functions and User Function can be very powerful. As you prototype your application and refine it for added performance, you might first design your function as a user object that you make into a UserFunction. This is the easiest and most consistent way to create a UserFunction. Then, as you need additional performance, you could implement your design in a compiled language. Your program can then take advantage of the additional speed, while needing to change only a single import object.



HP VEE

Compiled Functions vs. Execute Program
(HP-UX Escape)

+	Short start-up time	+	Has protected memory space
+	Communicates by passing data on the stack	+	Can service asynchronous events
-	Shares memory space with VEE	-	Much longer start-up time
-	Runs synchronously	-	Communicates through pipes sockets, DDE
-	Should not block or catch signals		

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.

HEWLETT PACKARD

HP VEE: 08COMPIL:0395:05
E2110C+24D

There are several valid reasons to use (and not use) Compiled Functions vs. a standard escape to the operating system. Because the Compiled Function actually ties to HP VEE, and can stop a HP VEE program, it is always safer to integrate code in another process (Execute Program).

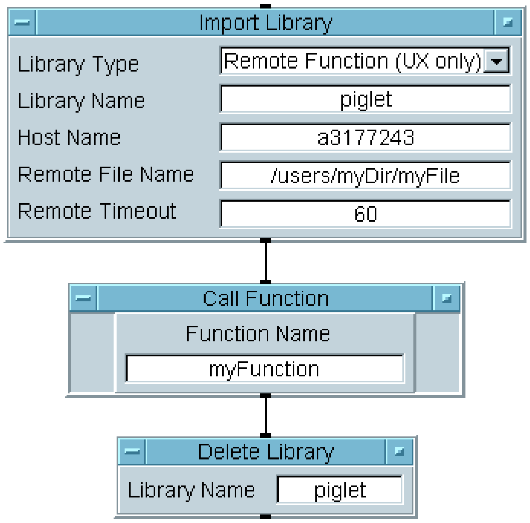
What you give up with Execute Program is speed. If you want to integrate specific functions for speed, Compiled Functions is the best way. You don't have to suffer the decreased performance as a result of waiting for another process to be initiated.

HP VEE

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

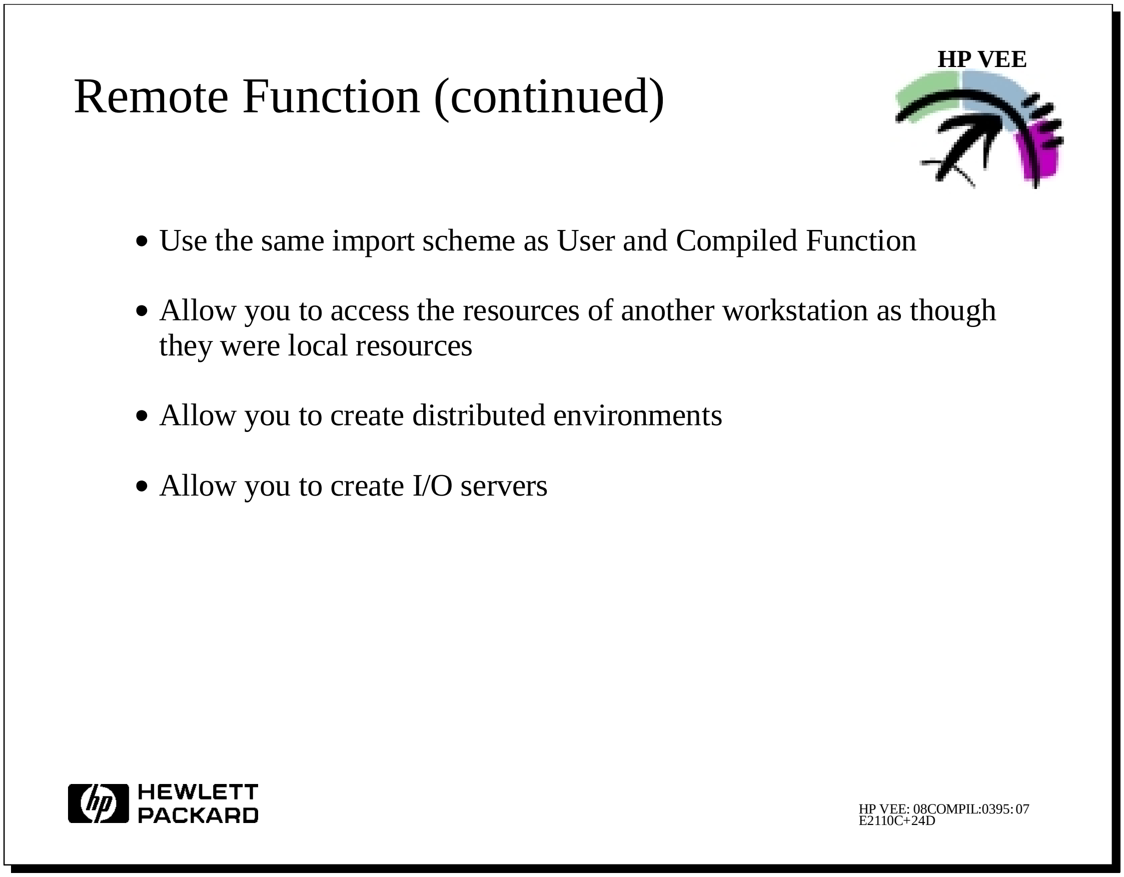
HEWLETT PACKARD

HP VEE: 08COMPIL:0395:06
E2110C+24D

The Remote Function is imported by the Import Library Object; called by the Call Object and deleted by the Delete Library Object.

By selecting Remote Function on the Import Object, you will see that the object now has five fields. The library type and name work the same as with User Functions and Compiled Functions. The remote host name field is the name of the remote workstation as assigned by /etc/hosts or the equivalent IP address. The remote file name field specifies the name of a saved HP VEE program that contains User Functions. The last entry specifies a time-out. The import object will wait no longer than the specified amount of time for the remote host to respond.

As with Compiled Functions or Imported User Functions, you must Import the function before you can call it. Again, selecting Load Lib from the object menu will allow you to see what functions are available under the Call object's menu choice 'Select Function'.



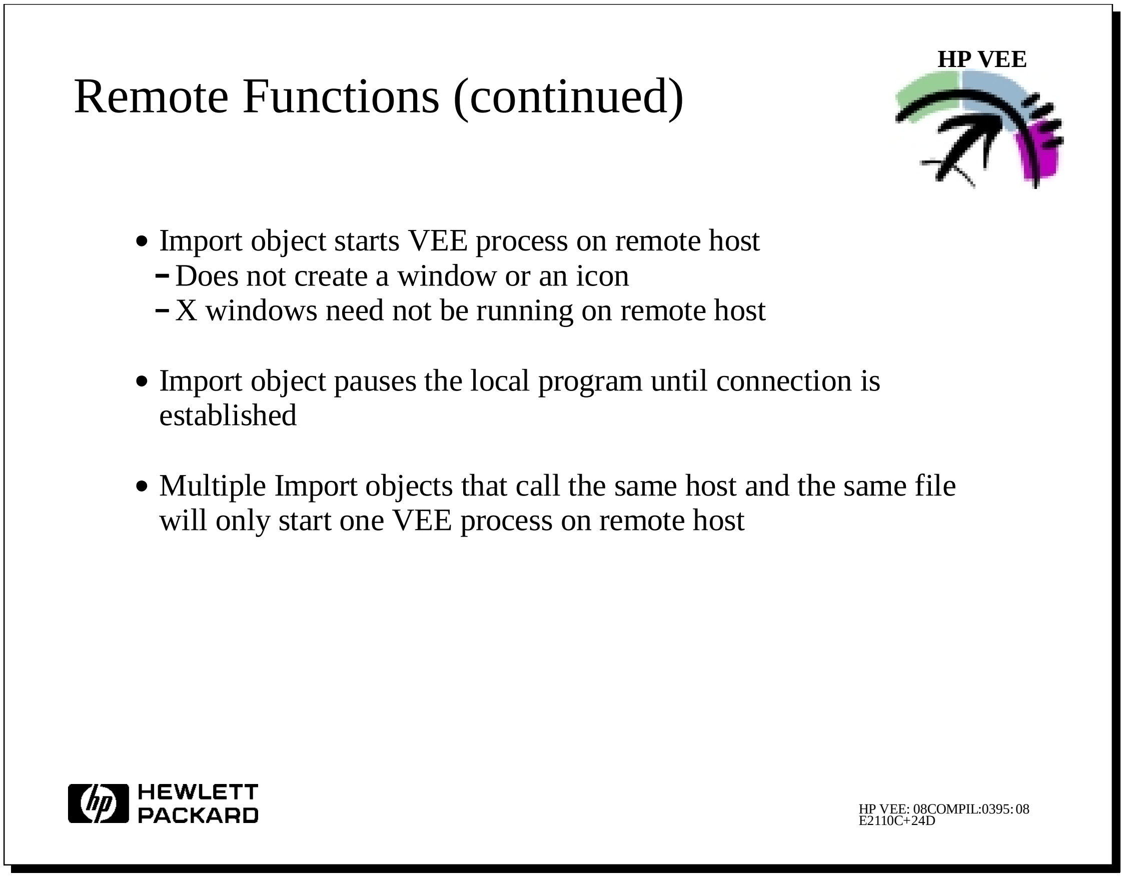
Remote Function (continued)

HP VEE

- 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

HEWLETT PACKARD

HP VEE: 08COMPIL:0395: 07
E2110C+24D



HP VEE

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

HEWLETT PACKARD

HP VEE: 08COMPIL:0395: 08
E2110C+24D

When the Import object is executed (either from Load Lib or programmatically), the remote machine will start up a HP VEE process and load in the file named in the local Import Object. The two HP VEE processes are connected over the network and are able to communicate.

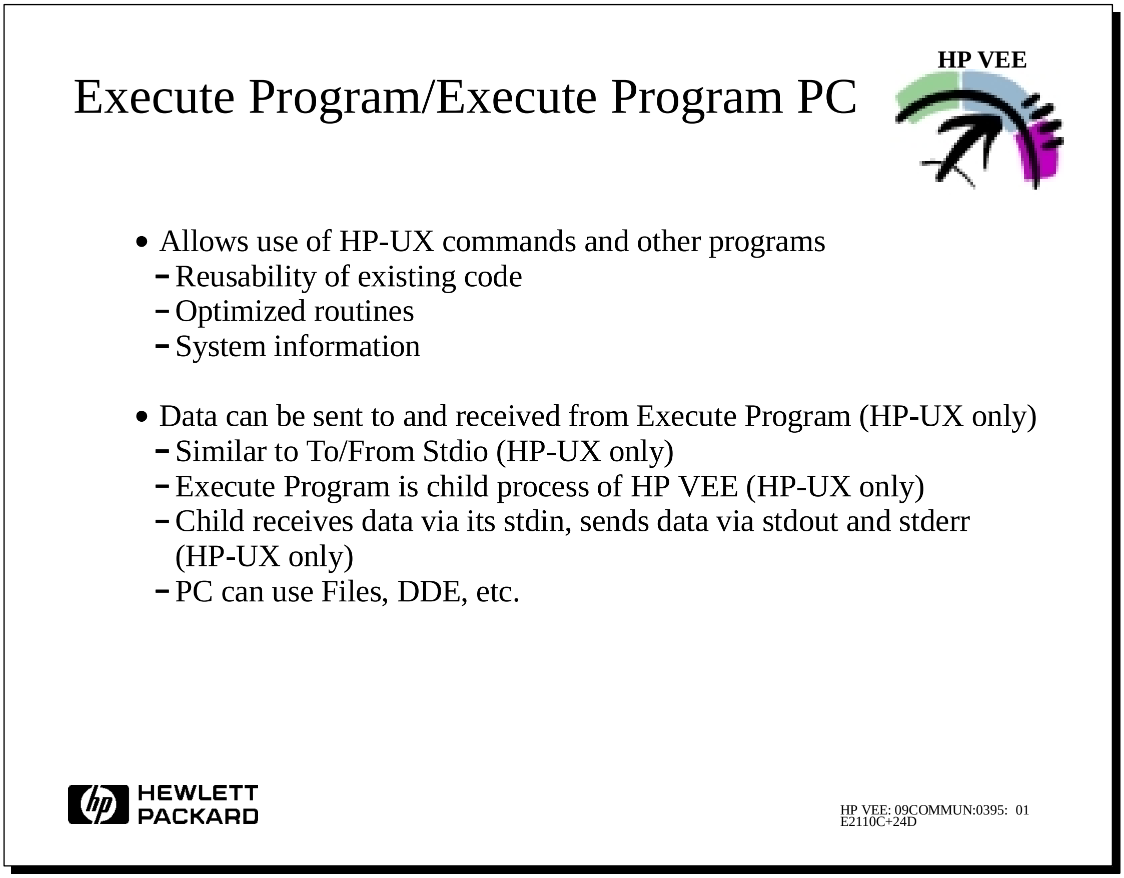
When the Call Object in the local HP VEE is used to call a Remote Function, the data in the Call object is sent over the network to the remote function. After the remote function is executed, the results are sent back to the Call object and output on its data output pins.

If the Delete Library Object is used or the "Delete Lib" menu choice on the Import Object is selected, the remote HP VEE process is terminated and the connection is broken.

The remote HP VEE process has some attributes which are different than a normal HP VEE process. The remote HP VEE process will only access User Functions from the specified file. Any other objects, threads, etc. are ignored. The remote HP VEE process has NO views. This means that there will be no HP VEE icon or HP VEE work area appearing on the screen where the HP VEE program is running; X does not even need to be present. What runs is the pure functionality of the user functions; there is no user interaction. This means that user functions will run faster remotely than they will locally since there are no user events (keystrokes, mouse clicks) to worry about. No views also means that there are no displays, screen dumps, bitmaps, etc.

Communications

Introduction to HP VEE
V3.1 ©1996



HP VEE

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.

