Welcome to
Introduction to
HP VEE

Number E2100+24D

© 1991 Hewlett-Packard Company

The Approach for Learning HP VEE

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

© 1991 Hewlett-Packard Company

VEE Operation Fundamentals

- Synchronous Operation
- Propagation Rules
- Multiple Threads

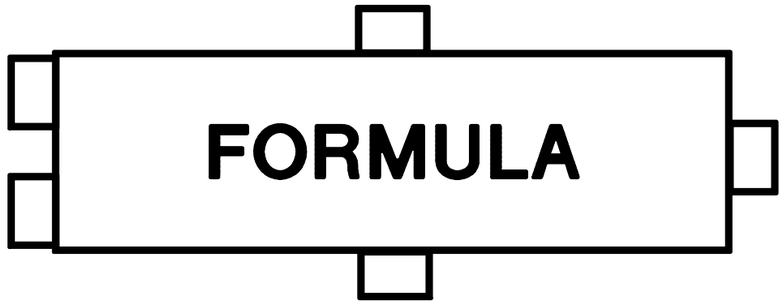
© 1991 Hewlett-Packard Company

Useful Definitions For HP VEE

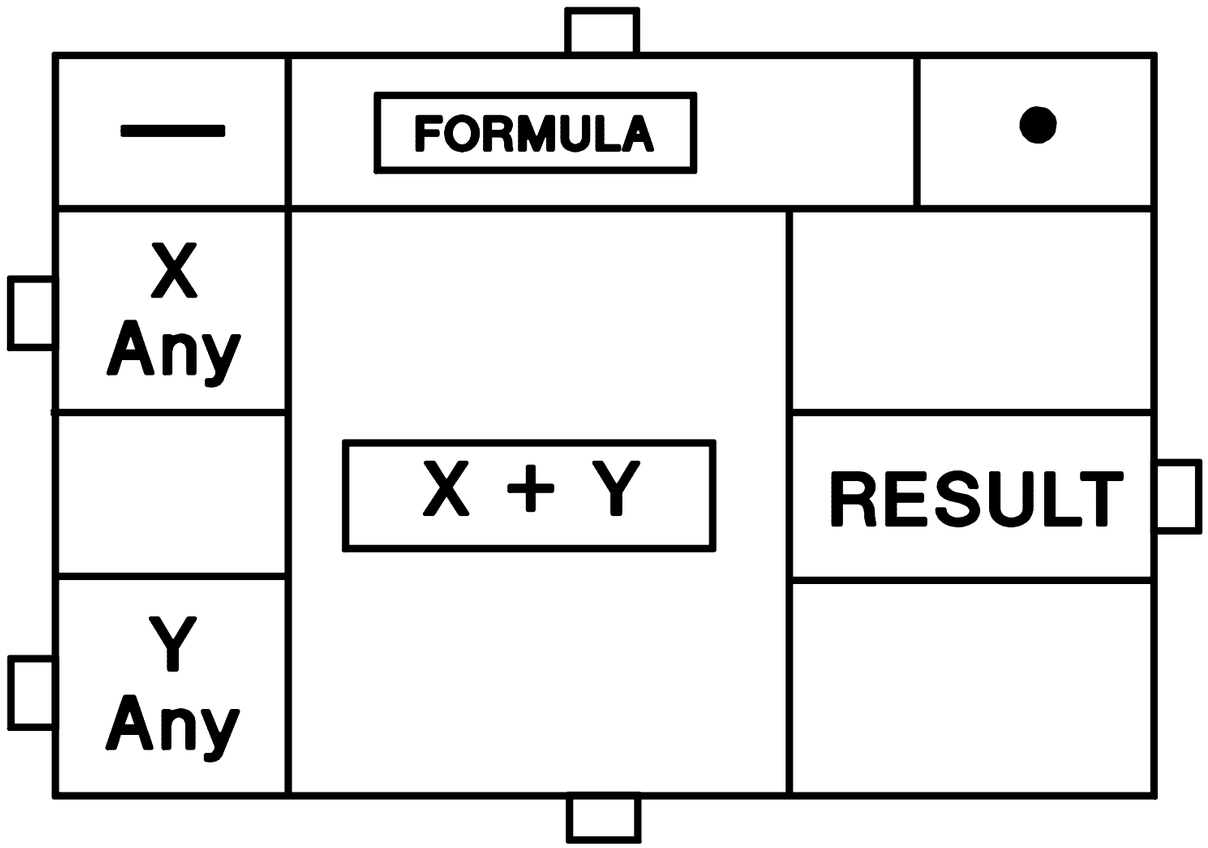
■ Model	The finished solutions with objects linked together (not a program – a model)
■ Work Area	"the executable block diagram" The area within the HP VEE window in which you build models
■ Object	Any item placed on the work area
■ Icon	A small, graphical representation of an object
■ Open View	The maximized view of an object

©1991 Hewlett-Packard Company

Icon vs Open View



ICON



OPEN VIEW

© 1991 Hewlett-Packard Company

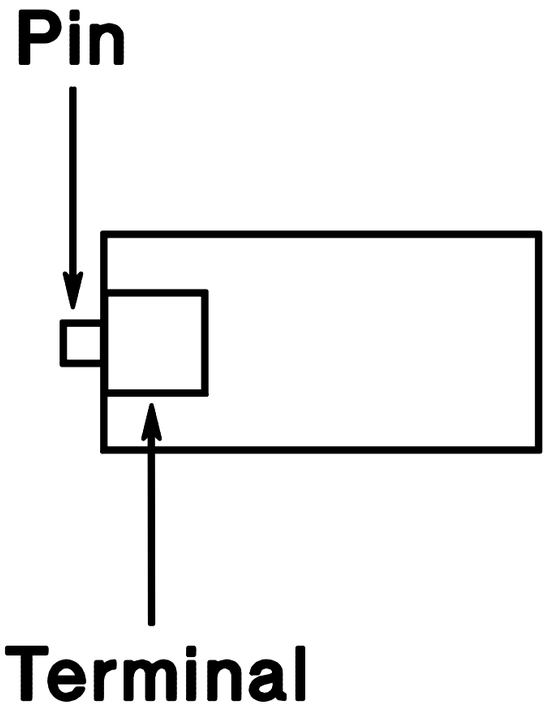
More Useful Definitions For HP VEE

- **Thread** — A set of objects connected by solid lines
- **Activate** — Send data or sequence instructions to a terminal or pin
- **Operate** — To activate an object
- **Ping** — A message that is used to initiate execution
- **Container** — The package that is transmitted over lines and is processed by objects

©1991 Hewlett-Packard Company

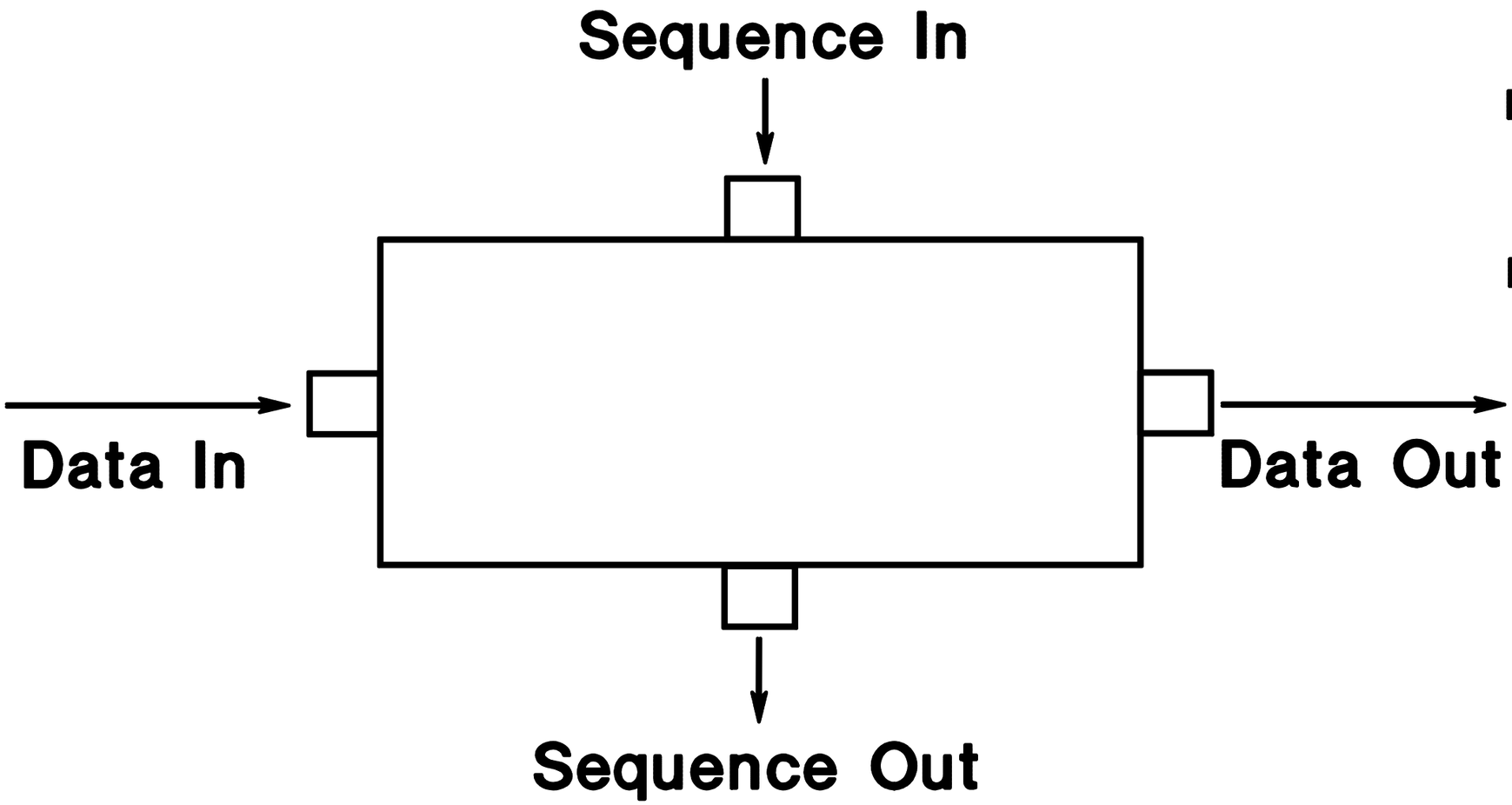
Pins & Terminals

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



©1991 Hewlett-Packard Company

Synchronous Operation

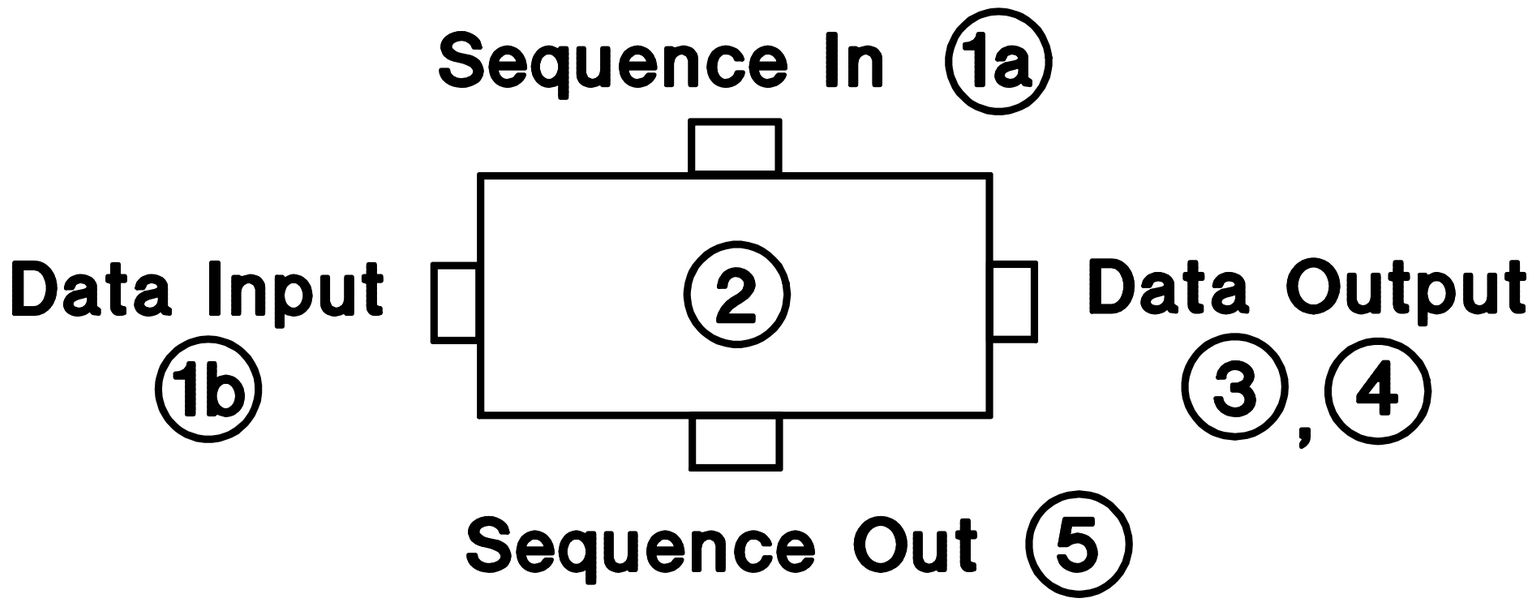


- Data flows left-to-right
- Sequence flows top-to-bottom

- All DATA IN pins must be connected for an object to fire
- A single DATA IN pin cannot accept more than one line