HEWLETT PACKARD

HP VEE: 09COMMUN:0395: 01
E2110C+24D

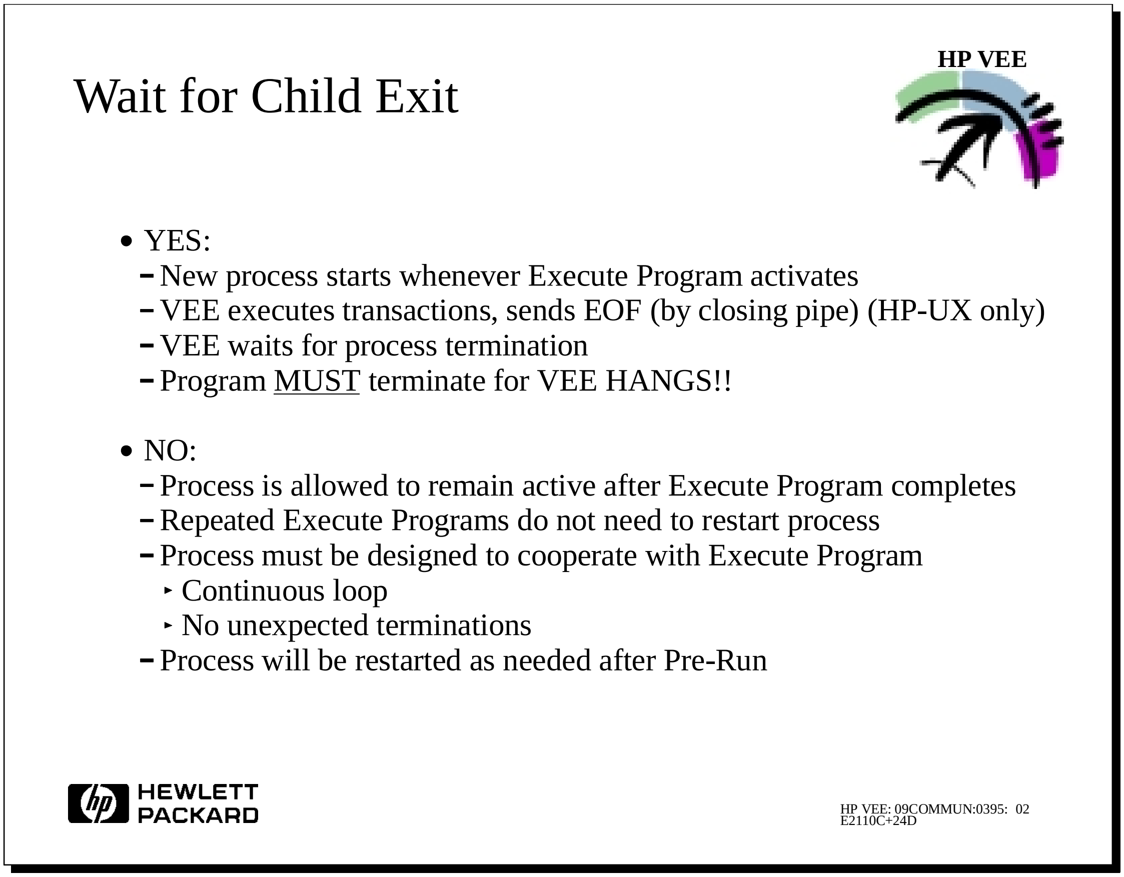
Execute Program gives you the ability to use commands, shells and other programs with HP VEE. With this object you can:

Reuse existing code. Call programs written in C, Pascal or other programming languages

Call libraries of optimized programs, and routines

Obtain system information from the operating system

You can send and receive data from a Execute Program object (HP-UX only). This object uses the transaction box syntax that we saw in other I/O objects previously. Execute Program is similar to the To/From Stdio objects. Execute Program calls the program as a child process of HP VEE (HP-UX only). The program receives data and sends data through its standard I/O channels, stdin, stdout and stderr.



HP VEE

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

HEWLETT PACKARD

HP VEE: 09COMMUN:0395: 02
E2110C+24D

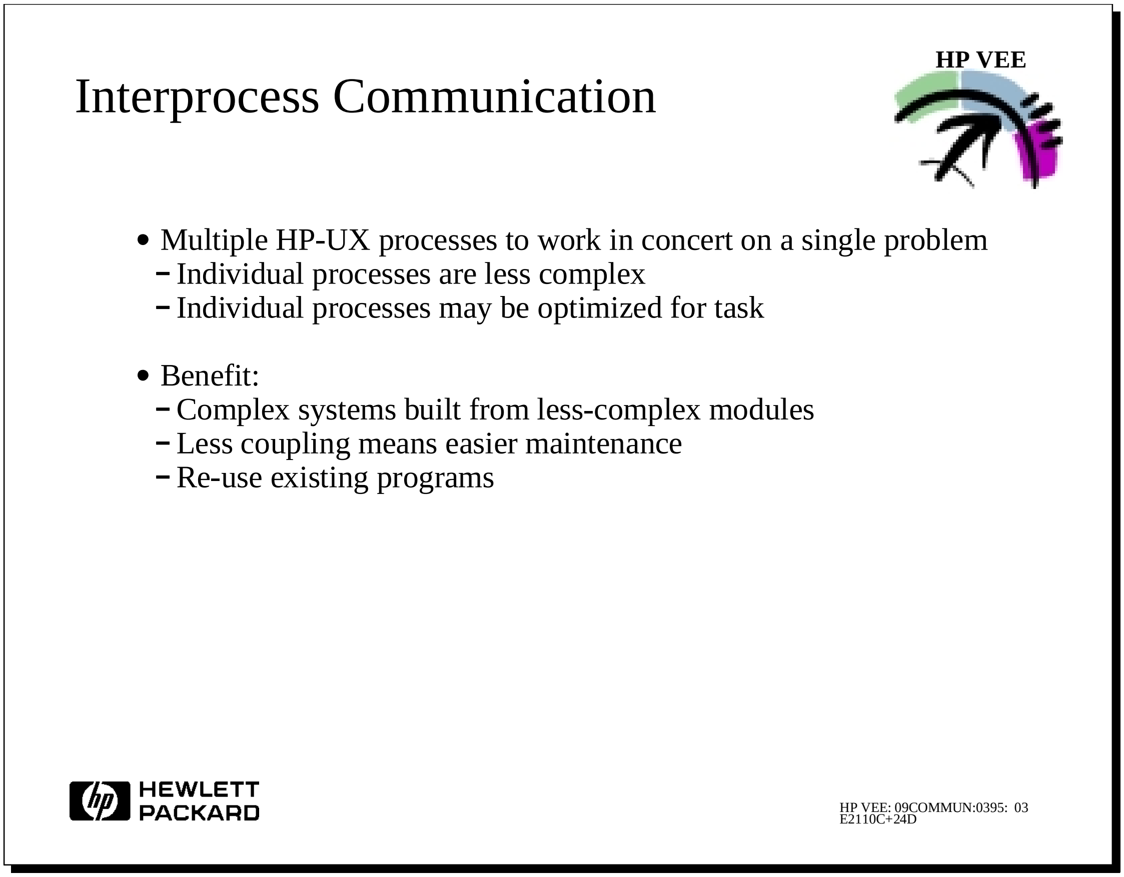
When do you Wait for Child Exit? What does that really mean? When do you want to exit?

Wait for child exit : YES

HP VEE starts a new process each time it activates this Execute Program object. HP VEE executes all of the transactions contained in the Execute Program object. HP VEE then waits for the new process to terminate. The process MUST terminate for the Execute Program object to complete operation.

Wait for child exit: NO

The process remains active after the Execute Program object terminates. Repeated activations of an Execute Program do not need to restart a process. The called process must be designed to cooperate with the Execute Program object. It should be a continuous loop with no unexpected terminations. HP VEE restarts the called process as needed after a PreRun.



HP VEE

Interprocess Communication

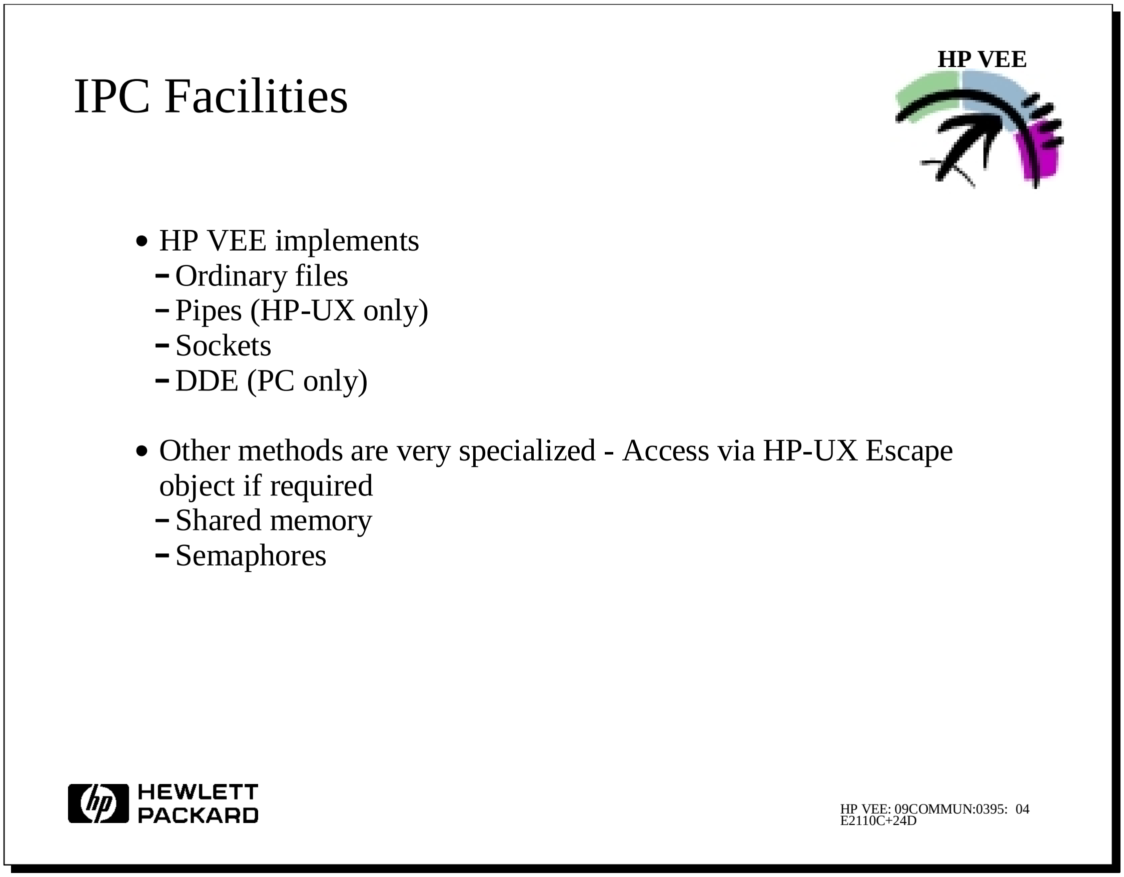
- Multiple HP-UX processes to work in concert on a single problem
  - Individual processes are less complex
  - Individual processes may be optimized for task
- Benefit:
  - Complex systems built from less-complex modules
  - Less coupling means easier maintenance
  - Re-use existing programs

HEWLETT PACKARD

HP VEE: 09COMMUN:0395: 03
E2110C+24D

You will occasionally want several processes to run at the same time, passing information back and forth to each other. In this manner, each process is less complex and may be optimized for an individual task.

By doing so, you build complex systems from less complex modules which are easier to maintain.



IPC Facilities

HP VEE

- 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

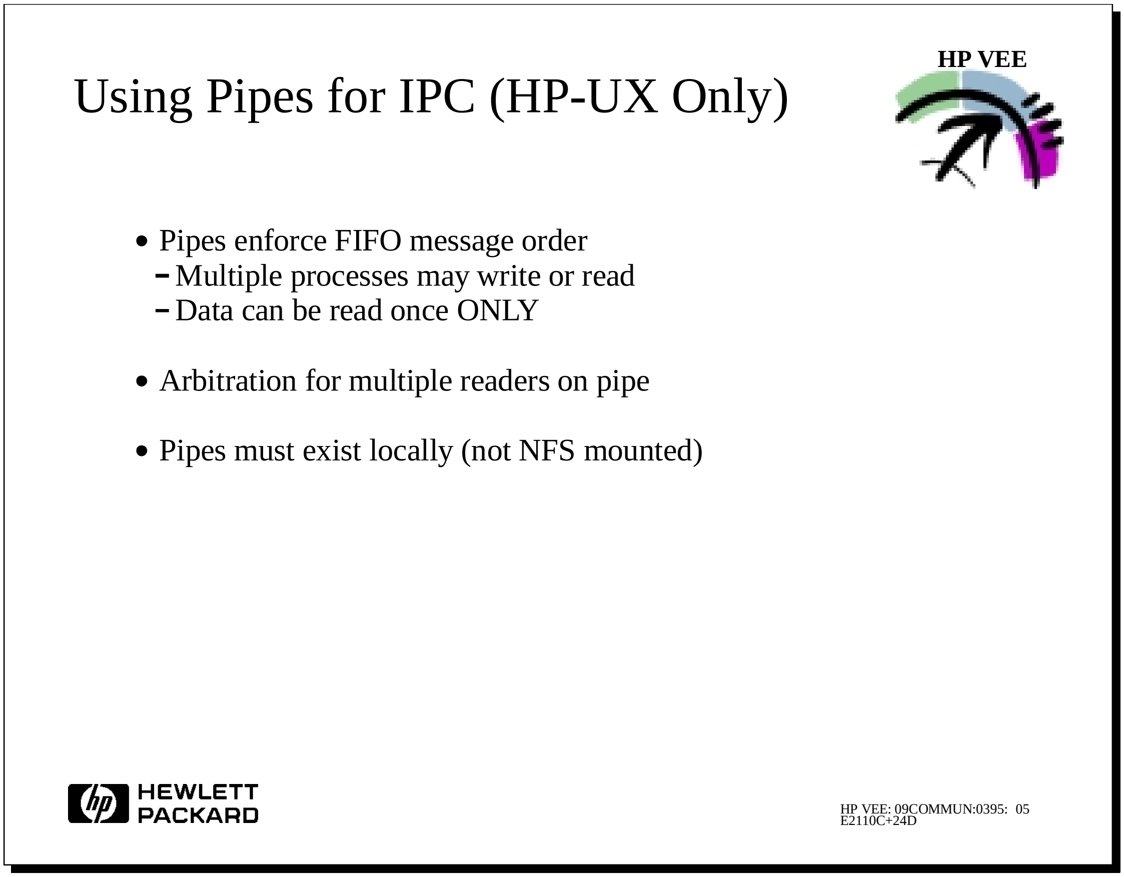
HEWLETT PACKARD

HP VEE: 09COMMUN:0395: 04
E2110C+24D

You may use other methods through the Execute Program object if you find them necessary. These other methods include:

Shared memory

Semaphores



HP VEE

Using Pipes for IPC (HP-UX Only)

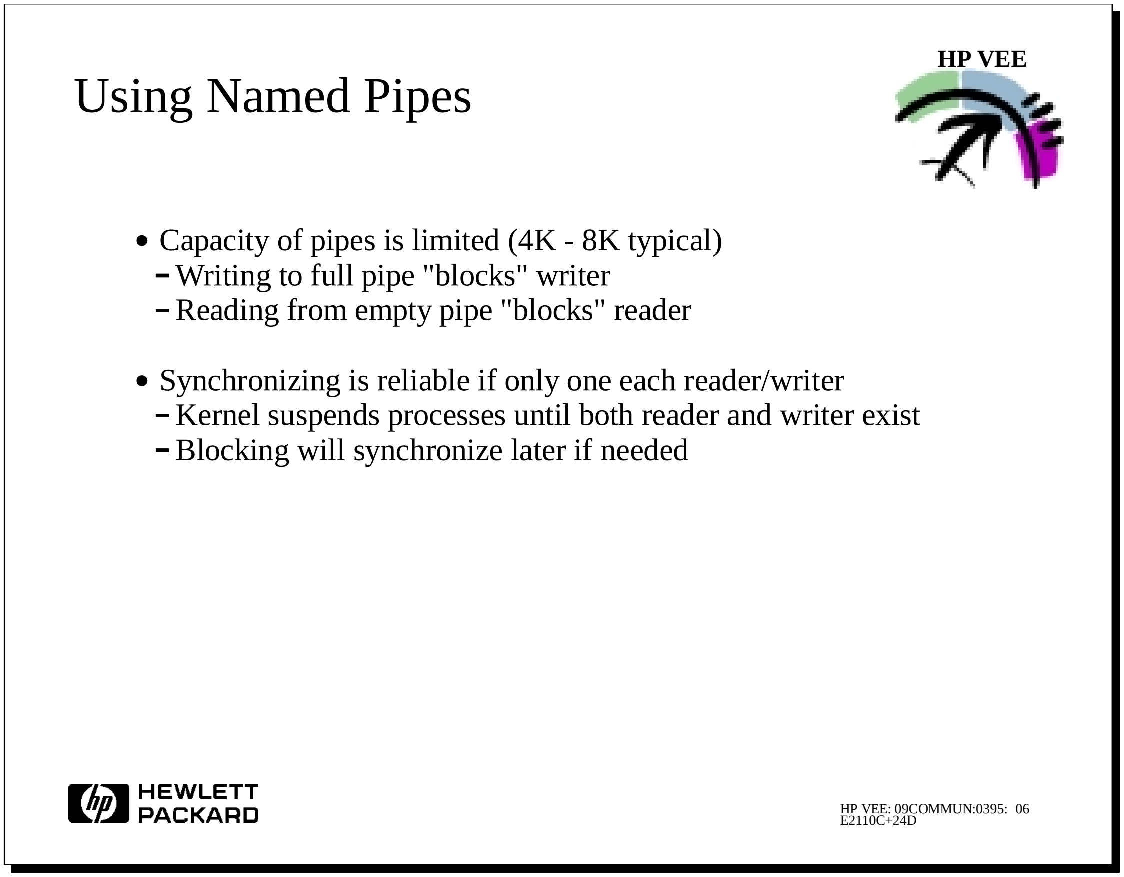
- Pipes enforce FIFO message order
  - Multiple processes may write or read
  - Data can be read once ONLY
- Arbitration for multiple readers on pipe
- Pipes must exist locally (not NFS mounted)

HP VEE: 09COMMUN:0395: 05
E2110C+24D

HP VEE provides access to pipes as an alternate communication link. Pipes enforce FIFO message order. Multiple processes can read from or write to pipes. However, you must remember that you can only read the data once.