©1991 Hewlett-Packard Company

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"

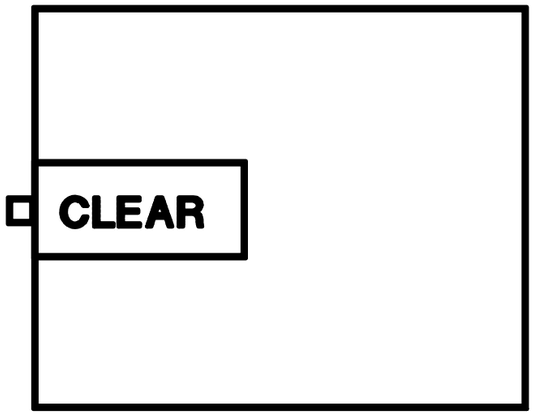
5 Sequence out fires

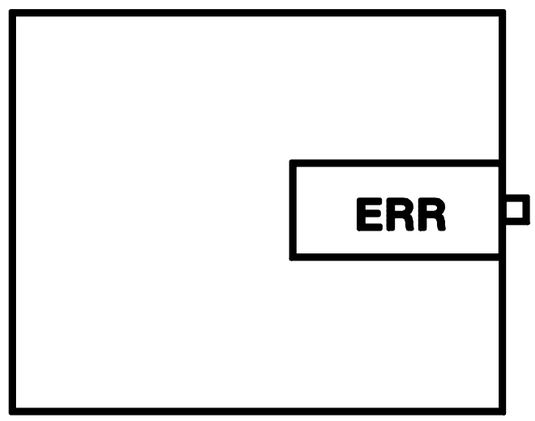
6 Object ceases operation

© 1991 Hewlett-Packard Company

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





©1991 Hewlett-Packard Company

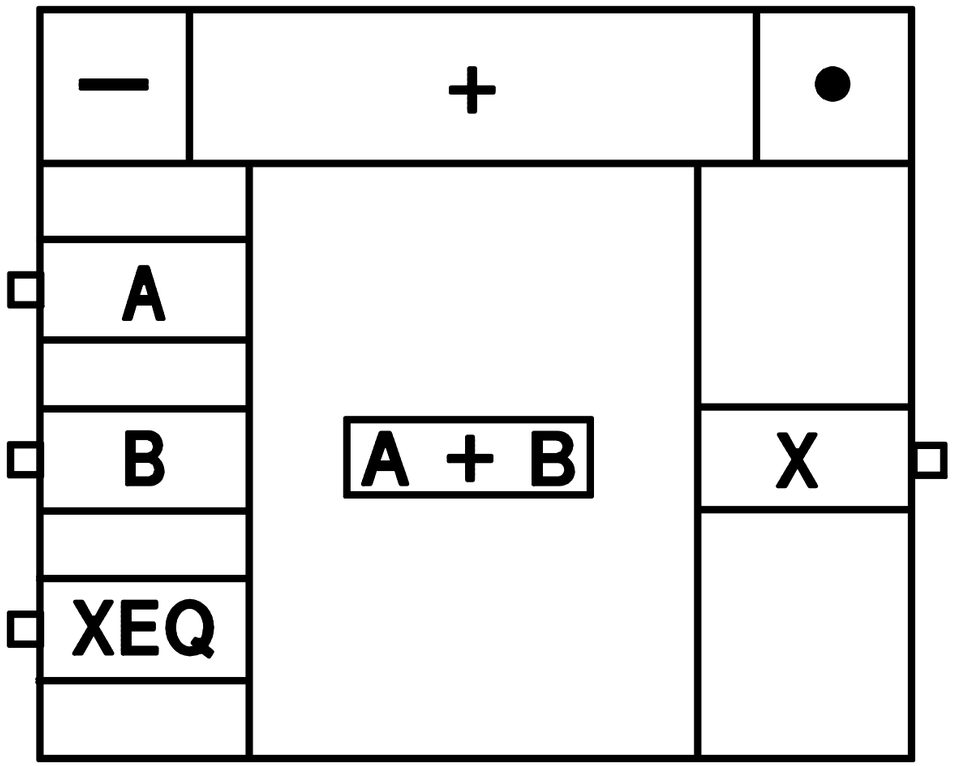
Adding Optional Inputs

- Object menu provides ability to add terminals to objects
  - Data, Control, Trigger (User Objects Only)
  - Inputs and Outputs
- Terminal can be opened to edit name
  - Type and shape can be modified if required

© 1991 Hewlett-Packard Company

Overriding Constraints

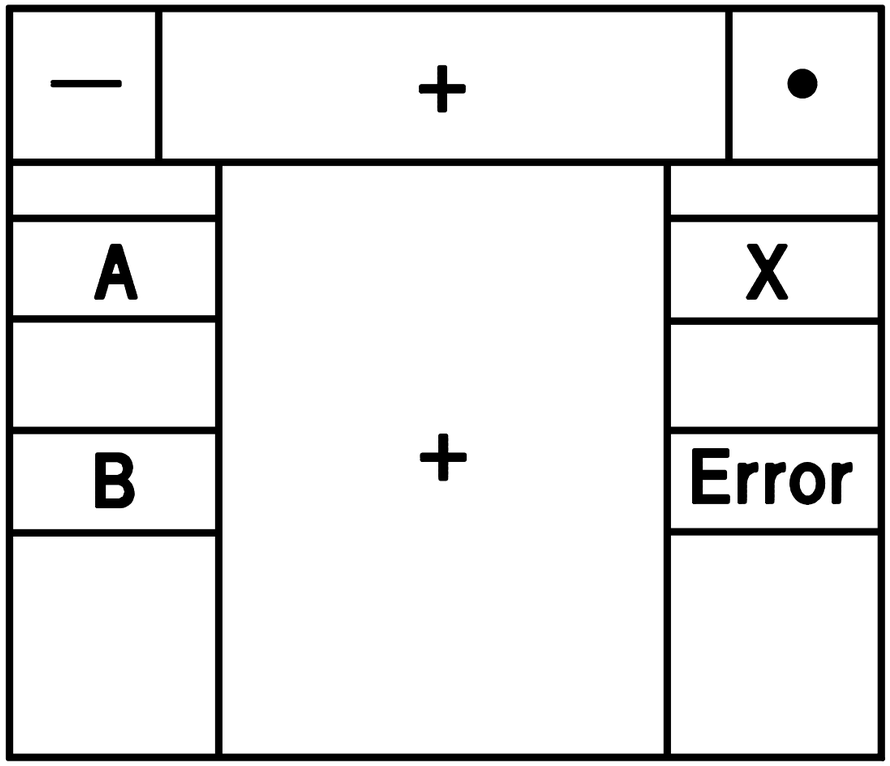
- XEQ control causes immediate object operation
  - Available data used
- Useful for continuing after error
- Required by some data building objects



© 1991 Hewlett-Packard Company

Trapping Errors

- Error output pin can be added
- Allows HP VEE to continue execution after error
  - Error pin activates
    Instead of data output pins
  - Output container holds error number



© 1991 Hewlett-Packard Company

Propagation Rules

- Pre-Run & Activation, Auto Execute
- Order of Execution
- Parallel Subthreads

©1991 Hewlett-Packard Company

Propagation Definitions

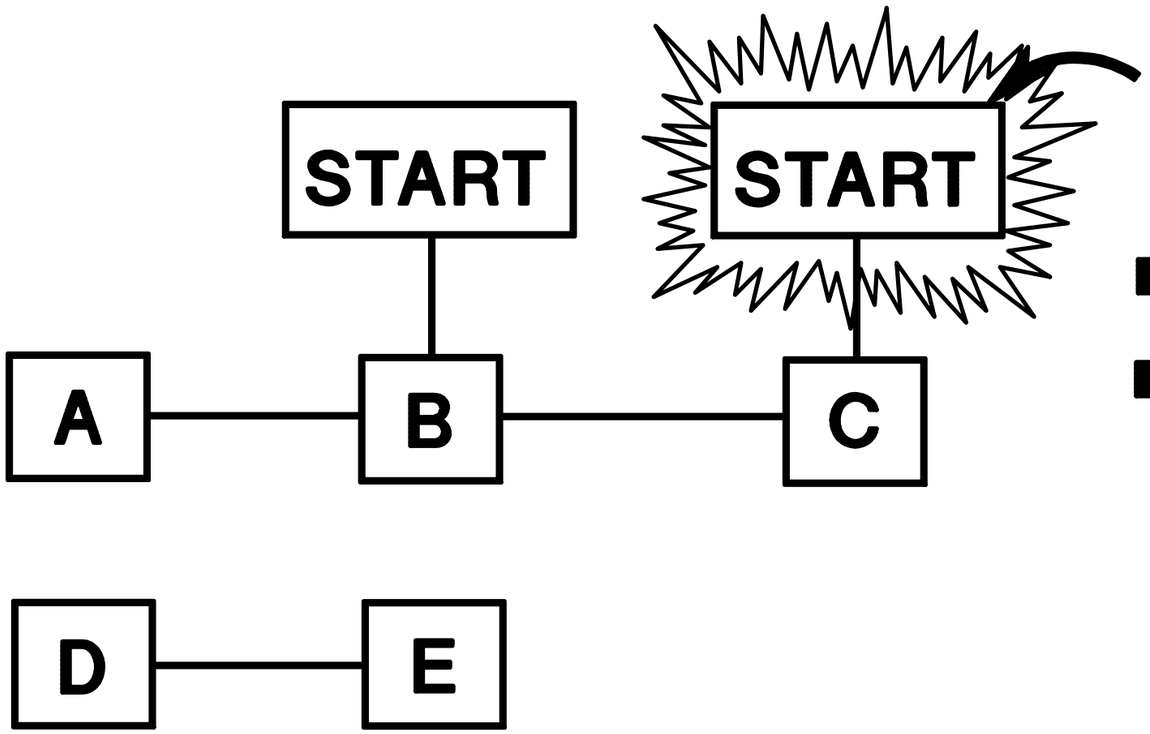
- PreRun
  - Checks for "static" structure of model
  - Feedback loops, connected inputs
  - Occurs for entire model when Run pressed
  - Occurs for single thread if Start pressed
  - Objects reset to initial conditions
  - Files rewound
  - Errors cleared

- Activate
  - Analogous to procedure call
  - Analogous to PreRun for individual UserObject
- Auto Execute
  - Propagation initiates at Data object, after user input

©1991 Hewlett-Packard Company

Start Objects

- 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

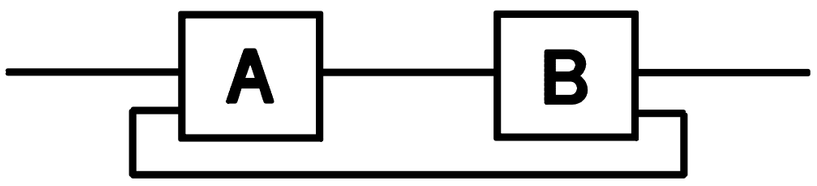


- Initiates thread propagation
- D & E unaffected

©1991 Hewlett-Packard Company

Start Object

- Never required EXCEPT to resolve FEEDBACK

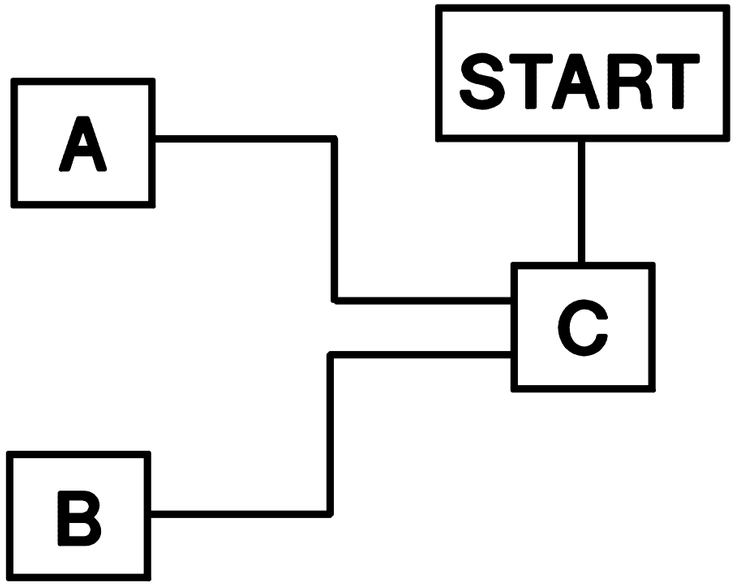


- A cannot operate until BOTH its data pins are pinged
- B cannot operate until A does
- At PreRun, dialog box will advise you to supply a Start

©1991 Hewlett-Packard Company

Propagation

- Unconstrained Objects
  - No input constraints
- Constrained Objects
  - Have either Data Input or Sequence Input pins connected

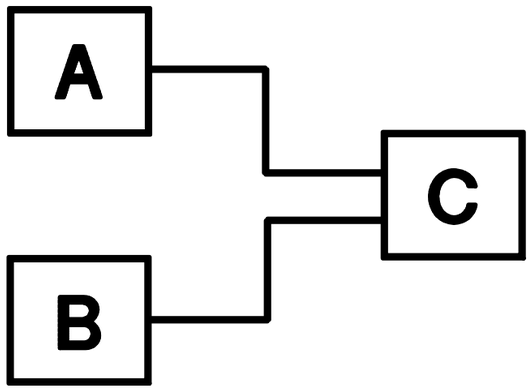


- A or B may operate anytime after Start

©1991 Hewlett-Packard Company

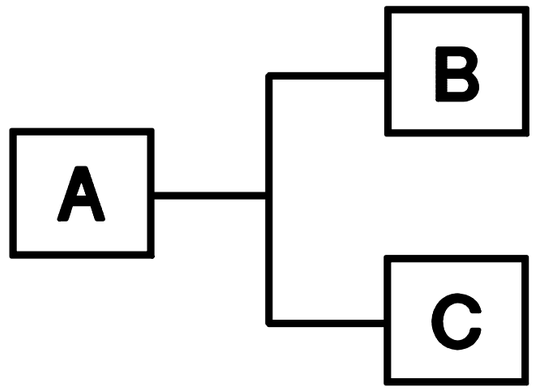
Propagation Rules

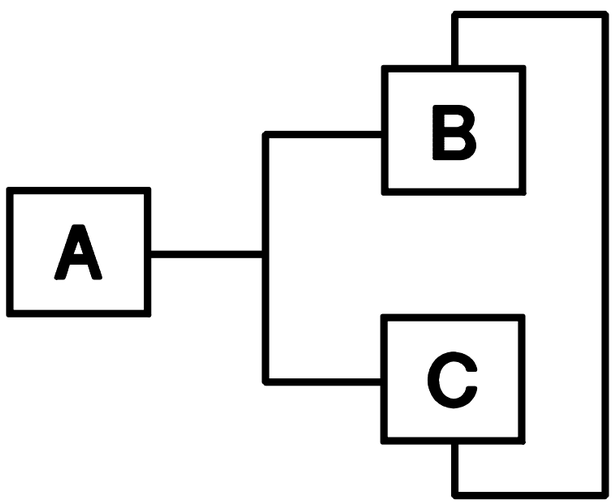
- Unconstrained objects may operate at any time



  - 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

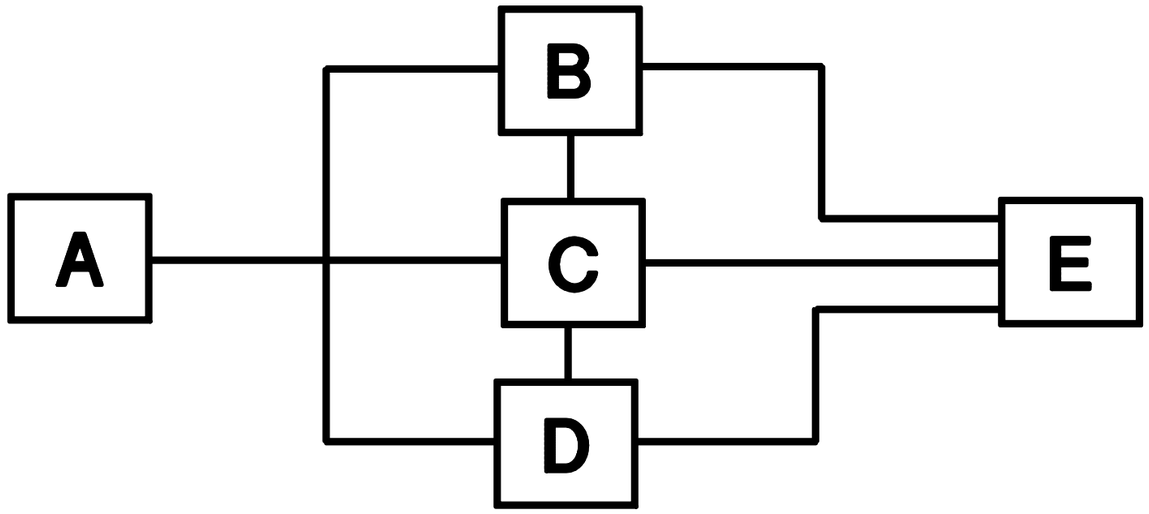




  - Use sequence pins if order matters

©1991 Hewlett-Packard Company

Propagation Example

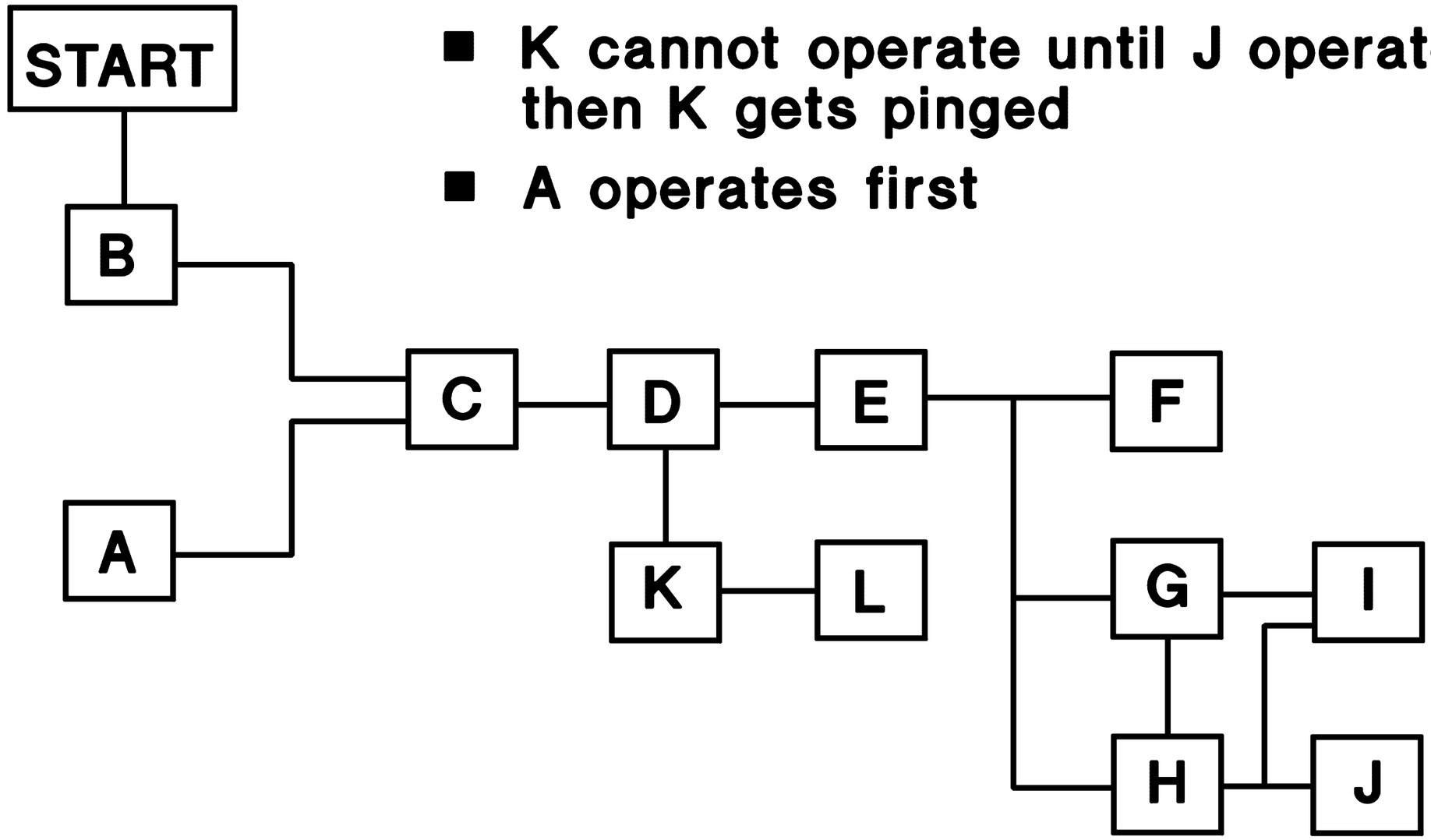


- Notice that the sequence lines order the correct execution of B, C, and D

EXECUTE	DONE

©1991 Hewlett-Packard Company

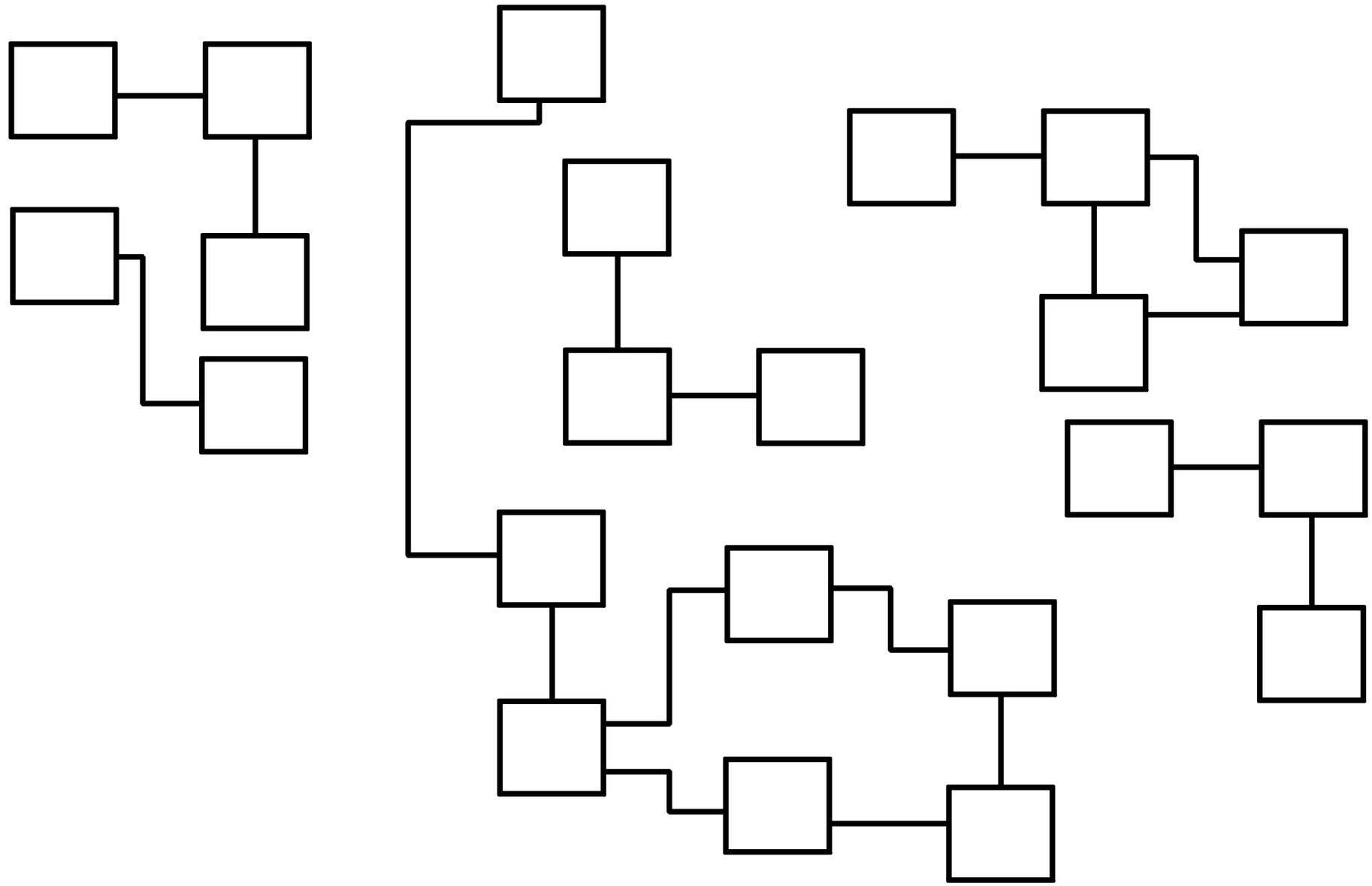
Propagation Example



- K cannot operate until J operates then K gets pinged
- A operates first

© 1991 Hewlett-Packard Company

Multiple Threads



- Many threads can be built in the same work area
- A Start object will selectively execute a thread
- Press Run to have them all operate (time-slice between threads)

© 1991 Hewlett-Packard Company

Multiple Thread Propagation

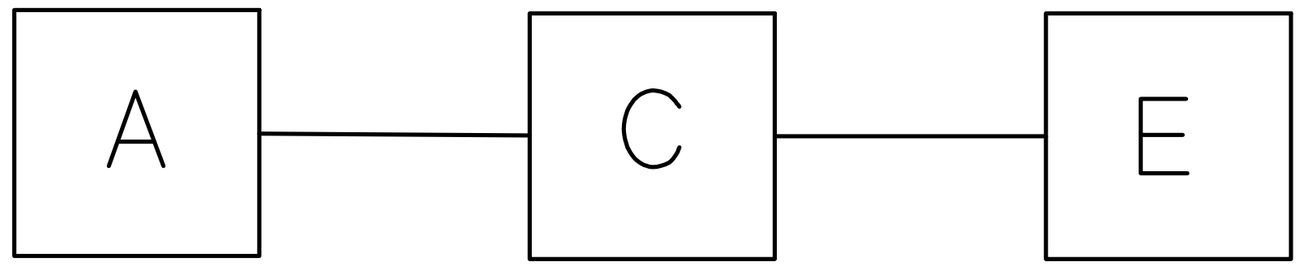
- Parallel threads are time-sliced by "propagation engine"
- Time slice ≡ 1 primitive object
  - Note:
    - Each object on an iterating subthread of a repeat device (iterator) counts as one timeslice
    - UserObjects are MULTIPLE objects