VEE's To/From Named Pipe will create the pipes for you.

Note that the pipes used with HP VEE must exist locally. They cannot be NFS mounted.



Using Named Pipes

HP VEE

- 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

HEWLETT PACKARD

HP VEE: 09COMMUN:0395: 06
E2110C+24D

Here are a few things to remember while using pipes:

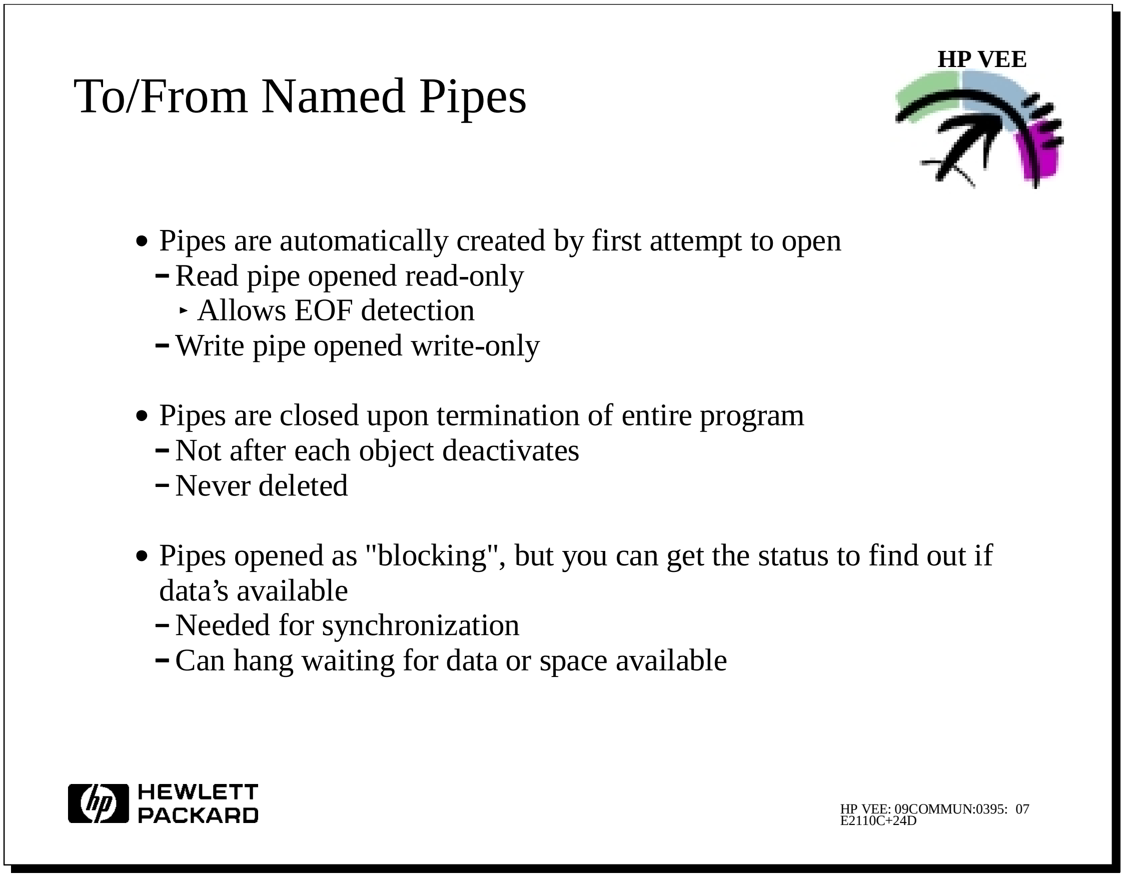
Pipes have a limited capacity, typically 4 to 8 Kbytes

Writing to a full pipe "blocks" the writing processes, causing it to wait until another process reads data from the pipe

Reading from an empty pipe "blocks" the reading processes, causing it to wait until another process writes data to the pipe. Read I/O Status

Synchronizing processes with pipes is only reliable if there is only one reading process and one writing process. The kernel suspends processes until there is both a reader and a writer. The kernel will block one of the processes if needed for synchronization

Pipes must exist locally - they cannot be NFS mounted



HP VEE

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

HEWLETT PACKARD

HP VEE: 09COMMUN:0395: 07
E2110C+24D

HP VEE uses the To/From Named Pipe object to communicate with pipes. It automatically creates the pipe on the first attempt to open the pipe. HP VEE opens a read pipe as read- only, allowing it to detect an EOF. It also opens a write pipe as write-only.

HP VEE closes pipes upon termination of the entire model, rather than at the termination of an object. It never deletes a pipe.

Also note that HP VEE opens pipes as "blocking," for synchronization.

Lab 9

How To Use (and NOT Use) Complied Functions

Objective: This lab is designed to show you what is required from HP VEE to call C functions. This lab does not address how to create the function in C or how to compile the function and build the shared library. Please reference the HP VEE documentation set for more information

Your HP VEE product includes an example C program designed to add the number 1 to a real array that is sent to the program.

Step 1
Import and **Call** the function from the following shared library:

/usr/lib/veetest/examples/concepts/manual49.sl

The header file is stored in:

/usr/lib/veetest/examples/concepts/manual49.h

The function expects two input pins and one output pin.

Use a five element array to send to the array data in pin. Use the new **totSize** object for the array size input pin.

Display the results.

Step 2
Replace the **totSize** object with an integer value of 3. Subtract the array you sent to the **Call Function** object from the array you receive back from the function. Notice that this only adds 1 to the number of elements you specified.

Extra Credit:
WARNING...if you do this exercise, be prepared to kill your VEE process and re-start VEE.

Now, change the integer value on array size to 50. Press **RUN.**

This is telling the C program that although VEE is giving it a five element array, it is OK to use enough memory for 50 elements. This allows the C program to step on VEE's memory.

If this happens, you should always re-start your VEE process!!

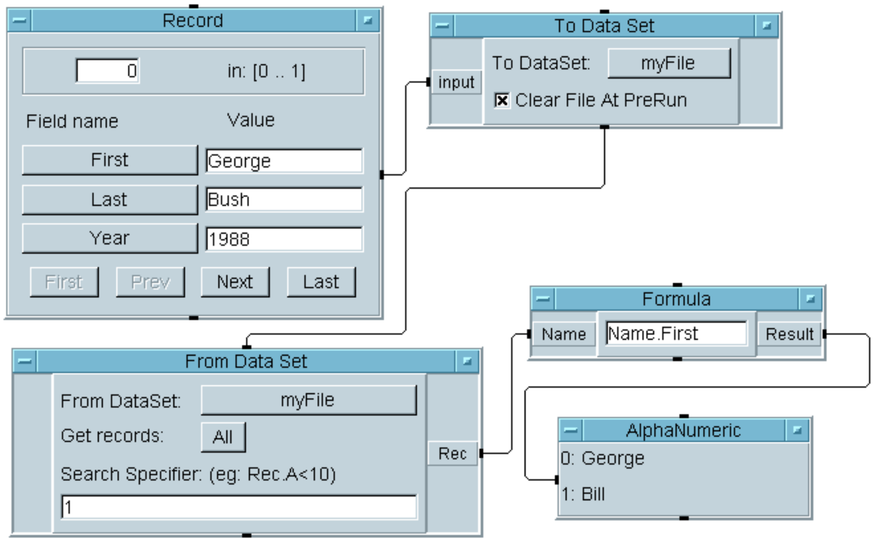
Hence, **totSize** is a good approach.

HEWLETT
PACKARD

Records and DataSets

HP VEE

HP VEE Records and Datasets



Record: 0

First: George

Last: Bush

Year: 1988

To DataSet: myFile

☒ Clear File At PreRun

From DataSet: myFile

Get records: All

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

1

Formula: Name.First

AlphaNumeric:

0: George

1: Bill

HEWLETT PACKARD

HP VEE: 10RECORD:0395: 01
E2110C+24D

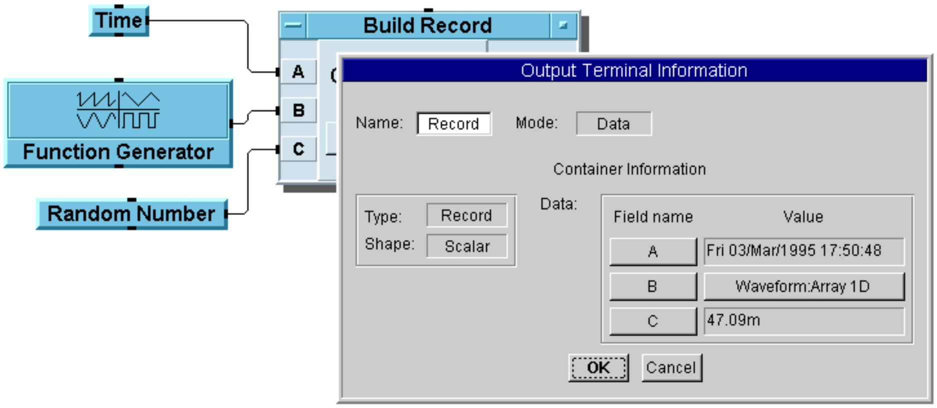
Records
Combining multiple data types in a single container
| text | real | array | waveform

Dataset
An array of records in a file
| text | real | array| waveform
| text | real | array| waveform
| text | real | array| waveform
| text | real | array| waveform
| text | real | array| waveform

HP VEE

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

HEWLETT PACKARD

HP VEE: 10RECORD:0395: 02
E2110C+24D

Records allow you to "package" together, in a single container, many pieces of data which would have previously been represented as individual data lines. There are a number of potential benefits, but probably the most important of these is the ability to reduce the amount of data lines in a program.

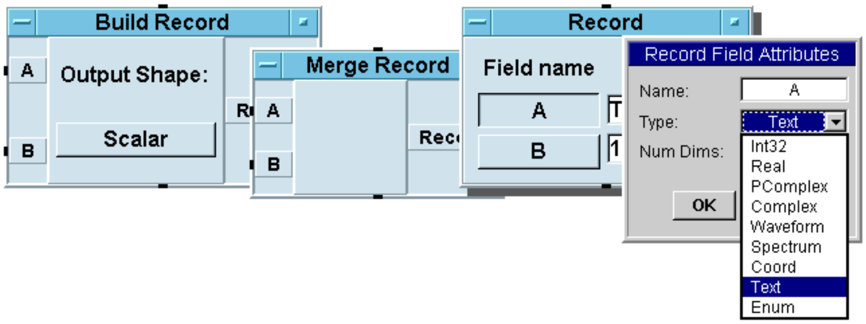
Records can carry many types of data in the same program using only one line between boxes. By combining different types of data in one container, you can have all the information associated with a particular Unit Under Test maintained in one place.

HP VEE

Records

Build 3 Ways

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



Name: A

Type: Text

Num Dims:

HEWLETT PACKARD

HP VEE: 10RECORD:0395: 03
E2110C+24D

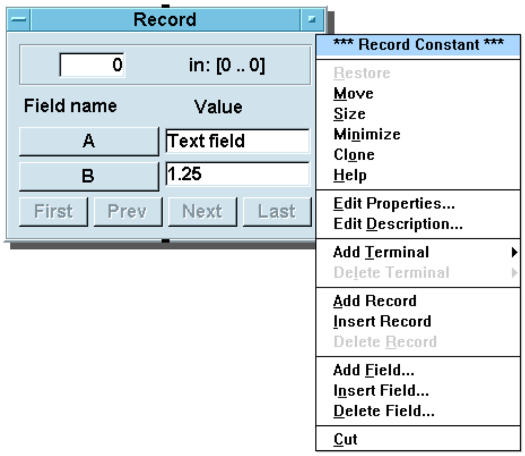
There are three ways to build records. You can use a Build Record object and combine data into a Record. You can enter information into a record constant, or you can merge together two existing records.

The following data types can all be used in record fields.

Int32, Real, Complex, PComplex, Waveform, Spectrum, Coord, Text, Enum, Record

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



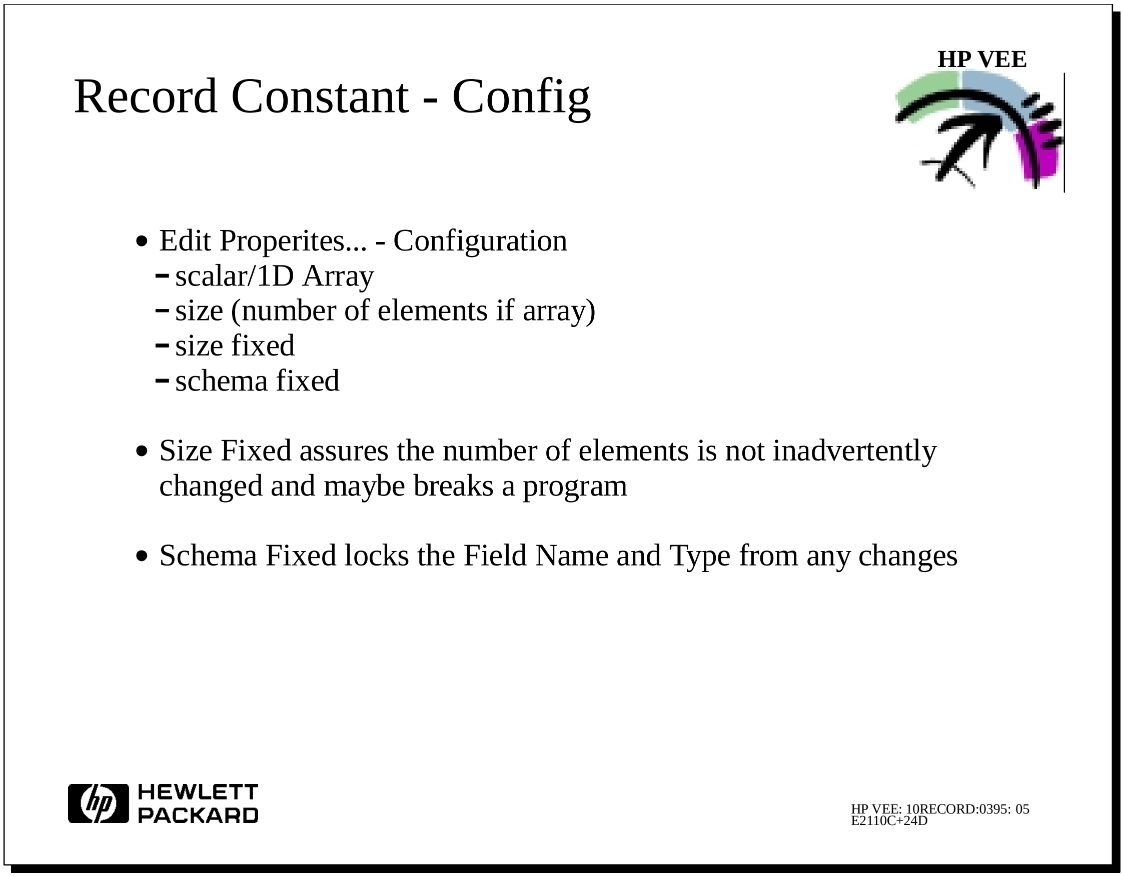
Record

0 in: [0 .. 0]

Field name	Value
A	Text field
B	1.25

Another way to build a record is to start with a record constant. This object is used like other HP VEE constants, except for those specific choices that would only make sense for records. To add additional records to the default, use the object menu choice: ADD RECORD. Notice that you can click on the field name and modify the field name and the type of data in the field.