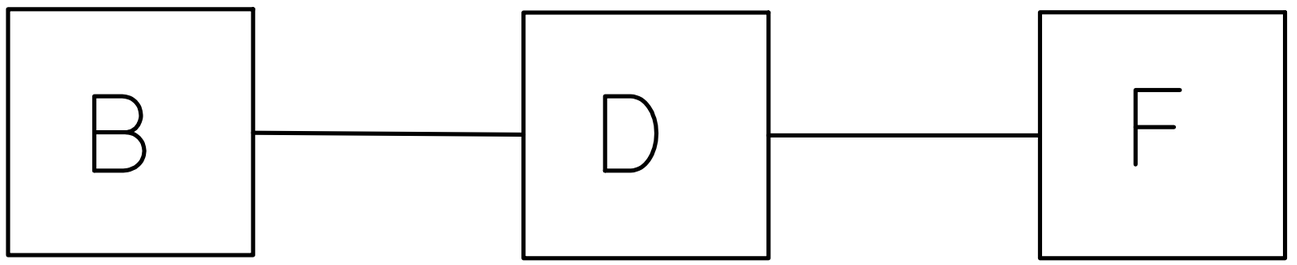
Each object in a UserObject is a primitive object

- Parallel sub-threads also time-slice

©1991 Hewlett-Packard Company

Parallel Subthread Example





© 1991 Hewlett-Packard Company

Debugging

- Show Execution Flow
  - Highlights each object on operation
- Show Data Flow
  - Shows data container moving along threads
- Set Breakpoints
  - Stops execution at this point
- Line Probe
  - Shows data container on thread

©1991 Hewlett-Packard Company

Data Types & Structures

- Int16
- Int32
- Real
- Coord
- Enum
- String
- Complex
- PComplex
- Spectrum
- Waveform

©1991 Hewlett-Packard Company

INT16

- Signed 16 - Bit Integer
- Integer Numbers from -32768 to +32767
- Used only for I/O to files and instruments

©1991 Hewlett-Packard Company

INT32

- Signed 32 - Bit Integer
- Integer Numbers from -2^{32} to $2^{32}-1$

©1991 Hewlett-Packard Company

Real

- Real Numbers from -1E308 to +1E308
- IEEE 754 64-bit format
- Includes Date/Time REAL (seconds since midnight, January 1, 1 A.D.)

©1991 Hewlett-Packard Company

Coord

- (REAL x1, [REAL x2, ...] REAL y)
- Explicit n-dimensional data points

- One independent variable

- Multiple dependent variables

©1991 Hewlett-Packard Company

Enum

- {TEXT value, TEXT value, ...}
- Maps onto {0,1,2...}
- Use ordinal(x) to go from TEXT representation to positional representation
- An array of ENUM becomes TEXT

©1991 Hewlett-Packard Company

Text

- Arbitrary length of characters

©1991 Hewlett-Packard Company

Complex/PComplex

- (REAL real-part, REAL imag-part)
- (REAL magnitude, @REAL phase)
  - Phase can be degrees, radians or gradians

©1991 Hewlett-Packard Company

Spectrum

- {PCOMPLEX data[ARRAY], REAL fstart, REAL fstop, ..}
- {PCOMPLEX data[ARRAY], REAL center, REAL span, ...}
- Data is mapped onto frequency domain
- Assumes uniform frequency (linear or logarithmic)

©1991 Hewlett-Packard Company

Waveform

- {REAL data[ARRAY], REAL timespan}
- Data array is mapped onto time domain
- Assumes uniform $\triangle t$

©1991 Hewlett-Packard Company

Automatic Data Typing

- Data containers have "data type" tag
- Many objects accept "any" data type
- Objects can generate many different data types
- Type conversion can happen to resolve dissimilar types:
  - Conversion to match input constraints
  - Promotion only so operands match
  - Conversion to match transactions

©1991 Hewlett-Packard Company

Data Promotion & Demotion

- Principles
  - When combining data types, lower is promoted whenever possible (for math operations)
  - Must be same shape (on terminals)
- Data Loss
  - When device expects a fixed data type, "higher" types may lose information to conform

©1991 Hewlett-Packard Company

Data Promotion & Demotion

FROM TYPE / TO TYPE

	Int16	Int32	Real	Complex	P–Complex	Wave–form	Spectrum	Coord	Enum	String
Int16	Y	Y	Y	*	*			*		Y
Int32	*	Y	Y	*	*			*		Y
Real	*	*	Y	*	*			*		Y
Complex				Y	Y					Y
PComplex				Y	Y					Y
Waveform	*	*	*			Y	*	Y		Y
Spectrum				*	*	*	Y			Y
Coord								Y		Y
Enum									Y	Y
String	*	*	*	*	*			*		Y

©1991 Hewlett-Packard Company

Data Objects

- Enum – User selects one of a list of choices
- Toggle – User toggles the object on or off
- Slider – User slides a bar to select a value
  – Step value (detents) is selectable

©1991 Hewlett-Packard Company

Constant Types

Allows the user to type in values to define the fields

- **Text**
- **Integer**
- **Real**
- **Coord**
- **Complex**
- **PComplex**
- **Date/Time**

Evaluates simple calculations (no variables)

©1991 Hewlett-Packard Company

Data Objects – Unique Capabilities

- Initial Value (Initialize at PreRun, Activate)
- Auto Execute
- Config (Array Size)
- Set Number Formats
- Set Object Format (Enum: List, Cyclic, Buttons)
- Edit Values

Many of these capabilities can be added as Control Inputs

©1991 Hewlett-Packard Company

Object Menu

ICON

- Move
- Size
- Clone
- Help
- Show Description
- Breakpoint
- Terminals → (Add or Delete Terminal)
- Layout → (Allows user-selectable bit maps for Icons)
- Cut

OPEN VIEW

- Move
- Size
- Clone
- Help
- Show Description
- Breakpoint
- Terminals → (Add or Delete Terminal)

{ OBJECT SPECIFIC INSTRUCTIONS

- Cut

©1991 Hewlett-Packard Company

Data Builder/UnBuilder

- Create specific data types
- Retrieve parts of data types
- Requires allocated array:
  - Get/Set values (array), Get/Set mapping (function), build spectrum, build waveform
- Does NOT require allocated array:
  - Build Coord, build PComplex, build Complex, build arb waveform

Other objects build data

See Also

- Collector; virtual arbitrary waveform; generate step, ramp, impulse, etc.

©1991 Hewlett-Packard Company

Sequence Control

- Start
  - Initiates execution of a thread
  - All START boxes are "pressed" (in no particular order) if RUN is pressed
- Confirm (OK)
  - Awaits user confirmation before continuing sequence
- Do
  - Allows developer to specify which subthread fires first
  - Propagates a sub-thread

©1991 Hewlett-Packard Company

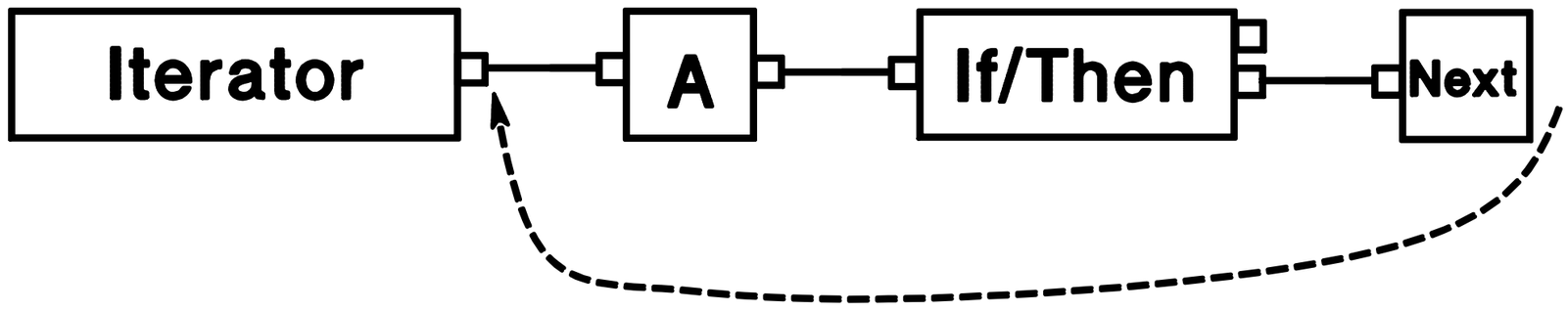
Repeat (Iterators)

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

©1991 Hewlett-Packard Company

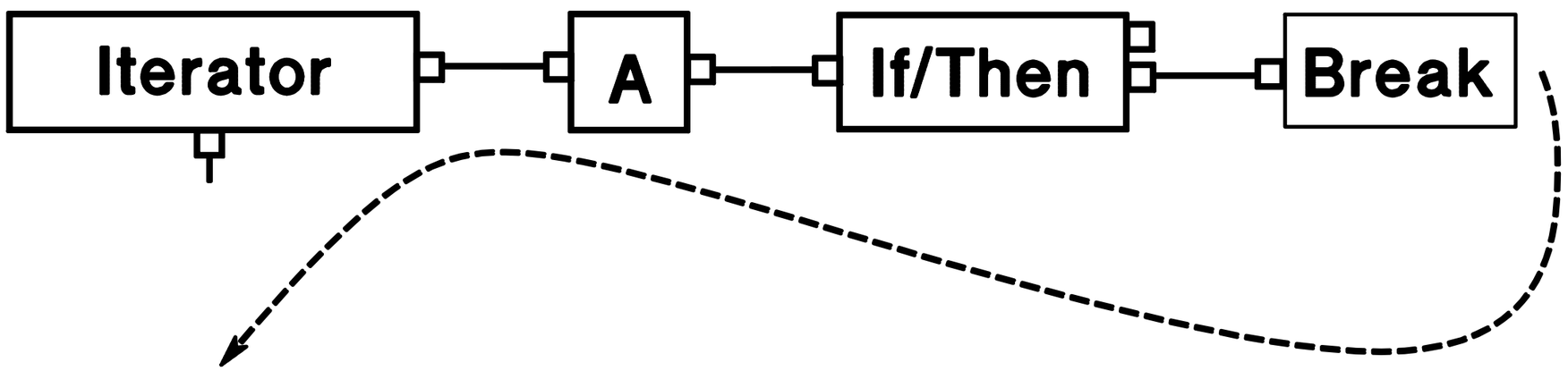
Early Loop Termination

- **Next – terminates propagation of current iteration**



– A executes repeatedly

- **Break – terminates current and future iterations**



– A executes repeatedly

©1991 Hewlett-Packard Company

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 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)

©1991 Hewlett-Packard Company

Conditional Branching

- If/Then
  - 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

==	Equal
!=	Not equal
>,<	Greater, less than
>=,<=	Greater or equal, less or equal

©1991 Hewlett-Packard Company

Termination (Exits)

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

©1991 Hewlett-Packard Company

Time Related Objects

- Delay
  - Delays propagation for n seconds
- Timer
  - Measures execution time between two objects
- Time Stamp
  - Indicates time of execution
- Resolution
  - HP-UX system clock : 1/60 sec

    (platform dependent)

©1991 Hewlett-Packard Company

Misc. Objects

- Counter
  - Counts activations
- Accumulator
  - Running total
- Shift Register
  - Holds previous values
- DeMultiplexer
  - Redirect data or flow to 1 of n outputs
- Comparator
  - Compares data
  - Counts failures
  - Collects failures

©1991 Hewlett-Packard Company

Textual Displays

- Alphanumeric
  - Displays a single value
- Logging Alphanumeric
  - Scrolling text display
  - Configure buffer size
- VU Meter
  - Analog meter
  - Red, yellow, green limits available

All textual displays allow:

- Clear at PreRun
- Clear at Activate
- Number Formats

©1991 Hewlett-Packard Company

Graphical Displays

- XY Trace
  - Y data points plotted on an arbitrary X axis
- Strip Chart
  - Y data points plotted on an arbitrary axis
  - X and Y axis both scroll as data is received
- Complex Plane
  - Plots Real vs. Imaginary
- X vs Y Plot
  - Plots pairs of X,Y data points
- Polar Plot
  - Plots radius vs. angle
  - Includes options for Smith, Inverted Smith
- Waveform (Time)
  - Plots waveform data type data
- Spectrum

©1991 Hewlett-Packard Company

Spectrum Displays

- **Magnitude**
  - Plots magnitude vs frequency
- **Phase**
  - Plots phase vs. frequency
- **Magnitude vs Phase**

  Polar
  - Data plotted on a polar plane

  Smith
  - Data plotted on a Smith chart

©1991 Hewlett-Packard Company

Display Customization

- Multiple Data, Control Inputs
- Autoscale (X only, Y only, Both)
- Clear control (at Activate, at PreRun, Next Curve)
- Zoom (In, Out, Etc.)

©1991 Hewlett-Packard Company

Display Customization (cont.)