You can't have a 2D array of Records; only a scalar or a 1D array. However, you can have a Record whose Fields are themselves; Records and nesting is unlimited. Note that Records of Records can only be built with the Data ==> Build Data ==> Record.



HP VEE

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

HEWLETT PACKARD

HP VEE: 10RECORD:0395: 05
E2110C+24D

If you choose ’Config’from Edit Properties... of a Record Constant, you will see three basic choices.

The ’# of elements’ is the same as any other constant. This will allow you to configure this constant to be an array of records, instead of a single record.

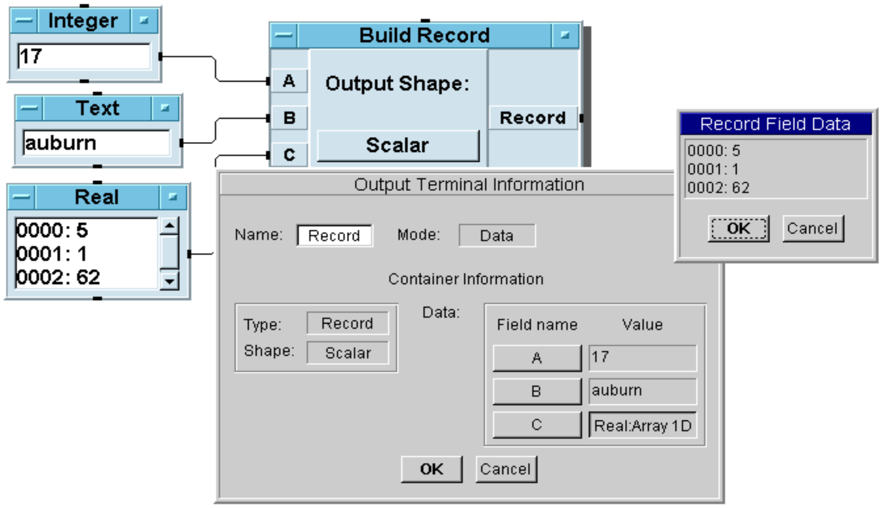
The ’Size Fixed’ choice allows you to lock the number of elements in the array. This choice is also available with the Config menu of other constants.

The ’Schema Fixed’ choice is really important if you are using a Record Constant for operator input. This will prevent someone from editing the field name or type of data required for the field. Additionally, they would not be able to add or delete fields from the record.

HP VEE

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

Output Shape: Scalar

Name: Record Mode: Data

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

HEWLETT PACKARD

HP VEE: 10RECORD:0395: 06
E2110C+24D

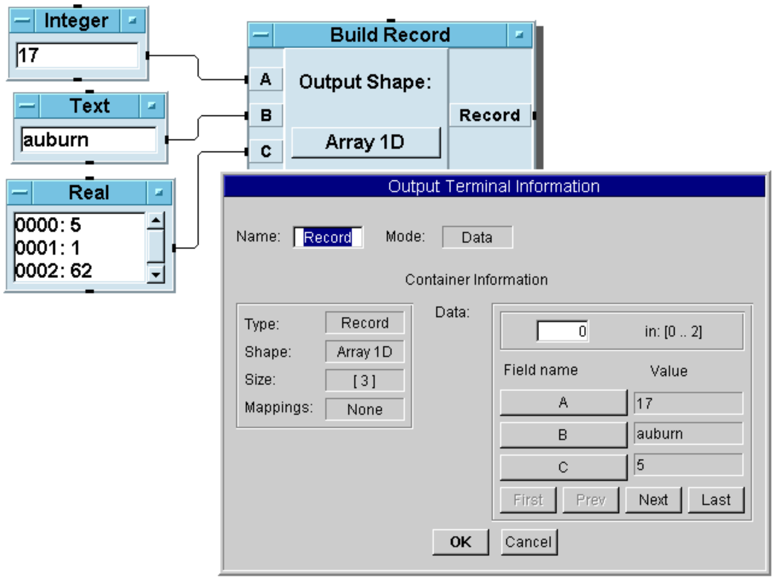
The most common way to build a record is with a Build Record object. This defaults with two input terminals, A and B that can accept any valid data. This default object will yield a record with two fields, one named A, the other named B.

Field names are extremely important. That is how you will later access the information from within any given record. To change the field names of a Build Record object, simply double-click on the terminal and type the new name. You can add fields to the record by adding additional input terminals.

The Build Record object also includes an additional choice: SCALAR vs. ARRAY. The purpose here is to give HP VEE users the flexibility of building a single record with one field equal to an entire array, vs. building an array of records, with the first field equal to each element of the array.

HP VEE

Build Record As Array



Integer: 17

Text: auburn

Real:
0000: 5
0001: 1
0002: 62

Build Record
Output Shape: Array 1D

Output Terminal Information

Name: Record Mode: Data

Container Information

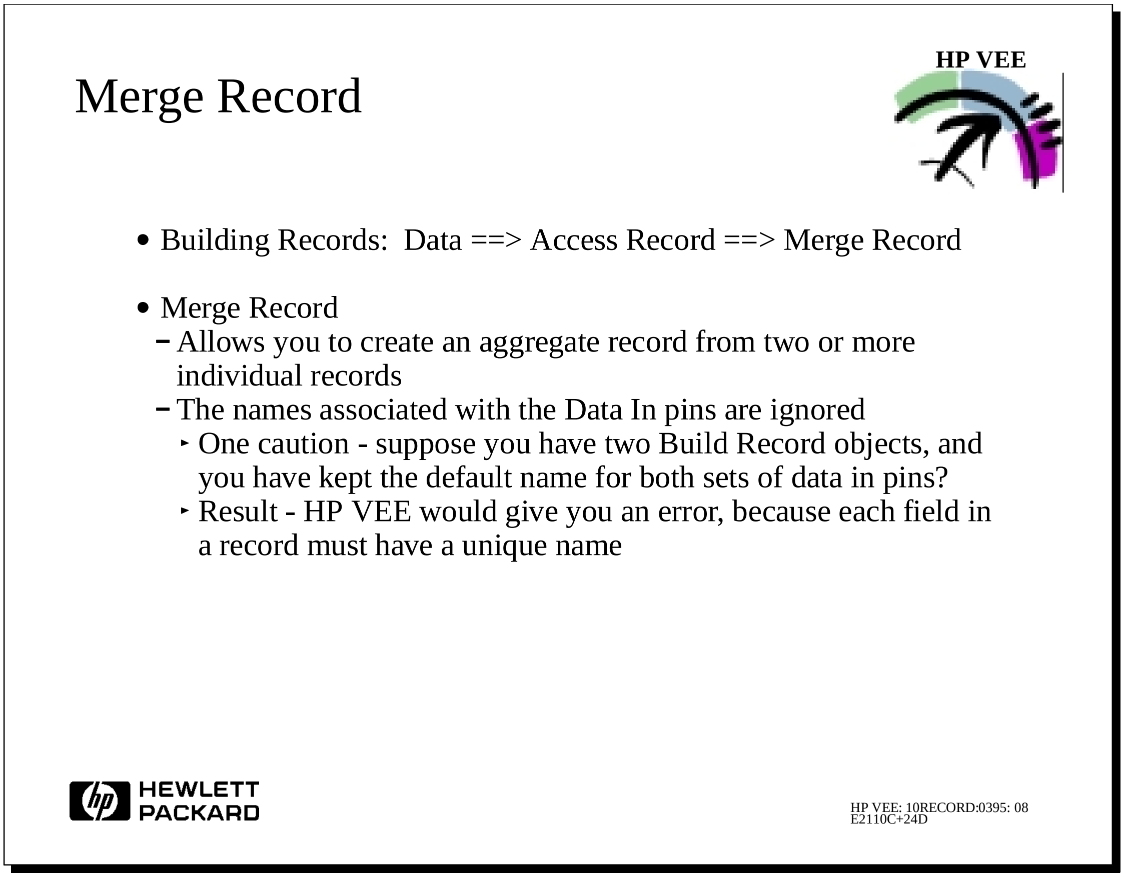
Type: Record
Shape: Array 1D
Size: [3]
Mappings: None

Data: 0 in: [0 .. 2]

Field name	Value
A	17
B	auburn
C	5

HEWLETT PACKARD

In the case of the ARRAY choice, the resulting array of records will have the same number of elements that the original array contained.



HP VEE

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

HEWLETT PACKARD

HP VEE: 10RECORD:0395: 08
E2110C+24D

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

The third way to build records relies on your having at least two other records already defined. The Merge Record object allows you to create an aggregate record from two or more individual records. The names associated with the data in pins on the Merge Record object are ignored.

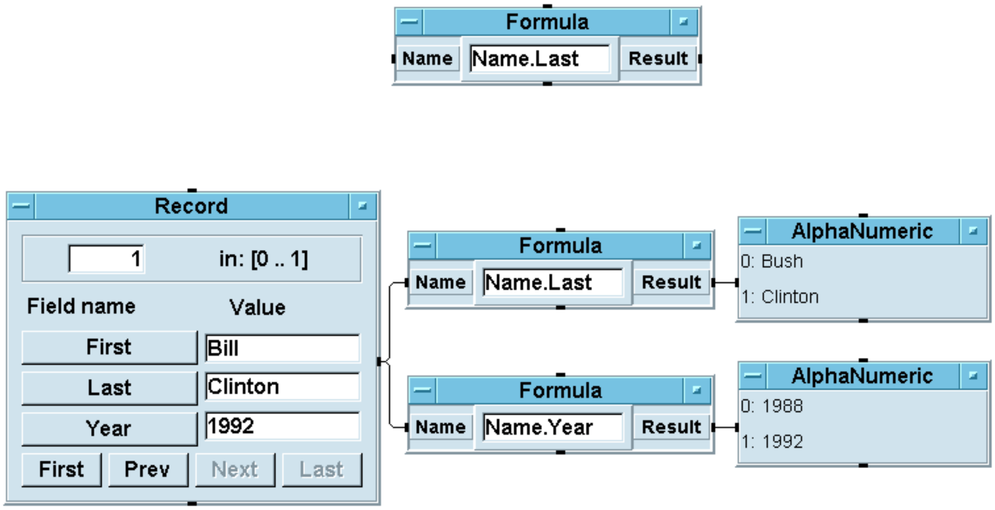
One caution: Suppose you have two Build Record objects, and you have kept the default names for both sets of data in pins?

Result: HP VEE would give you an error, because each field in a record must have a unique name.

HP VEE

Records in Expressions

- Instead of UnBuilding Data - use formulas!



Formula: Name.Last

Record: 1 in: [0 .. 1]

Field name	Value
First	Bill
Last	Clinton
Year	1992

Formula: Name.Last → AlphaNumeric:

```
0: Bush
1: Clinton
```

Formula: Name.Year → AlphaNumeric:

```
0: 1988
1: 1992
```

HEWLETT PACKARD

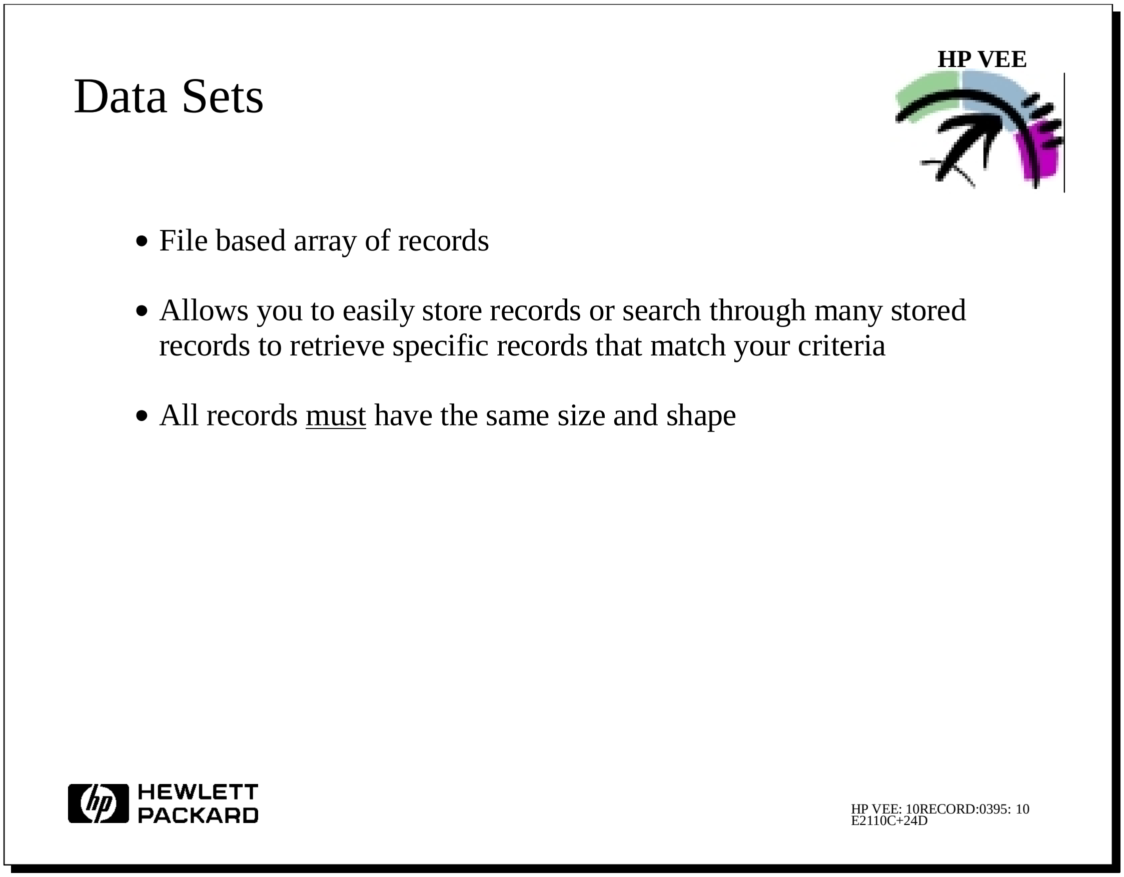
HP VEE: 10RECORD:0395: 09
E2110C+24D

Record fields can be used in formulas without the use of the UnBuild Data object. This gives us the capability to directly access the data contained within the field. This is an extremely powerful way to access information from a record.

The syntax to access the field is simply TERMINALNAME.FIELDNAME.

Notice that the name of the Record in a formula is represented by the name of the input terminal (or the name of a global). The field name is the name assigned to that field when the record was created. You cannot use a variable for the fieldname.

Note that the same data types and shapes you could normally apply parallel math to will also work with the fields within a record.



HP VEE

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

HEWLETT PACKARD

HP VEE: 10RECORD:0395: 10
E2110C+24D

A Data Set is simply an array of records, stored to a file.

The primary rule of DataSets (or any array of records) is that all fields of the records must have the same type and shape. So, if the first record in the array has a field with a 25-element array, all records in the array must have the same field with a 25-element array.

To create a DataSet, choose the I/O ==> To ==> DataSet object from the menu. This object is very similar to a TO FILE object, except that it only accepts record inputs. The file that is created is a standard ASCII file with a Schema header that defines the fields and their attributes.

To read data from a Data Set, choose the I/O ==> From ==> DataSet object. This object has three basic fields, the name of the file, a toggle button for ONE vs. ALL , and a criteria field.

The toggle field either extracts the first record from the dataset to meet your criteria, or it will give you all records from the dataset that meet the criteria.

The bottom field allows the user to specify which records to get from the dataset. This is a full expression box, so it has all the same capabilities as a formula object, and allows field comparisons against values and Boolean expressions.

It is important to note that the name you specify for record field attributes in the expression field must match the name you assigned that particular record field when you created the original record.

Lab 10

Manipulating Records

Objective: Learn how to use Records and Datasets.

Step 1
Build a **Record** with the following:
The first field should accept an integer. The second field should be named **time** and include the time right **NOW().** The third field should be a four element array.

Hint: A **Real Constant** can be configured to have multi-elements (like 4).

Merge with that record another record that has a random number between 0 and 1, and a waveform (**Virtual Source**). The resultant record should now have 5 fields.

Step 2
Use a **Triadic Operator** in a formula box to test the random number in the record and display either the integer from the record or a text string.

If the value is less than .5, then display the first field of the record (the integer).

If the value is greater than .5, send the text string "More than 0.5".

Next, extract the time and the waveform. (Hint: do not use a **Formula** object or a **Get Field** object). Display **This Record** on an **AlphaNumeric** object.

Step 3
Replace the integer with a **For Count** object and step it through 10 iterations. (Be certain to "ping" the random number generator and the time function on each iteration).

Send the complete record into a **To Dataset** object.

In a separate thread, get all records from the dataset where the random value is greater than 0.5. Put the resultant records into a record constant.

Hint: **Record Constant Add Terminal=> Control Input=>Default Value.**

Step 4
Create a panel display that allows you to edit the information from the Dataset.

Include a button that sends your changes back to the **Dataset.**

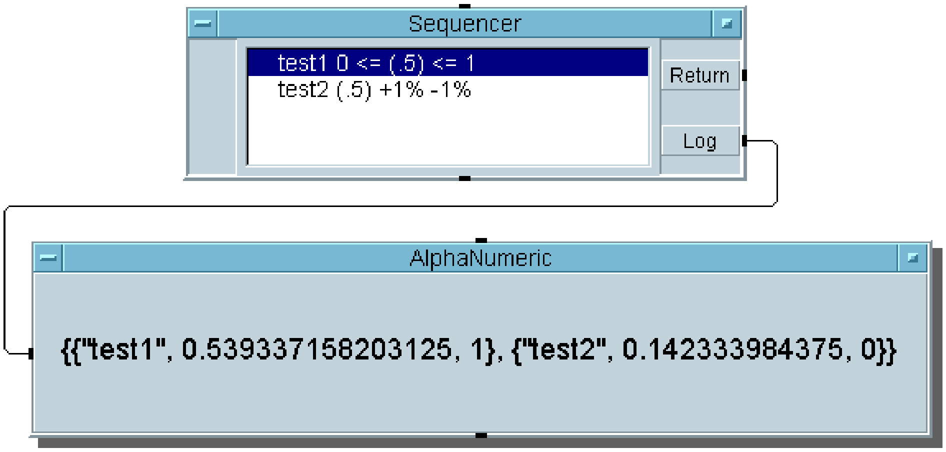
Run this program several times to see that your changes are being stored into the Dataset.

Sequencer

Introduction to HP VEE
V3.1 ©1996

HP VEE Sequencer

HP VEE



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

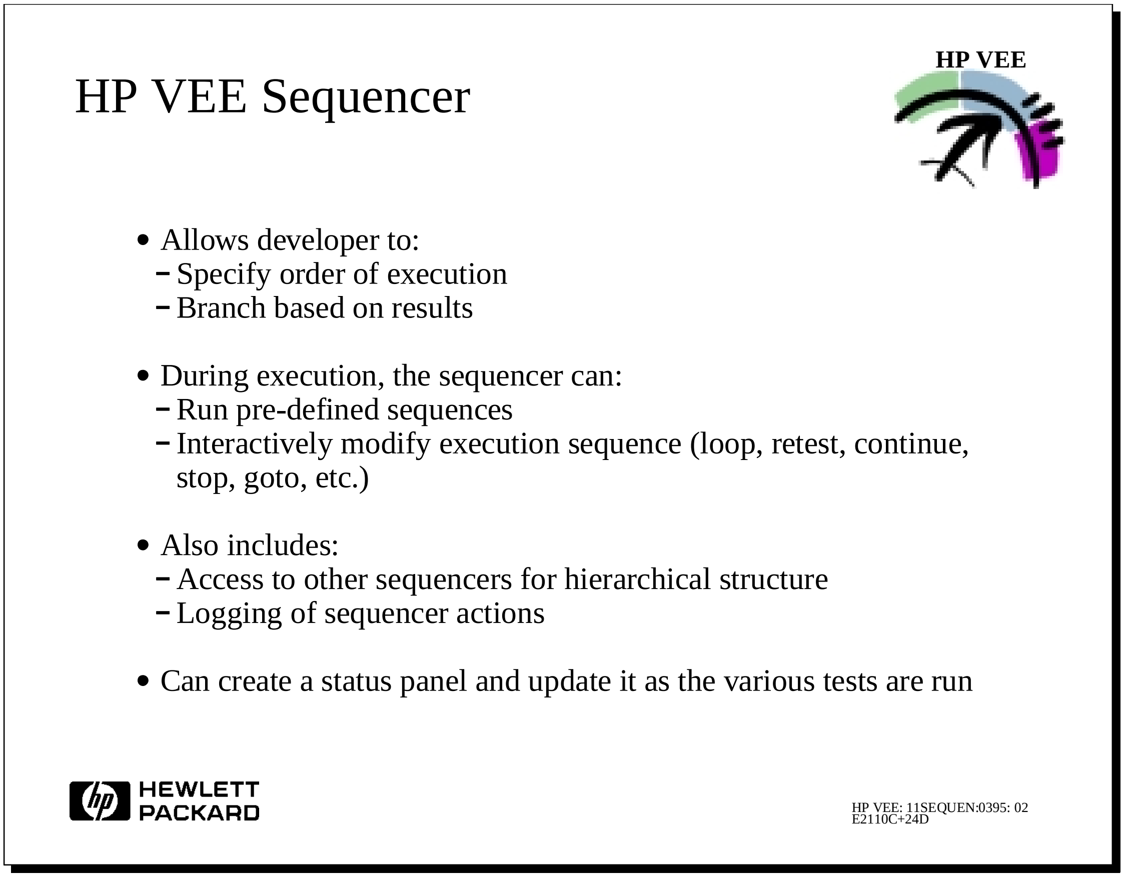
```
{{"test1", 0.539337158203125, 1}, {"test2", 0.142333984375, 0}}
```

HEWLETT PACKARD

HP VEE: 11SEQUEN:0395: 01
E2110C+24D

The purpose of the sequencer is to add a ’Control Flow’ capability to this ’Data Flow’ language.

The sequencer is the fundamental building block to create a ’Test Executive’. As a programmer builds their modular tests, the sequencer can be used to execute the tests given the configuration of the transactions.



HP VEE

HP VEE Sequencer

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

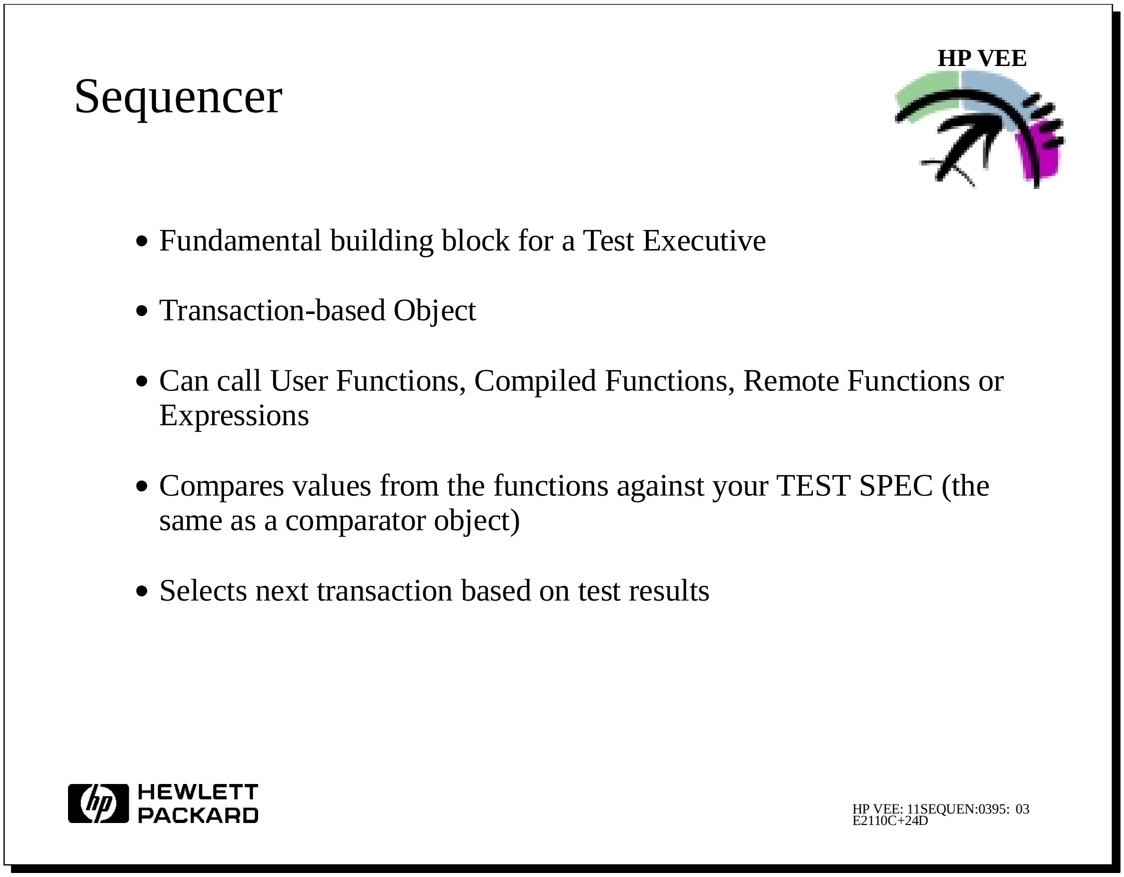
- During execution, the sequencer can:
  - Run pre-defined sequences
  - Interactively modify execution sequence (loop, retest, continue, stop, goto, etc.)

- 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

HEWLETT PACKARD

HP VEE: 11SEQUEN:0395: 02
E2110C+24D



Sequencer

HP VEE

- 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

HEWLETT PACKARD

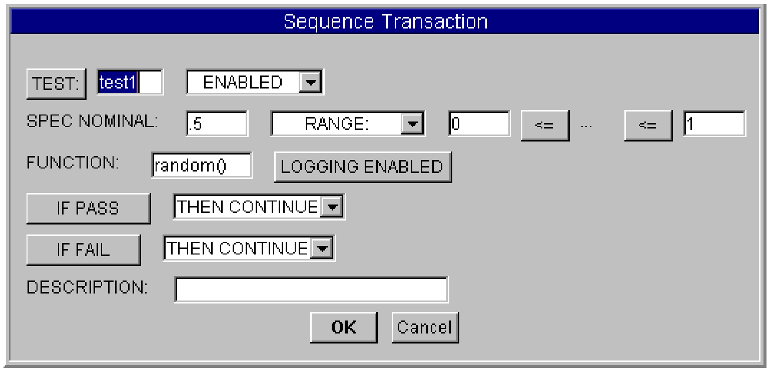
HP VEE: 11SEQUEN:0395: 03
E2110C+24D

The Sequencer contains a list of tests to be called. Each test called is a Sequence Transaction, which specifies how and when to call a specified test routine. Each Sequence Transaction evaluates a HP VEE expression which may contain a call to a User Function, Remote Function, or Compiled Function. The sequencer will get a return value from the test and then compare the value returned by that expression against a test specification. Depending on whether the test passes or fails, the sequencer will then select the next transaction to be executed.

Test Transactions may optionally log their results to the Log output pin, or to a User Function specified in the Logging Config dialog box.

ShortCuts: See Help on Sequencer

HP VEE


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.

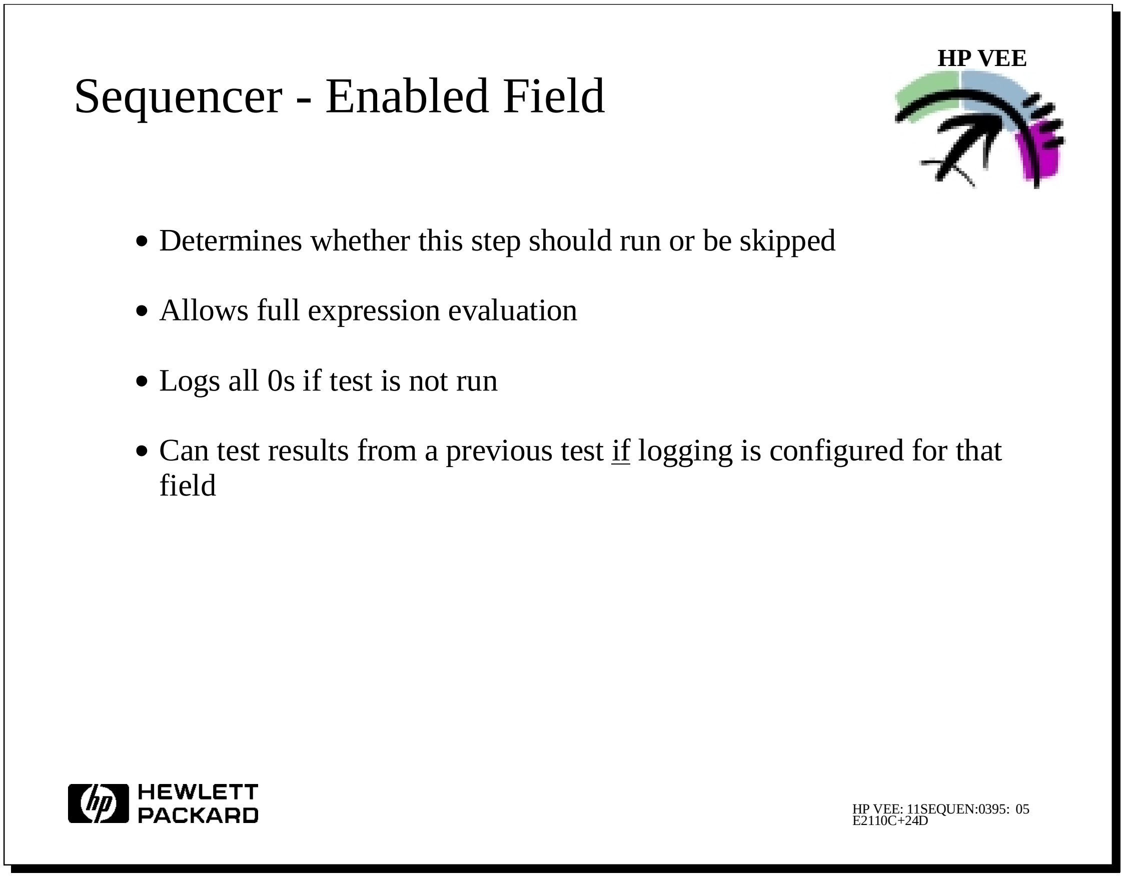
Sequencer Fields

Test Name:
Is simply a "handle." It allows you to associate a test name with each sequencer transaction. All test names in a sequencer must be unique. The Sequencer will default the test name to "test1, and increments for new tests.