- Markers (One, Two, Delta,Interpolate, Etc.)
- Grid Type (Tic Marks, Lines, Etc.)
- Panel Layout (Graph Only, Show Scales, Etc.)
- Set Trace (Scaling, Color, Line Type, Etc.)
- Add Additional Scales on the Right

Note that many functions can be added as control pins

©1991 Hewlett-Packard Company

Math

- Formula Box
  - Accepts any HP VEE Math function
  - Includes expression evaluation and conditional capabilities

©1991 Hewlett-Packard Company

Math

- Generate (Functions)
  - Ramp
  - Log Ramp

©1991 Hewlett-Packard Company

Math

- **General**
  - +
  - –
  - *
  - /
  - ^
  - Mod
  - Div
- **Relational**
  - ==
  - !=
  - <
  - >
  - <=
  - >=
- **Bitwise**
  - Bit
  - Bits
  - Set bit
  - Clear bit
  - Bit and
  - Bit or
  - Bit xor
  - Bit compl
  - Bit shift
- **Logical**
  - And
  - Or
  - Xor
  - Not

©1991 Hewlett-Packard Company

Math

- **Real Parts**
  - Abs
  - Sign of
  - Ordinal
  - Round
  - Floor
  - Ceil
  - Int part
  - Frac part
- **Complex Parts**
  - J
  - Re
  - Im
  - Mag
  - Phase
  - Conj

©1991 Hewlett-Packard Company

Math

- **Power**

  Sq
  Sqrt
  Cubert (cube root)
  Recip
  Log
  Log 10
  Exp
  Exp 10

- **Polynomial**

  1: Poly (x,[aØ a1])
  2: Poly (x,[aØ a1 a2])
  3: Poly (x,[aØ a1 a2 a3])
  N: Poly (x,[aØ a1 ...aN])

©1991 Hewlett-Packard Company

Math

- **Trig**

  Sin
  Cos
  Tan
  Cot
  Asin
  Acos
  Atan
  Acot
  Atan2

- **Hyperbolic Trig**

  Sinh
  Cosh
  Tanh
  Coth
  Asinh
  Acosh
  Atanh
  Acoth

©1991 Hewlett-Packard Company

Math

- **Time & Date**

  Now
  Wday
  Mday
  Month
  Year
  Dmytodate
  Hmstosec
  Hmstohour

©1991 Hewlett-Packard Company

Advanced Math

- **Array**

  Init
  Rotate
  Concat
  Sum
  Prod

- **Matrix**

  Det
  Inverse
  Transpose
  Identity
  Minor
  Cofactor
  Matmultiply
  Matdivide

- **Calculus**

  Integral
  Deriv(x,1)
  Deriv(x,2)
  Deriv(x,order)
  Defintegral
  Derivat(x,1,pt)
  Derivat(x,2,pt)
  Derivat(x, order, pt)

©1991 Hewlett-Packard Company

Advanced Math

- **Regression**

  Linear
  Logarithmic
  Exponential
  Power curve
  Polynomial

- **Data Filtering**

  Polysmooth
  Meansmooth
  Movingavg
  Clipupper
  Cliplower
  Minindex
  Maxindex
  Minx
  Maxx

©1991 Hewlett-Packard Company

Advanced Math

- **Probability**

  Random (low, high)
  Randomize
  Random seed
  Perm
  Comb
  Gamma
  Beta
  Factorial
  Binomial
  Erfc
  Erf

- **Statistics**

  Min
  Max
  Median
  Mode
  Mean
  Sdev
  Vari
  Rms

- **Freq. Distribution**

  Lin mag dist
  Log mag dist

©1991 Hewlett-Packard Company

Advanced Math

- **Bessel**
  - JØ
  - J1
  - Jn
  - YØ
  - Y1
  - Yn
  - Ai
  - Bi
- **Hyper Bessel**
  - IØ
  - I1
  - KØ
  - K1
- **Signal Processing**
  - Fft
  - Ifft
  - Convolve
  - Xcorrelate
  - Bartlett
  - Hamming
  - Hanning
  - Blackman
  - Rect

©1991 Hewlett-Packard Company

Virtual Source Objects

- Simulated function, pulse, noise, arb waveform generators
- Generate dynamic data models
  - Useful for prototyping
- Full control of model
  - Phase, amplitude, sample points, etc.
- Waveform data type output

© 1991 Hewlett-Packard Company

Function Generator

- **Functions**
  - Sine, Cosine, Square, Triangle, +Ramp, −Ramp, DcOnly
- **Frequency**
- **Amplitude**
- **DC Offset**
- **Phase**
  - Deg, Rad, Grad
- **Time Span**
  - Time interval for waveform sample
- **Num Points**
  - Determines sampling detail
  - Too few points can cause aliasing

© 1991 Hewlett-Packard Company

Pulse Generator

- Rep Rate
  - Repetitions per second
- Pulse Width
- Pulse Delay
  - From 0 seconds
- Thresholds
  - 0%–100%, 10%–90%, 20%–80%
    (rise/fall time calculation)
- High – Maximum Value
- Low – Minimum Value
- Burst Mode
  - On/Off
  - Burst Count
  - Burst Rep Rate (per second)

© 1991 Hewlett-Packard Company

Pulse Generator

- Interval
  - Waveform time interval
- Num Points
  - Sampling size

© 1991 Hewlett-Packard Company

Noise Generator

- Generates Pseudo Random Noise
- Control
  - Amplitude
  - Interval (time)
  - Num Points (sampling size)

© 1991 Hewlett-Packard Company

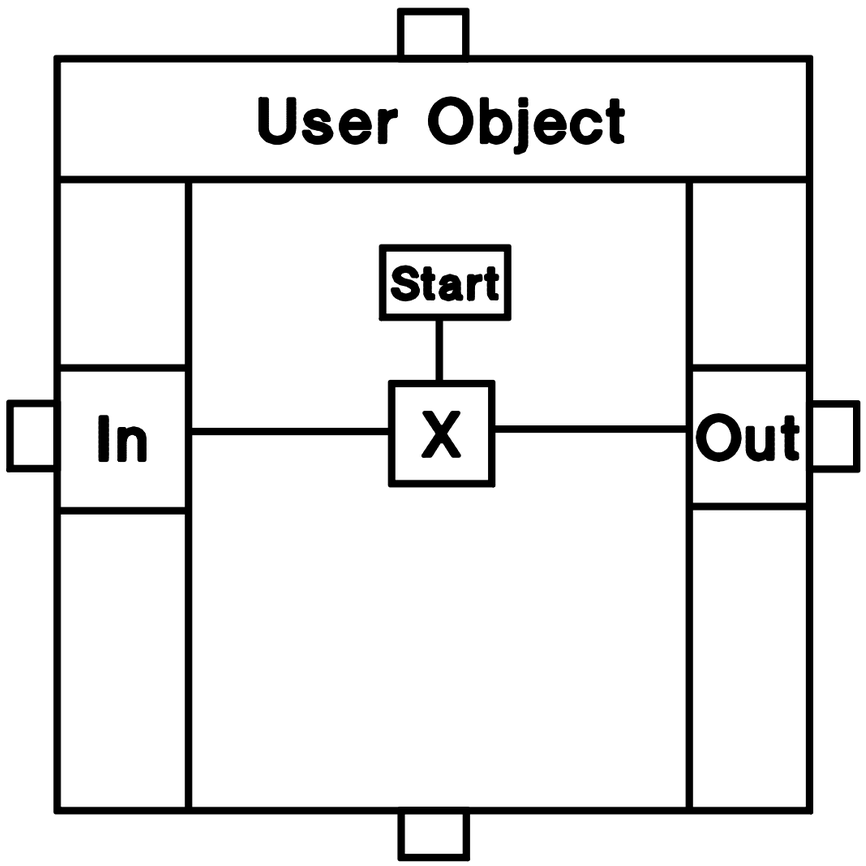
Other Waveform Sources

- **Build Waveform**
- **Build Arb Waveform**

© 1991 Hewlett-Packard Company

UserObjects

- A work area within an object



- A Context

©1991 Hewlett-Packard Company

UserObjects

Properties

- Obey all object rules
  - Operate once per thread execution
  - Need all data and sequence inputs satisfied
- Behave like work area
  - Supports all objects
  - Supports multiple threads

©1991 Hewlett-Packard Company

Purpose

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

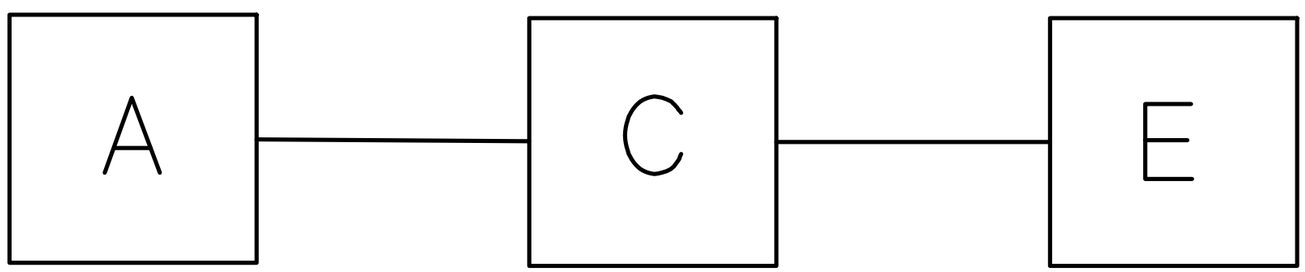
©1991 Hewlett-Packard Company

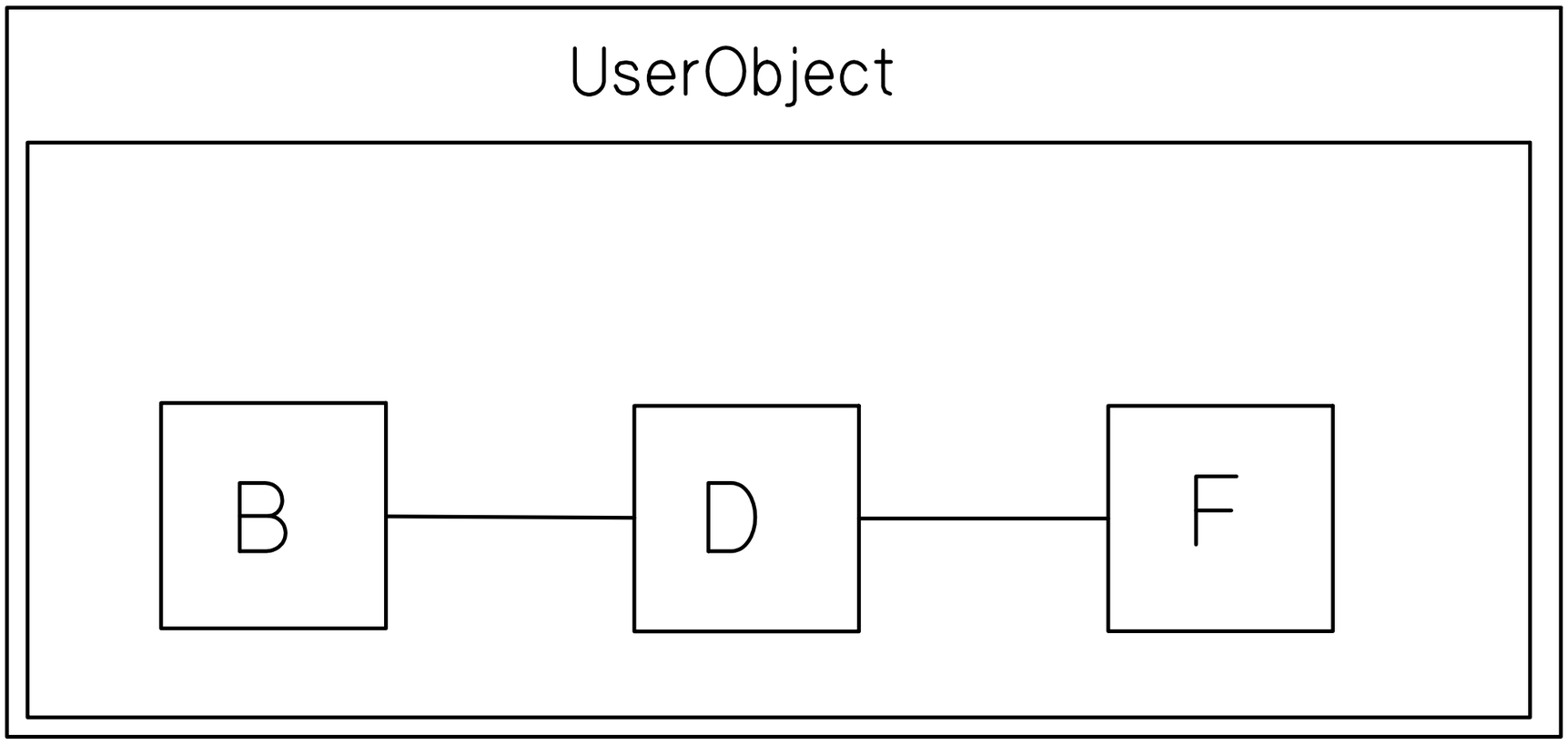
Initiation and Execution of UserObjects

- Activated when data and sequence inputs are satisfied
- Data inputs act as Start objects
- Each object operates in a time-sliced fashion
- Internal vs. external activation
  - Internal activation (Start); data does not activate data out pins

©1991 Hewlett-Packard Company

Parallel Subthread Example





© 1991 Hewlett-Packard Company

Termination of UserObjects

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

©1991 Hewlett-Packard Company

Early Termination

- Exit UserObject
  - All threads in context halt
  - Outputs which received data activate
  - Sequence out activates
- Escape
  - User-generated error
  - All threads in context halt
  - NO data pins activate
  - Error pin generates escape code
    Else "error" dialog

- Errors can "bubble up" through nested UserObjects

©1991 Hewlett-Packard Company

Building a UserObject - Encapsulate Existing Objects (Method 1)

- Select desired object(s)
- Create UserObject

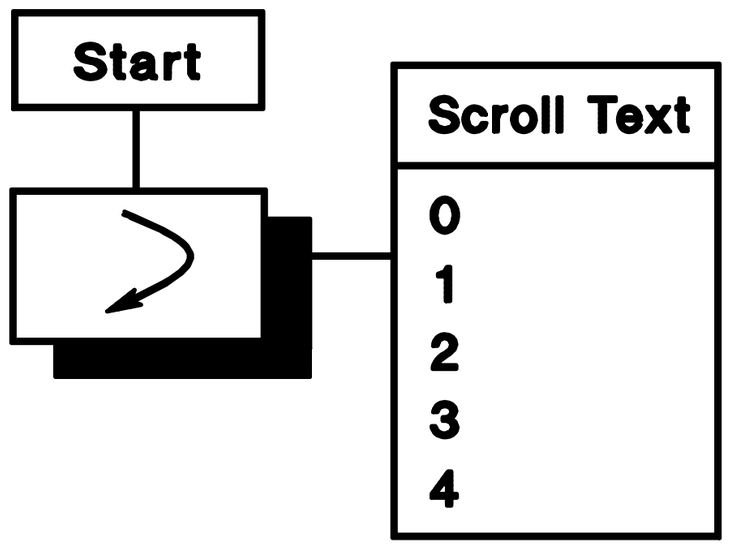
- **Advantages**
  - All connections become data pins
  - Allows prototyping in top-level workarea

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

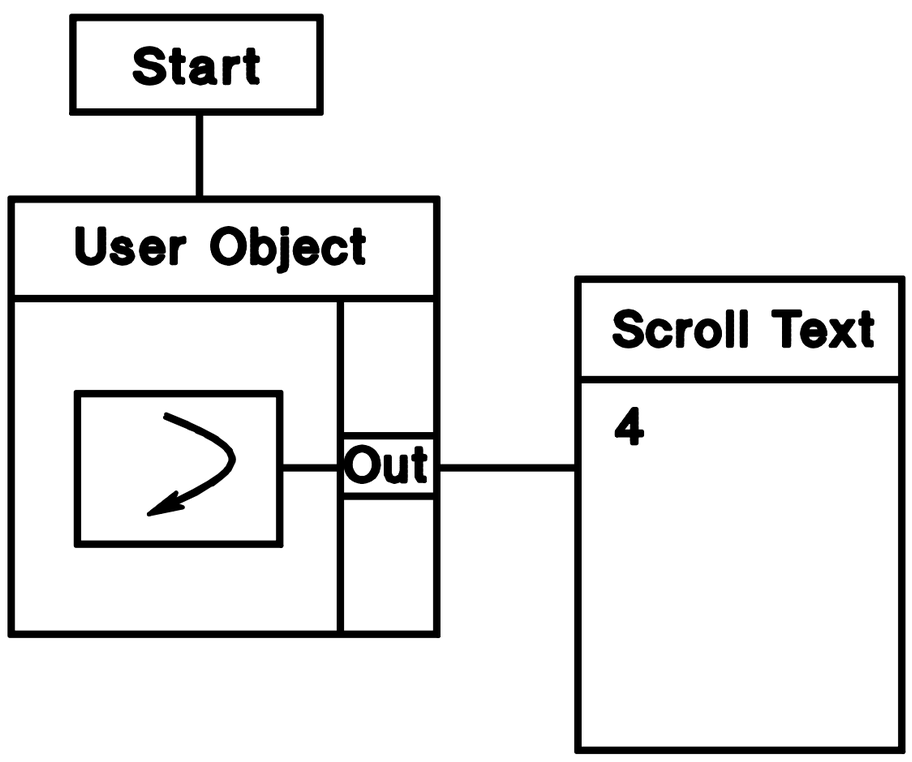
©1991 Hewlett-Packard Company

Common Problem in Create User Object

- The working model...



- ... partially encapsulated



- GIVES DIFFERENT RESULTS!

©1991 Hewlett-Packard Company

Benefits of Structured Modelling

- Logically correct
- Easy to see and understand
- Easy to change and maintain
- Easy to review by peers

© 1991 Hewlett-Packard Company

Top Down Design

- Define the problem and its constraints
- Identify and define logical order and sequence
- Define subtasks
- Further define each subtask into manageable units
- Implement units
  - UserObjects
- Structured programming
  - Exactly same principles apply as in languages

© 1991 Hewlett-Packard Company

Building a UserObject (Method 2)

- Start with empty objects: use as stubs initially
- Build the model that will provide the basic unit of functionality: procedure calls
- Add data inputs and outputs: parameters and results
  - No sequence lines or control lines or trigger lines attached to user objects data terminals
- Test individually

- No symbolic procedure calls
  - No recursion
  - Multiple occurrences = multiple copies

© 1991 Hewlett-Packard Company

User Interaction

Definition – a user is someone who runs a model developed by someone else

- **User Inputs**
- **Customization**
- **Panel Views**
- **Secure Models**
- **Combining Panels and UserObjects**

©1991 Hewlett-Packard Company

User Inputs

- Enum, Toggle, Sliders, Constants, Dialog Box
  - Allow developer to prompt user for a variety of inputs
  - Each input object allows "AUTO EXECUTE"
  - Users input values without having to RE-START the model

©1991 Hewlett-Packard Company

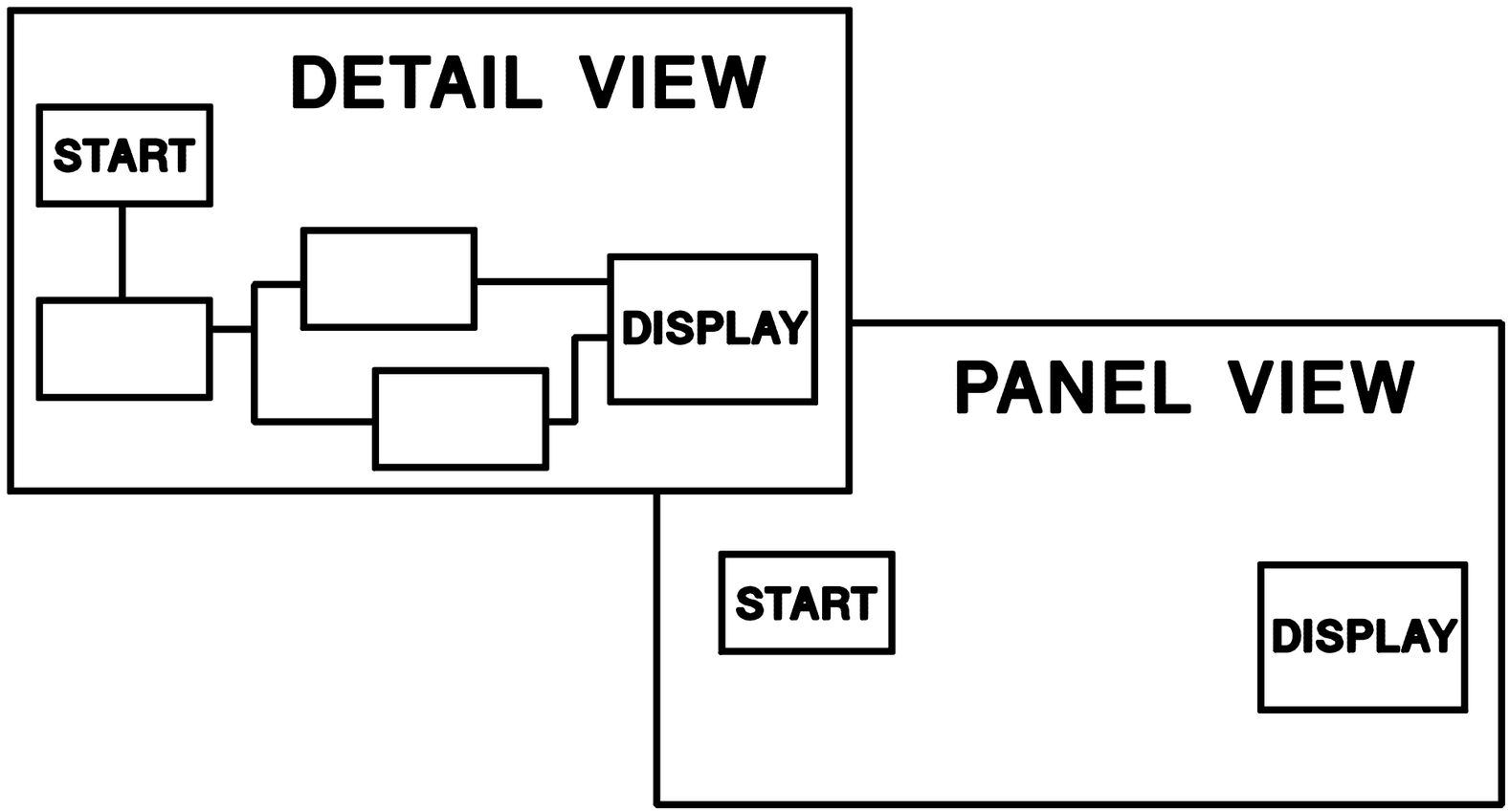
User Customization Features

- Ability to size objects
- Ability to customize display features and colors
- Ability to annotate a model
  - Notepads
  - Custom object or model titles
  - Object descriptions
  - Add/Change Bitmaps

©1991 Hewlett-Packard Company

Panel View

- An alternate view of the model
- Developer chooses objects from the Detail View
- Data and sequence lines are not shown on the Panel View



©1991 Hewlett-Packard Company

Panel Views

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

©1991 Hewlett-Packard Company

Creating a Panel View

1. Build the model and verify that it runs properly
2. Select the one or more objects you want to show on the panel
3. Select Edit – Add to Panel
4. Move and size objects on the panel to maximize its effectiveness
5. Press Panel and Detail to move between views

©1991 Hewlett-Packard Company

Panel View Characteristics

- Fewer choices appear on the main menu in panel view
- If you 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
  - Number Formats
  - Scaling
  - Etc.

©1991 Hewlett-Packard Company

Securing a Panel View

- Creates a panel that does not allow a user to access the detail view
- 3 Step Process:
  1. Create the model and the panel view;
  2. Select Secure, and save the source file
     Both detail and panel views are available, yet the panel view can no longer be edited
  3. To remove access to the detail view, go to the panel view and save the model
     Be certain to select a unique name so that you don't overwrite the source file

©1991 Hewlett-Packard Company

UserObjects With Panel Views

- A UserObject is an independent work area within an object
- The UserObject allows developers to create a panel view within the object
- Select objects within the UserObject and use the object menu – Edit to add them to a panel

©1991 Hewlett-Packard Company

Using "Show Panel on Exec"

- Create a UserObject with a panel view
- Select the object menu, and select Show Panel on Exec.
- When the UserObject operates, the panel "Pops Up" on the work area

©1991 Hewlett-Packard Company

Show Panel on Exec

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

©1991 Hewlett-Packard Company

Application Development: Building Complex Models Visually

Benefits of Using HP VEE

- Time spent solving the problem – no time spent remembering syntax
- Development time is decreased
  - No edit, compile cycle
  - Changes made quickly
- Multifunctionality of objects based on data types and shapes
- Inherent user interface
  - Visual orientation
- Automatic Data Typing

© 1991 Hewlett-Packard Company

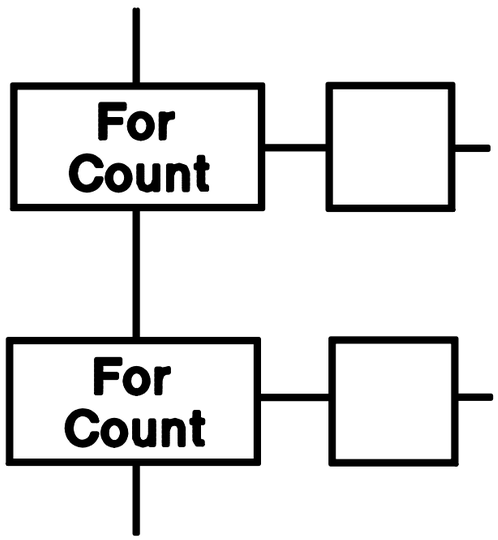
The Visual Engineering Environment: Paradigm Shift

- The same, only different: programming languages
  - Highly cognitive, visually based, less abstract, more conceptual
  - One picture is worth a thousand lines (of code)
- Data flow easy here, control flow takes some thinking
- Remember block diagrams, sketches, flow charts
- In the end, hands-on experience facilitates the paradigm shift