Test/Execute Toggle:
This toggle box allows you to specify whether the named test should only be executed (ignoring the test limits and return values) or should actually perform the comparisons to the specifications you provide. In "EXEC" option, the Sequencer transaction will remove the references to test specifications and pass/fail operations. Also, selecting EXEC will disable all logging for these tests.

Function:
The function field specifies which test to run. This line can call User Functions, Compiled Functions, Remote Functions, or an expression. It is the result of this function call that will be tested against the specifications. The return value from a UserFunction comes from the top output pin on the function.

Description Line:
A text entry field where you can add descriptive comments to your Sequencer. These comments will show on the sequencer object and can be logged with the data.



HP VEE

Sequencer - Enabled Field

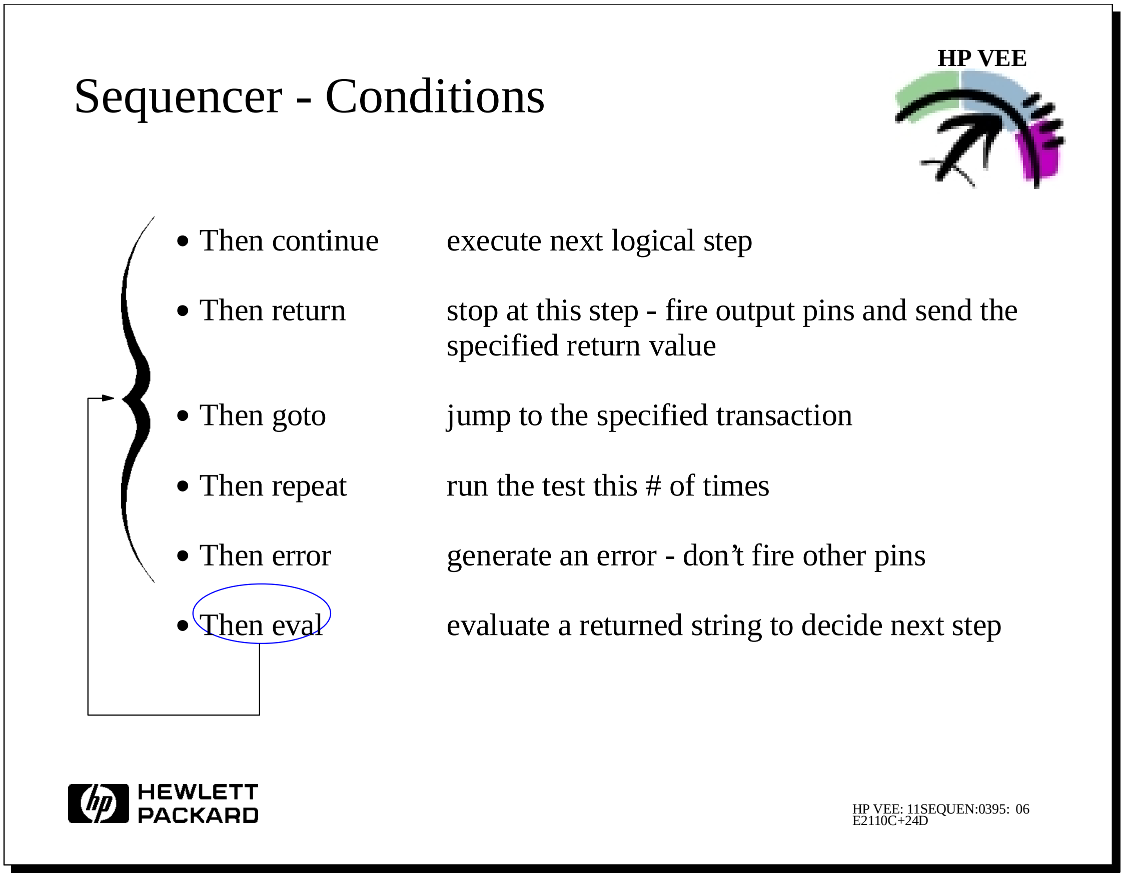
- Determines whether this step should run or be skipped
- Allows full expression evaluation
- Logs all 0s if test is not run
- Can test results from a previous test if logging is configured for that field

HEWLETT PACKARD

HP VEE: 11SEQUEN:0395: 05
E2110C+24D

An ENABLED button appears on the top line of the sequence dialog box. This allows you to specify running this test step only if certain conditions exist. There are four possible selections: ENABLED, DISABLED, ENABLED IF, and DISABLED IF. The choices to ENABLE and DISABLE allow you to unconditionally mark this test to always execute or never execute.

The choices ENABLED IF and DISABLED IF both evaluate an expression to determine whether this test gets executed. This field can accept any globals, calls to functions, etc., the same as in a formula box.



Sequencer - Conditions

HP VEE

• Then continue	execute next logical step
• Then return	stop at this step - fire output pins and send the 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	evaluate a returned string to decide next step

HEWLETT PACKARD

HP VEE: 11SEQUEN:0395: 06
E2110C+24D

Conditions:

Then CONTINUE - Go to the next transaction in the Sequencer list.

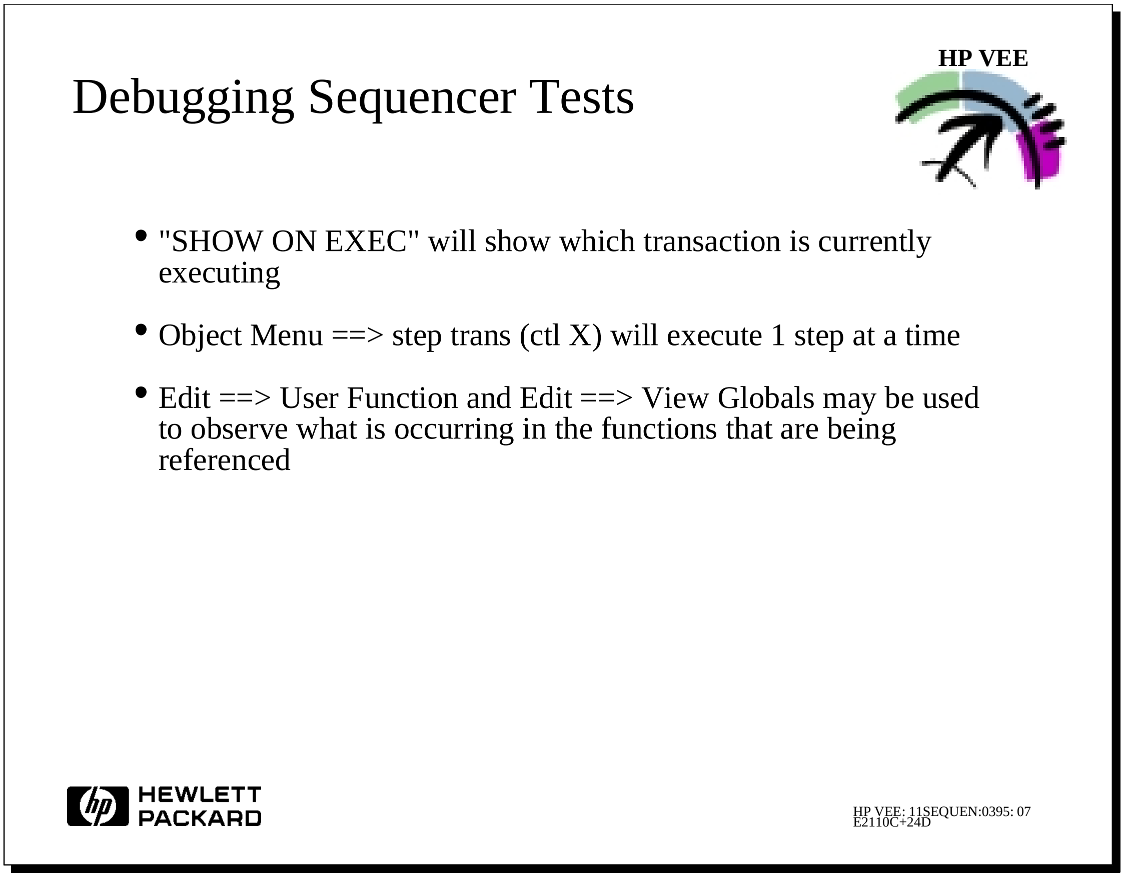
Then RETURN - Quit executing transactions in this Sequencer and place the value of the specified expression on the "Return" output pin of the Sequencer. This is a clean way to exit the sequencer at this point. A more direct and harsh method of ending the sequence is to generate an Error (see below).

Then GOTO - Go to the transaction in this Sequencer with the specified transaction name.

Then REPEAT - Execute this transaction again, repeating up to the specified number of times. If the Pass/Fail condition still exists after the maximum number of repeats, continue to the next transaction.

Then ERROR - Stop execution of the Sequencer by generating an error condition with the given error number. An error can be trapped with an Error output pin on this Sequencer. No other output pins will send data.

Then EVALUATE - Call out to another User Function and expect a return string that looks like the the other dialog choices. Valid string results from the expression are: "Continue", "Return <expr>", "Goto <name>", "Repeat <expr>", "Error <expr>", where <expr> is any valid HP VEE expression, and <name> is the name of a transaction in this sequence. This feature allows test developers to prompt operators for which action the sequencer should take next (Quit, Test Again, Start Over, etc).



Debugging Sequencer Tests

HP VEE

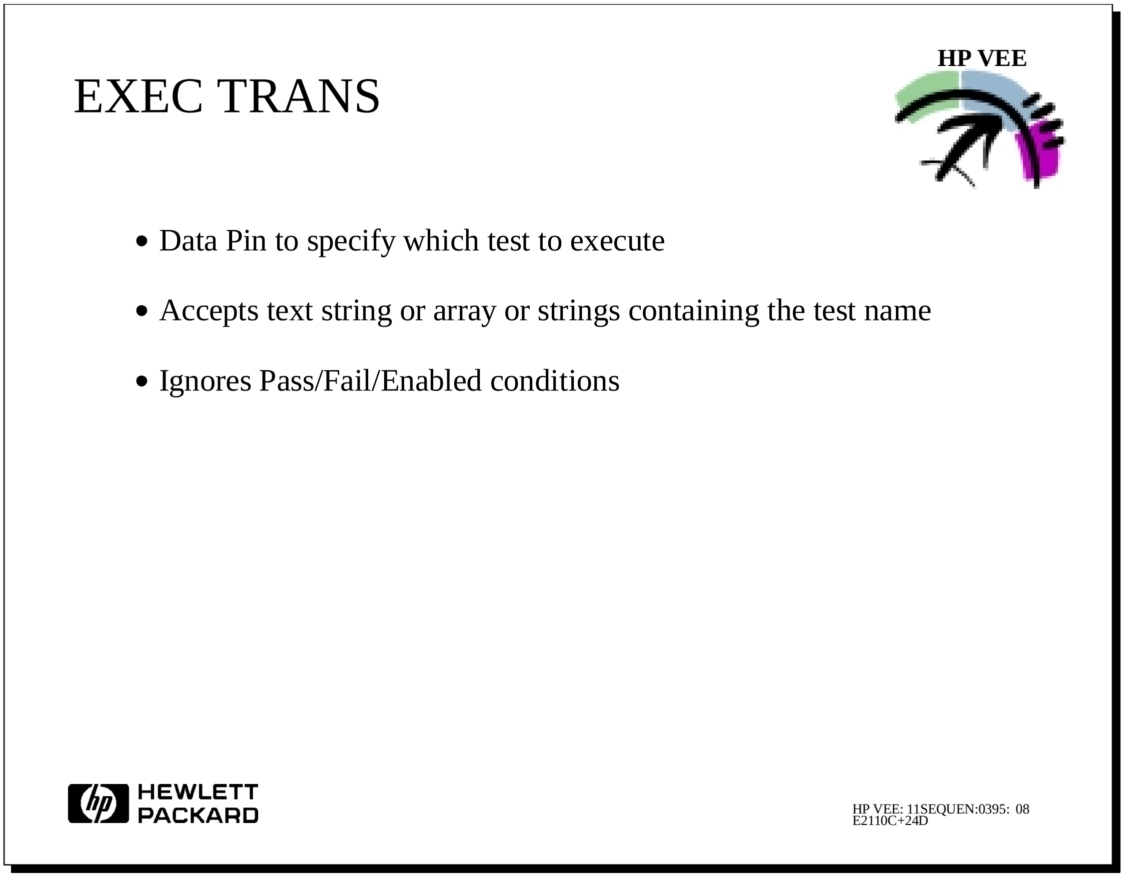
- "SHOW ON EXEC" will show which transaction is currently executing
- Object Menu ==> step trans (ctl X) will execute 1 step at a time
- Edit ==> User Function and Edit ==> View Globals may be used to observe what is occurring in the functions that are being referenced

HEWLETT PACKARD

HP VEE: 11SEQUEN:0395: 07
E2110C+24D

To debug the sequence of tests as you are developing your program, there are several features of HP VEE that will come in handy. The Edit->Animate choice will let you see which transaction is executing. The Step Key at the top of your HP VEE environment will not allow you to step from one test to the next, because you are in one object. To allow you to execute the sequencer one test at a time, HP VEE has a choice on the sequencer's object menu called STEP TRANS. There is also a short-cut key to STEP TRANS-Cntrl X.

HP VEE allows you to keep several Edit UserFunction windows open at a time allowing you to see several tests as they are being called from the sequencer.



HP VEE

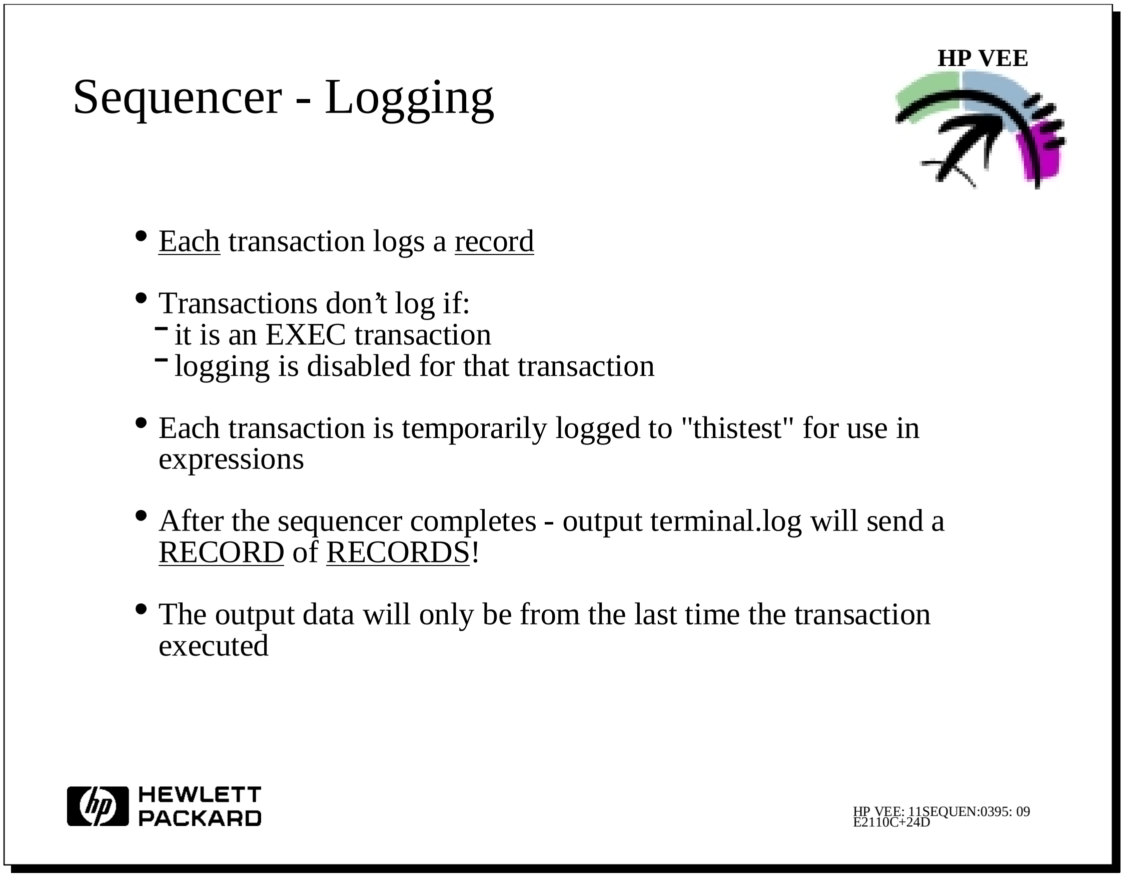
EXEC TRANS

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

HP VEE: 11SEQUEN:0395: 08
E2110C+24D

You may not always want the sequencer object to run all tests listed. For additional control over specific tests to run, you can add an ’Exec Trans’ data pin to the sequencer object. This pin causes the Sequencer to execute only the transaction(s) with the specified name(s) in the order given.

It requires a Text scalar or ID array on the control input. The highlight bar will be moved to the transaction being currently executed. The IF PASS or IF FAIL expressions will be evaluated as usual, but the result of the THEN field will have no effect on the next transaction to be executed.



HP VEE

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

HEWLETT PACKARD

HP VEE: 11SEQUEN:0395: 09
E2110C+24D

Every sequencer test transaction logs a record of information about that particular test step. The default record consists of the name of the test, the return value, and whether the test passed or failed. While each test is running, that record is stored in a temporary record called "thistest". After the test step is complete, the record can be accessed by calling the testname.field (test1.result). This way, a later test can use data from previous tests for enabled conditions, test limits, etc. (ENABLED IF: Test1.Pass = 1).

After the sequencer successfully completes, all of the records from the test steps will be logged from the LOG data output terminal of the sequencer.

If your test step is not logging the information, it could be because:

1. Logging can be disabled for the test. This will not log any record of the test.

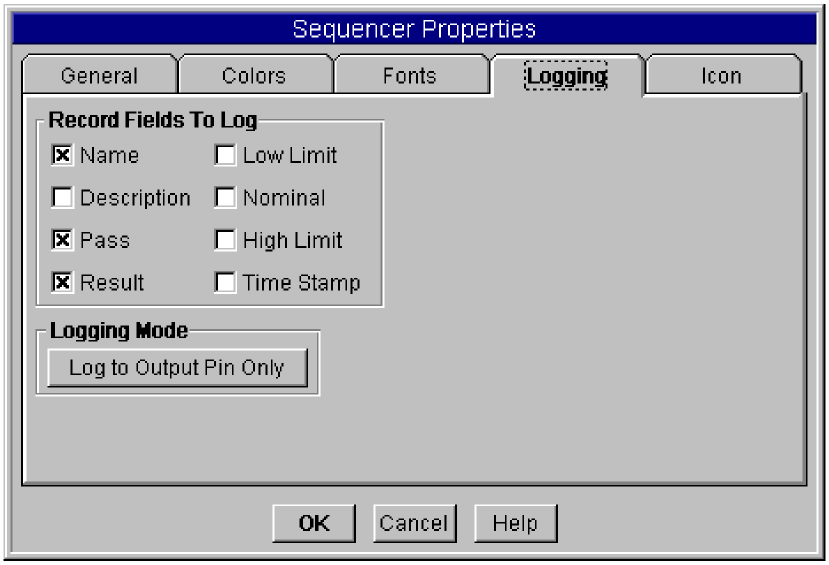
2. Your test may be an EXEC test that does not compare results. EXEC tests do not log information.

3. If you have disabled the test, the sequencer will log all 0's.

4. If your test choice is to ERROR, the LOG output terminal will not send any information.

HP VEE

Sequencer - Logging Configuration



Sequencer Properties

Record Fields To Log

☒ Name ☐ Low Limit

☐ Description ☐ Nominal

☒ Pass ☐ High Limit

☒ Result ☐ Time Stamp

Logging Mode

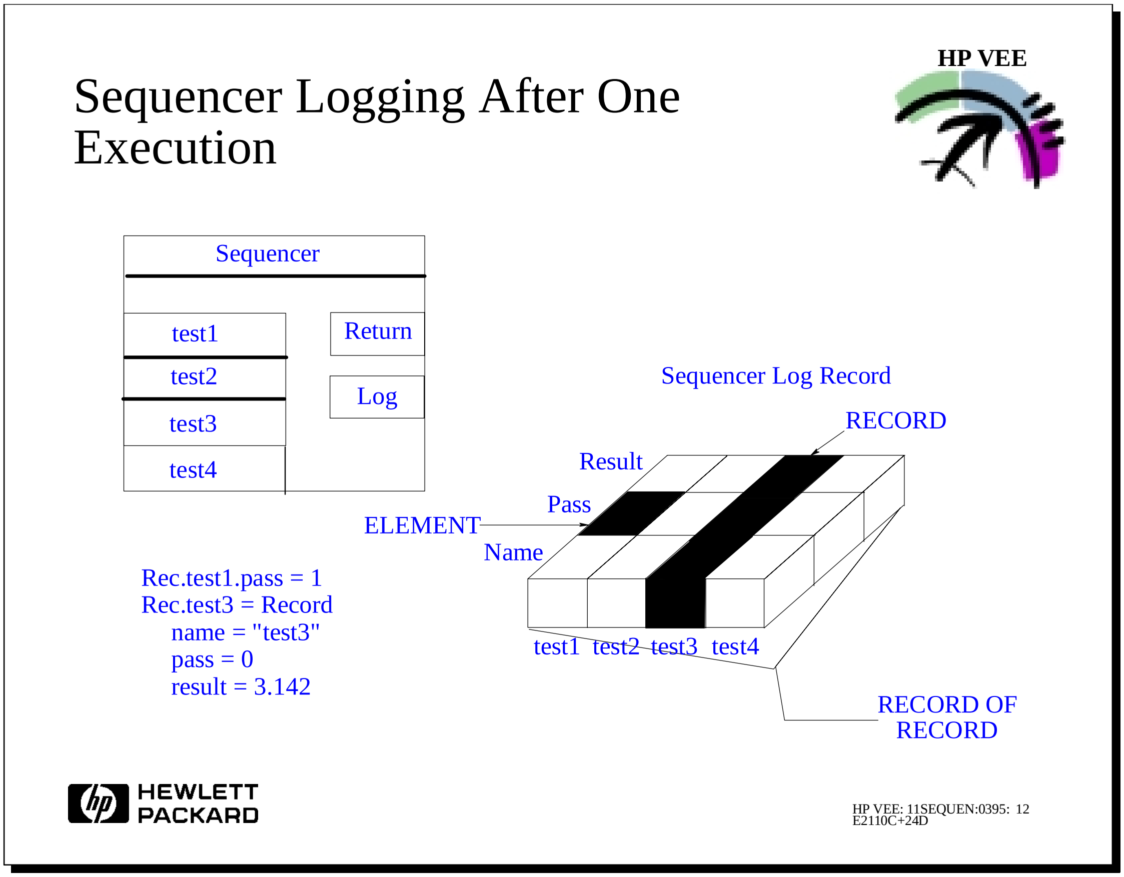
Log to Output Pin Only

HEWLETT PACKARD

HP VEE: 11SEQUEN:0395: 10
E2110C+24D

The records from each test will only include test name, test result, and pass status by default. To add additional fields to log, there is a choice on the Sequencer’s Edit Properties... labeled LOGGING CONFIG.

The selection box allows you to decide which pieces of information you wish to log for all tests in the sequencer. At the bottom of this dialog box is a choice to LOG TO OUTPUT PIN ONLY. This is a toggle box that allows you to log specific information to another function when each test is run. This is useful because the standard log output pin will only show the data from the last time each test was called. If your sequencer calls the same test step several times, you will not get a data on previous calls unless you use this config option.



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

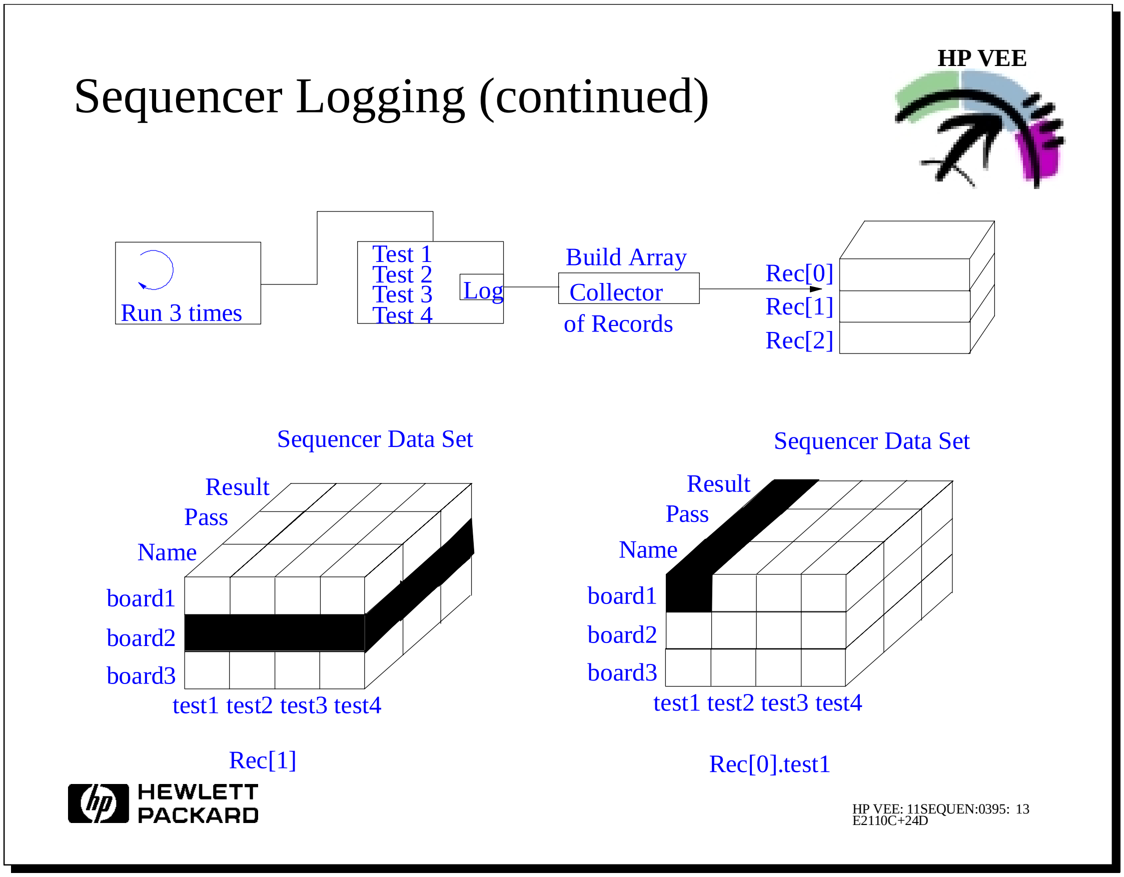
HEWLETT PACKARD

HP VEE: 11SEQUEN:0395: 12
E2110C+24D

This slide shows an example of one run of the sequencer. The Log output terminal would send a single record (REC) that contained four other records: test1, test2, test3, and test4. Each of the test records would contain three fields: name, pass, and result.

The reference to Rec.test1.pass will answer with a 1 if the test1 transaction test passed.

The reference to Rec.test3 would actually reference a record with the following fields: the name, ’test3’, whether it passed, ’1 or 0’ and the result that test3 returned to the sequencer when it was called.



Sequencer Logging (continued)

HP VEE

Run 3 times

Test 1
Test 2
Test 3
Test 4

Log

Build Array
Collector
of Records

Rec[0]
Rec[1]
Rec[2]

Sequencer Data Set

Result
Pass
Name
board1
board2
board3
test1 test2 test3 test4

Rec[1]

Sequencer Data Set

Result
Pass
Name
board1
board2
board3
test1 test2 test3 test4

Rec[0].test1

HEWLETT
PACKARD

HP VEE: 11SEQUEN:0395: 13
E2110C+24D

Here is another illustration of sequence logging based on the previous example. This time, however, after the test has been run three times on different units, the results are gathered into an array of records.

To Rec[1] is the reference to the second element of the array, and would return a single record with subrecords for test 1 through test 4 and the fields for each record.
The reference to Rec[0].test1 would return a record with three fields, the result of the first run, the first test, all fields.

Lab 11a

Using the Test Sequencer

Objective: Become familiar with the capabilities of the Test Sequencer

Step 1
Create a simple user function called **UpperLimit** that is a pop-up panel with a **Real Slider** and an **OK** confirm object. Send the output of the slider to a **Global** called **UL** and also to an output terminal.

Test1 in the Sequencer should be an **Exec** transaction that calls **UpperLimit**.

Create another **UserFunction** called **AddRand** that adds a **Random Number** to Zero "Passed" in by calling **AddRand(0)** from within the **Sequencer** object.

Test the value returned by **AddRand** using a limit compare < **Global UL** as the limit value.

> If it passes, **THEN RETURN** "Pass" +test2.result.
> If it fails, **THEN RETURN** "Failed"+test2.result.

Put an **AlphaNumeric** display on the **Return pin** of the Sequencer.

Have the **Sequencer** object's Sequence output pin ping a **Get Global** object (UL) and send that value to another **AlphaNumeric display.**

Run the program several times.

Step 2
Disable the first test step. Assuming you don't need the global anywhere else, you can call the **UpperLimit UserFunction** directly. Change **Test2** so that it compares the Output value against the result of the **UpperLimit** function directly no longer using the Global in the limit check.