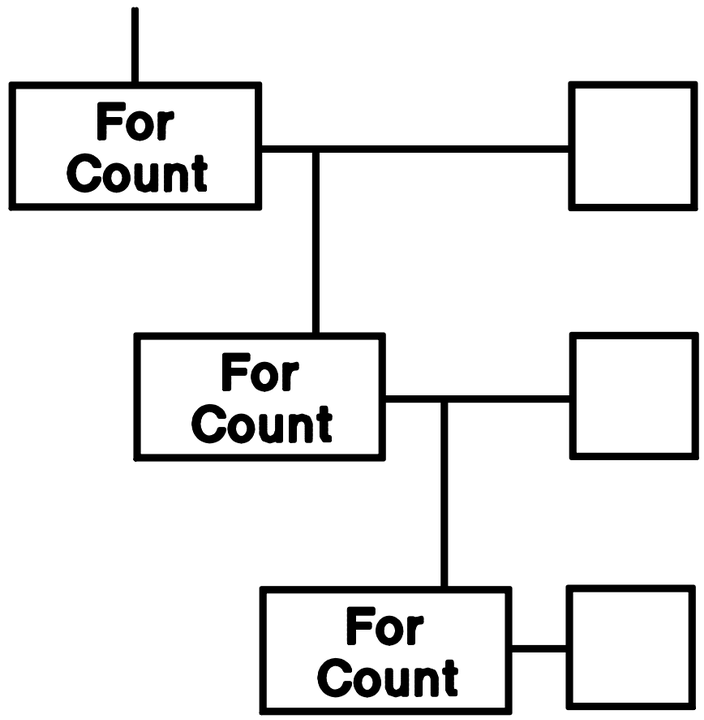
© 1991 Hewlett-Packard Company

Think Visually, Spatially: Structures

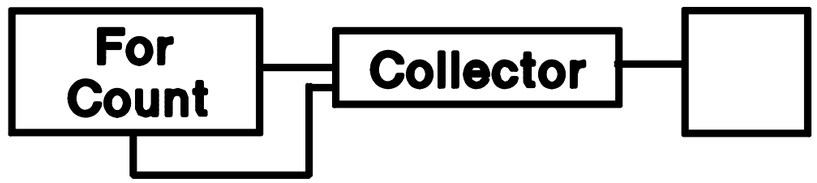
- Subthread basic unit of work
  - Applicable to functionality of UserObject
- Structures of subthreads



Sequential Loops



Nested Loops



Loop Terminating into Sequential Flow

© 1991 Hewlett-Packard Company

Top Down Design

- Define the problem and its constraints
- Identify and define logical order and sequence
- Define subtasks
- Further define each subtask into manageable units
- Implement units
  - UserObjects
- Structured programming
  - Exactly same principles apply as in languages

© 1991 Hewlett-Packard Company

Complexities

- User interfaces
  - Like conventional programming, can be complex
  - Use of UserObjects, panel views, Show on Execute
  - Make the distinction between execution of user input and execution of the algorithms
- Optimization
  - Features of data, displays, number crunching
  - HP-UX escape, named pipes

© 1991 Hewlett-Packard Company

Levels of Complexity

- Visual Calculator
  - Simple, straight forward
  - More data flow, less control
  - No programmatic change
- Applications
  - Complexity equal to that of 1,000 lines or more programs
  - Lots of control flow: conditionals, programmatic change
  - Robustness: dealing with a user
  - Error handling
  - Different from test & measurement: Basic
  - Lots of subtasks

© 1991 Hewlett-Packard Company

Beginnings and Endings

- Run vs. Start vs. Auto Execute
  - All not necessarily the same for given model
- User interaction
  - Fill-in, then execute
  - Execution and data input
  - The toggle object
- Control of execution
  - The OK object
- When is the model terminated
  - Implicit vs. explicit

© 1991 Hewlett-Packard Company

The Paradigm: An Example, The Fibonacci Sequence

- Current calculation needs the two previous results
  - The old way; variables and "i–1, i–2"
  - The new way the feed back loop and the shift register object
- The spatial orientation of objects
  - Left to right; data–in data–out; up thread down thread
- Feedback loops: down thread brought back up thread
  - A visual structure easily recognizable
  - Will only find in subthreads driven by iterators
- The shift register paradox
  - Thinking "n–1" will cause trouble

© 1991 Hewlett–Packard Company

I/O Transactions and Data Formatting

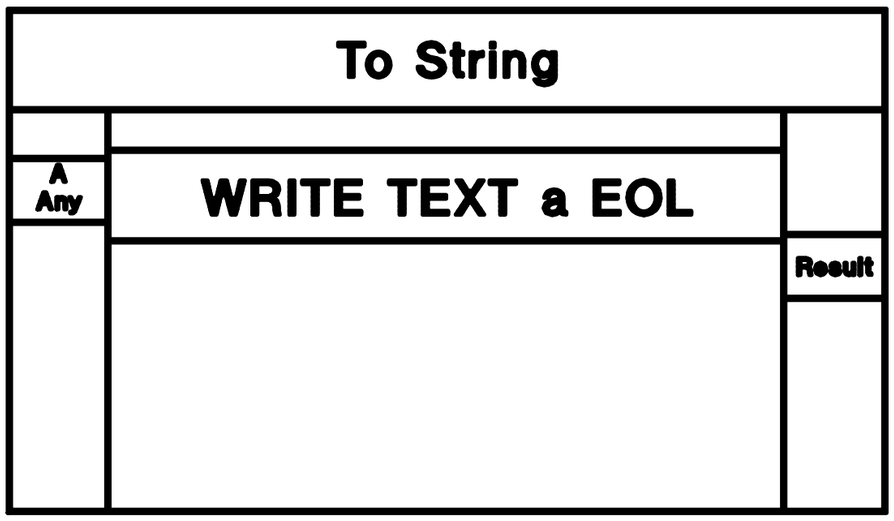
Purpose – To provide communication paths to:

- **"Standard I/O" facilities**
- **The file system**
- **Line printer spooler**
- **Strings**

©1991 Hewlett-Packard Company

I/O Transactions

- All communication paths are implemented as Transaction Objects
- Individual transactions handle multiple data items



©1991 Hewlett-Packard Company

Transactions

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

©1991 Hewlett-Packard Company

Actions

- **READ** – Transfer data from the external data source to the output pin
- **WRITE** – Transfer data from input pin to external data destination
- **EXECUTE** – Cause a device dependent action
  - REWIND for files
  - TRIGGER for HP-IB (HP VEE-Test only)
- **WAIT** – Suspend processing of transactions
  - FOR INTERVAL – inserts a time delay
  - UNTIL SPOLL MASK – waits for specified condition on HP-IB (HP VEE-Test only)
- **SEND** – Write low level HP-IB control/data sequences (HP VEE-Test only)

©1991 Hewlett-Packard Company

Data Encoding

- **TEXT** – Data stream consists of ASCII character streams
  – Data types are constructed character-by-character
  ex: "1.23456" EOL ⟶ REAL value 1.23456
- **BINARY** – Data stream consists of bytes which match VEE internal representation
  ex: REAL value ⟶ 64-bit IEEE format (8 bytes)
- **BYTE** – Data stream consists of 1 byte/variable
- **CASE** – Behaves like an enumerated type
  ex: CASE x OF "Zero", "One", "Two" will select string "Two" if x=2
  – Write only

©1991 Hewlett-Packard Company

Data Encoding

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

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

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

©1991 Hewlett-Packard Company

Data Encoding

- CONTAINER – Data stream is sent in HP VEE descriptive format

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


©1991 Hewlett-Packard Company

TEXT Formats for WRITE Action

- Used to "beautify" output
- Little type checking or conversion performed
- DEFAULT – All data in free-field notation
  - All characters of string data
  - All significant digits of numeric data
- STRING – Same as DEFAULT, except control of field width, justification
- QUOTED STRING – Same as STRING, but each data item in double quotes
  - Embedded quotes
- REAL – Same as STRING, except control of sign prefix, FIXED, STANDARD, or SCIENTIFIC, significant digits
- COMPLEX – Same as two REALS separated by commas, enclosed in parentheses
- PCOMPLEX – Same as COMPLEX, except angle value preceded by "@"

©1991 Hewlett-Packard Company

TEXT Formats for READ Action

- Match input data stream to required values 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

©1991 Hewlett-Packard Company

The Number Builder

- 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

©1991 Hewlett-Packard Company

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

©1991 Hewlett-Packard Company

Communication with File System

Purpose:

- **Storing and retrieving data from other programs**
- **"Permanent" data archival**
- **Simple communication with other processes**

©1991 Hewlett-Packard Company

About Files

HP-UX files are:

- Typeless - all data formatting done by application
- Sequential Access - no random access to file contents
- Buffered - HP-UX maintains many buffers for performance
- Extensible - file grows as required to accomodate data

©1991 Hewlett-Packard Company

Using Files

- Opening file occurs once per direction (READ/WRITE)
  - First transaction after pre-run
  - File closed upon model termination
- Existing file can be overwritten or have data appended
- To File and From File maintain separate file pointers
  - All To File To Same File share one pointer
  - All From File To Same File share one pointer
  - REWIND in From File does not affect To File

©1991 Hewlett-Packard Company

File I/O Transactions

- To File and From File support two EXECUTE commands
  - REWIND – All further READ or WRITE operations start at beginning of file
    – Cannot use in To File open in APPEND mode
  - CLEAR – Useful only in To File in OVERWRITE mode
    – Resets file to zero length (erases old data)

©1991 Hewlett-Packard Company

HP-UX Standard I/O

- HP-UX associates 3 communication paths per process
  - Standard Input ("stdin")
    - Normally the keyboard
  - Standard Output ("stdout")
    - Normally the display
  - Standard Error ("stderr")
    - Normally the display

©1991 Hewlett-Packard Company

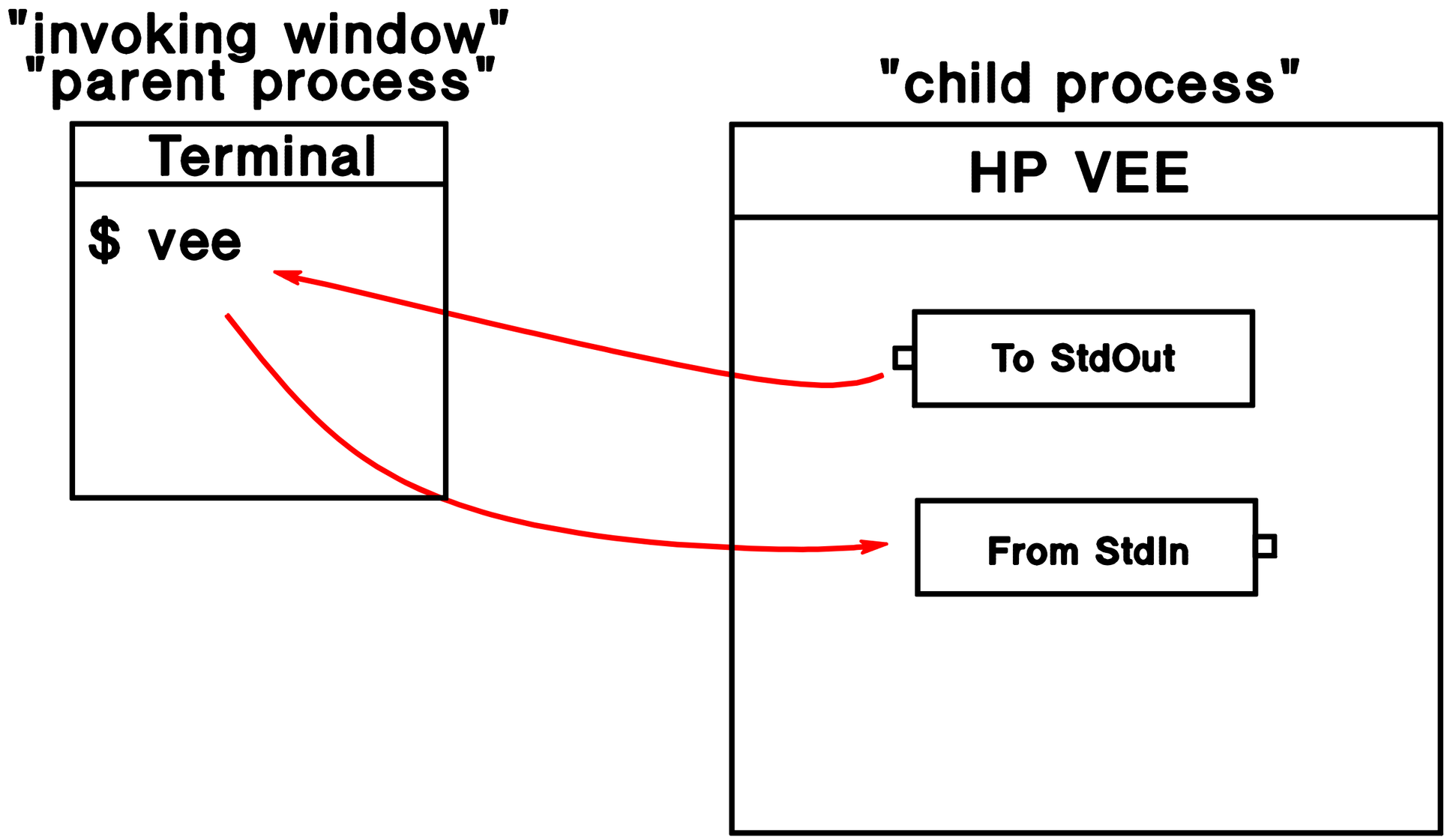
HP-UX Standard I/O

- Shells attach standard I/O ("stdio") to child process
- Stdio can be redirected
  - /bin/cat <file1 >file2 2>errs
- Stdio is ALWAYS buffered
  - No character-by-character access by READ - must have EOL

©1991 Hewlett-Packard Company

Communication via Standard I/O

- When VEE is invoked from terminal window, stdio is passed to VEE



©1991 Hewlett-Packard Company

Using Standard I/O

- Useful for prototyping
  - Program output appears in invoking window
  - Input can be supplied by keyboard entry
  - Can regain control of "hung" VEE with control-C
- Allows VEE to be invoked by another program

```
vee -r veeprogram  < datafile | sort | more
```


©1991 Hewlett-Packard Company

HP-UX Escape

- 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 single HP-UX Escape
  - Similar to To/From Stdio
  - HP-UX Escape is child process of HP VEE
  - Child receives data via its stdin, sends data via stdout and stderr