Extra Credit:
Load the example
Help=>Open Example=>Examples\new\mfgtest.vee

Look at the two Sequencer objects in this example. The first, **Login Control** prompts for a user's name and compares the password. The second Sequencer uses the **Exec Trans** control pin and accepts a real array of test strings to call specific tests.

Lab 11b

Logging Sequencer Data

Objective: To learn how the sequencer works and logs data.

Step 1
Edit a **Test1** Sequencer transaction that simply calls the VEE function **Random( )**. Compare the results against a Limit < 0.5.

Copy the **Test1** transaction and **Paste** it back several times until you have a total of 4 tests.

Step 2
Build a program to run the **Sequencer** five times with each **Sequencer Log** record going into a **Collector** array and a **Dataset**.

Using the Collected array data , find the **Min**, the **Max**, the **Mean**, and the **Standard Dev** of the results of the second test.

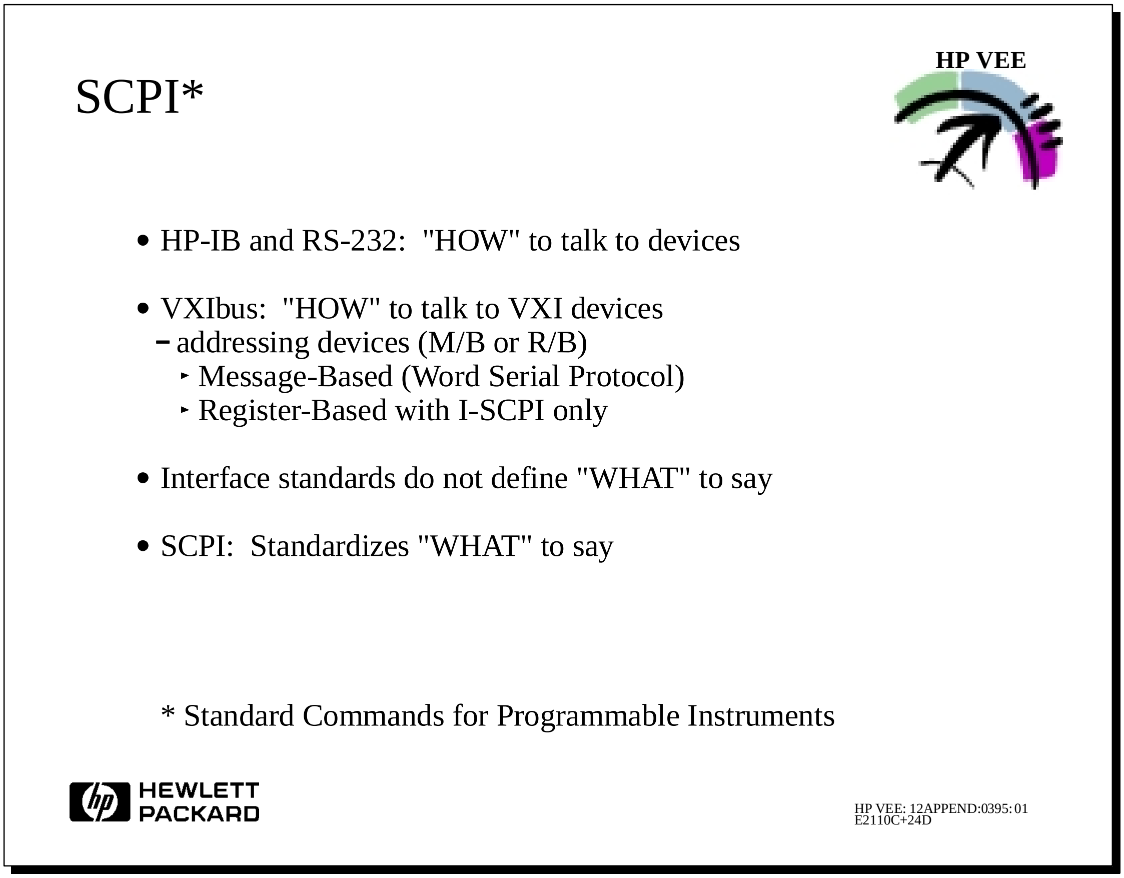
Hint: Use a formula box with **<record>.<record>.<field>**

Step 3

In a separate thread, get all of the records from the **Dataset** where the first test passed **OR** the second test failed. Print the timestamp field from the records on an **AlphaNumeric** display.

Hint: You may need to access the **Logging Config** menu of the **Sequencer's** object menu!

Appendix



HP VEE

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

HEWLETT PACKARD

HP VEE: 12APPEND:0395:01
E2110C+24D

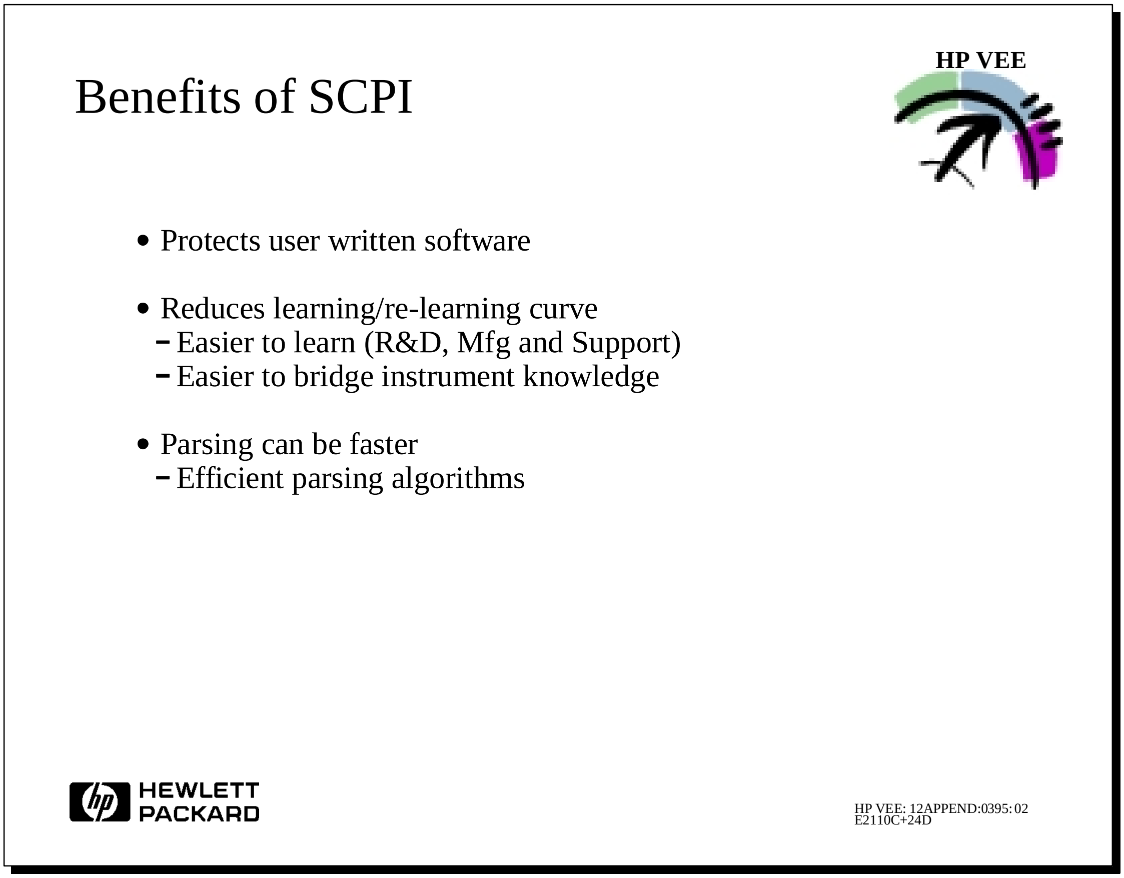
SCPI Consortium Founding Members

HP, TEK, FLUKE, Phillips, WaveTek, Racal Dana, Keithly, K&B, NI

Founded in 1990

Originated with TMSL (Test & Measurement Systems Language) from HP

Based upon IEEE 488.2 Codes, Formats, Protocols and Common Commands



HP VEE

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

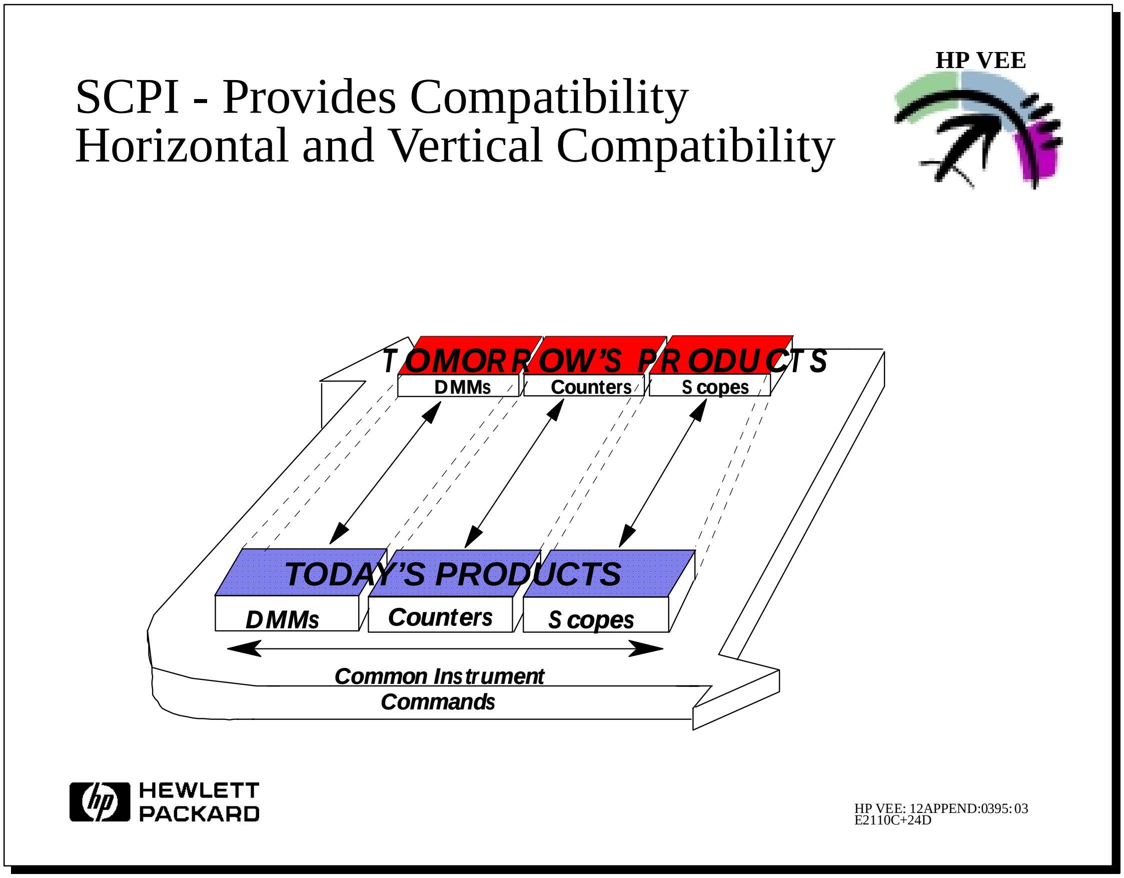
HEWLETT PACKARD

HP VEE: 12APPEND:0395: 02
E2110C+24D

SCPI compatible software should be protected since as instruments get replaced by new models for by different instrument types. See Horizontal and Vertical Comp.

Once the command structure is learned once, there is no need to learn new commands/mnemoics since they will stay the same. You would need to learn new command as Functions are added.

The parser is optimized for SCPI.



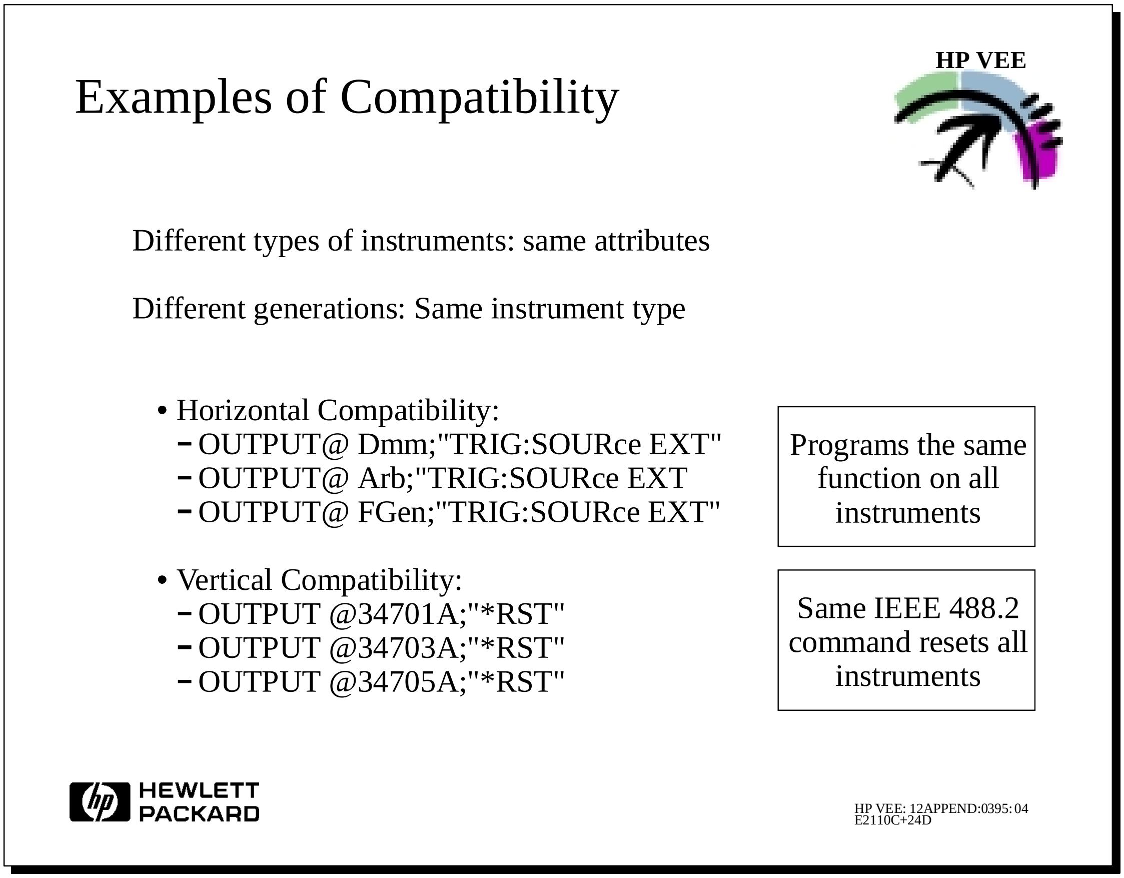
HP VEE

SCPI - Provides Compatibility
Horizontal and Vertical Compatibility

HP VEE: 12APPEND:0395:03
E2110C+24D

Horizontal compatibility implies that instruments will have a common mnemonic for common instruments. An example would be "MEAS:VOLT:DC" for a DMM and Scope.

Vertical compatibility implies that instruments will have common commands ACROSS a product line or as new generations of instruments come out (i.e. 34401, 3440x).



HP VEE

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

HEWLETT PACKARD

HP VEE: 12APPEND:0395: 04
E2110C+24D