©1991 Hewlett-Packard Company

Using Shell

Process can be invoked directly, or a shell can act as intermediary

- Advantages: Interpret metacharacters ("*", etc.)
  Set up pipes, stdio redirection
- Disadvantages: Increased overhead (number of processes)
  Increased startup time (due to reading .kshrc, .profile, .cshrc)

©1991 Hewlett-Packard Company

Wait for Child Exit

- YES:
  - New process starts whenever HP-UX Escape activates
  - VEE executes transactions, sends EOF (by closing pipe)
  - VEE waits for process termination
  - Program MUST terminate!!
- NO:
  - Process is allowed to remain active after HP-UX Escape completes
  - Repeated HP-UX Escapes do not need to restart process
  - Process must be designed to cooperate with HP-UX Escape
    - Continuous loop
    - No unexpected terminations
  - Process will be restarted as needed after Pre-Run

©1991 Hewlett-Packard Company

Pitfalls in Escape I/O

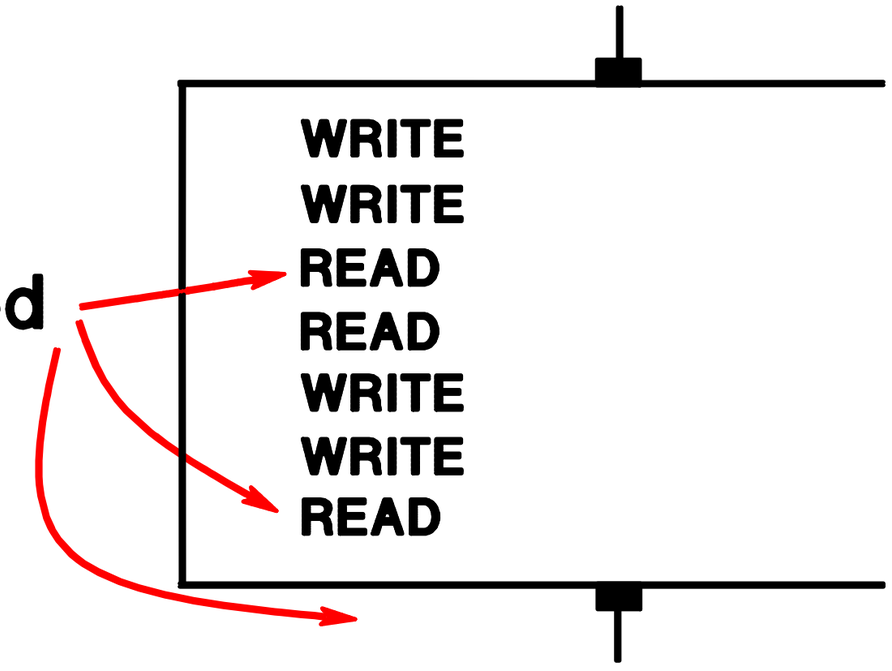
- Buffering must be disabled or buffers flushed
  - READ transaction may hang
  - Shell scripts cannot easily comply
  - C programs do:

    setbuf (stdout, NULL);

    OR

    fflush (stdout)
- VEE will flush its WRITE buffers
  - Before READ
  - After last transaction

Pipes flushed here



©1991 Hewlett-Packard Company

System Requirements

- HP VEE-Test - HP-UX application that requires 12MB file space and uses 5MB RAM (HP VEE-Engine uses 8 MB file space)
- SYSTEM - HP-UX 7.0 or later X11.4 windows

Recommended

345, 360, 375, 380, 400 (DIO)...

16 MBytes RAM

300 MByte Hard Disk

6 Plane, 1024x768, 1280x1024

Graphics

Remember....

RMB/UX, Multi-Tasking, VUE, Etc. will affect requirements!

©1991 Hewlett-Packard Company

Installation

- Software shipped on
  - 1/4" cartridge tape
  - CD-ROM

- installation is performed by /etc/update
  - HP VEE-Test filesets
    - VEE_TMAIN
    - VEE_TIDS
    - VEE_THELP
  - HP VEE-Engine filesets
    - VEE_EMAIN
    - VEE_EHELP

- HP VEE-Test requires device files to be created for each interface
  - run /usr/lib/veetest/vee_config on each system (cnode)

©1991 Hewlett-Packard Company

File System

- /usr/lib/veetest/

veetest	the actual executable (or veeengine)
hpidc	id compiler program
vee_config	program to set your /dev files
pcltrans	program needed for printing
xwd2sb	program needed for printing
./instruments	directory with .cid files *HP VEE-Test ONLY
./help	directory with help files
./examples	example HP VEE models
./config	directory of default config files
./lib	library of useful objects

- /usr/bin/veetest — symbolic link to run the /usr/lib/veetest/veetest executable

©1991 Hewlett-Packard Company

HP VEE Files that Assist With Customization

- .veeio
  - Config I/O file
  - Should be stored in default user directory
  - HP VEE includes a default version (d.veeio) at /usr/lib/vee/config/
- .Xdefaults

  File- to customize colors, fonts, paletes, etc.
  - HP VEE includes several default versions at /usr/lib/vee/config/
- .veerc
  - Contains preferences for trig mode, auto line routing, printer configuration,waveform setting, number format

©1991 Hewlett-Packard Company

HP-UX Configuration

- HP VEE imposes no special requirements on kernel configuration, except:
  - X11 must be installed
  - hpib, gpio, and serial drivers are needed for HP VEE-Test I/O
  - each HP VEE process requires >6MB swap space available

©1991 Hewlett-Packard Company

Invoking HP VEE

- veetest [options]
  veeengine [options]
- options:

-name NAME	: changes name from Vee to NAME for determining X11 attribute lookup
-help	: print all command-line options
FILE	: load FILE at startup
-r FILE	: load and run FILE at startup
-d DIR	: specify DIR as install directory (defaults: /usr/lib/veetest or /usr/lib/veeengine)
-iconic	: specify startup as icon, not open window view
-geometry 10WxH+X+Y	: specify windows geometry

©1991 Hewlett-Packard Company

Other Customizing Files

$HOME/.Xdefaults

/usr/lib/X11/app-defaults/Vee

- contain X customization strings for colors and fonts

Examples:

Mwm*Vee*clientDecoration : none

Vee*geometry : 1024X768+0+0

- creates full-size windows on 1024 X 768 display

©1991 Hewlett-Packard Company

Configuration and Customization

- **Printers**
  - Current screen or entire workspace
  - Spool destination, resolution, dot expansion, color, orientation
- **Global Preferences**
  - Trig mode
  - Auto line routing
  - Number formats
  - Waveform settings

©1991 Hewlett-Packard Company

- ☐ Review IEEE 488 Standard
- ☐ Review IEEE 488.2 Standard

© 1991 Hewlett-Packard Company

Objectives of 488

- Define a general purpose, limited distance system
- Specify device-independent mechanical, electrical and functional interface
- Specify terminology and definitions
- Enable interconnections of different manufacturers equipment
- Permit direct communication without routing messages through a control unit
- Define system with minimum restrictions on performance characteristics
- Define asynchronous communications system with wide range of data rates
- Define a low cost system

© 1991 Hewlett-Packard Company

IEEE 488.1 Bus Lines

- Data Lines
  - DIO1
  - DIO2
  - DIO3
  - DIO4
  - DIO5
  - DIO6
  - DIO7
  - DIO8
- Bus Management Lines
  - ATN
  - IFC
  - REN
  - SRQ
  - EOI
- Byte Transfer Lines
  - DAV
  - NRFD
  - NDAC

© 1991 Hewlett-Packard Company

IEEE 488 Key Specifications

- 15 Devices max
- Star or linear interconnection
- 16 Signal lines
- Byte serial, bit parallel messages
- 2 Metres per device, 20 Metres Total
- 1 MByte/sec maximum data rate
- 31 Primary addresses (992 secondary addresses)
- Multiple controllers (pass control)
- Hardware interface circuits (TTL, Schottky, Tristate)

© 1991 Hewlett-Packard Company

IEEE 488 Bus Device Functions

- Listener (receives data)
- Talker (sends data)
- Controller (assigned talkers and listeners)
- System controller (clear bus, put devices in remote mode)

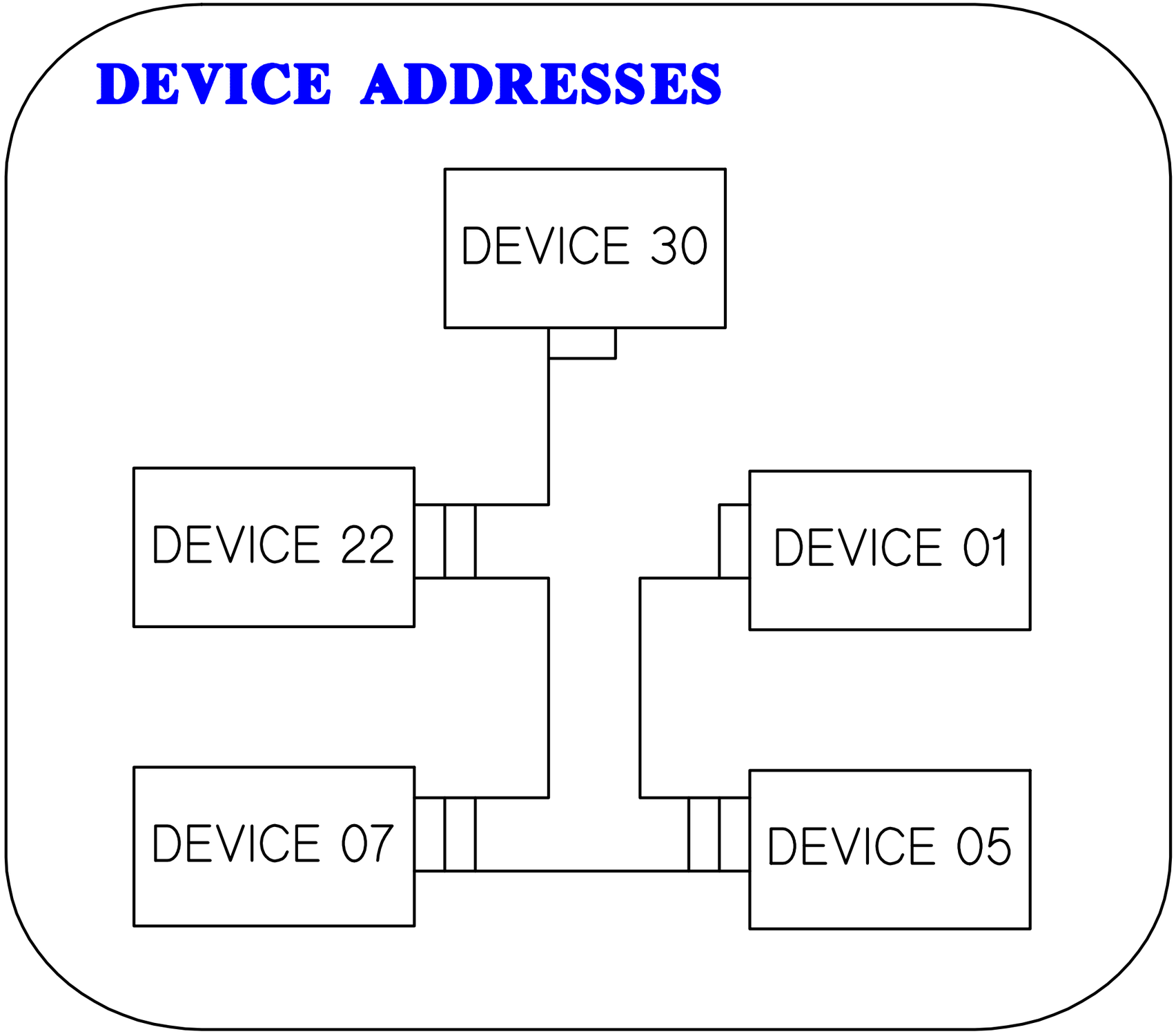
© 1991 Hewlett-Packard Company

IEEE 488 Device Capability Subsets

SH	Source Handshake
AH	Acceptor Handshake
T	Talker
L	Listener
SR	Service Request
RL	Remote Local
PP	Parallel Poll
DC	Device Clear
DT	Device Trigger
C	Controller
E	Driver Electronics

© 1991 Hewlett-Packard Company

DEVICE ADDRESSES



©1991 Hewlett-Packard Company

IEEE 488

IT DOES NOT DEFINE:

- Status reporting
- Data format
- Data coding
- Minimum capabilities
- Message protocol

©1991 Hewlett-Packard Company

IEEE 728 Recommended Practice for Code and Format Conventions

DEFINED:

- Data messages
  measurement, program, status, display
- Separators
- Headers

- Loosely defined with too many choices
- Wide open to "personal interpretation"

© 1991 Hewlett-Packard Company

Device Dependent Commands
What Does DCL
(Device Clear) Do?

- "Return to a pre-defined, device-dependent state"
- Some clear I/O buffers
- Some change the function state of the device
- Some do a complete device reset

©1991 Hewlett-Packard Company

Problems Encountered Using IEEE 488

- Minimum capability set
- Data format
- Data codes
- Message protocol
- Commands
- Status

©1991 Hewlett-Packard Company

IEEE Solution

- IEEE 488 renamed 488.1
- IEEE 728 obsoleted
- IEEE 488.2 defined

©1991 Hewlett-Packard Company

IEEE 488.2

- Required IEEE 488.1 Capabilities
  (Everyone can talk, listen and be serial polled)

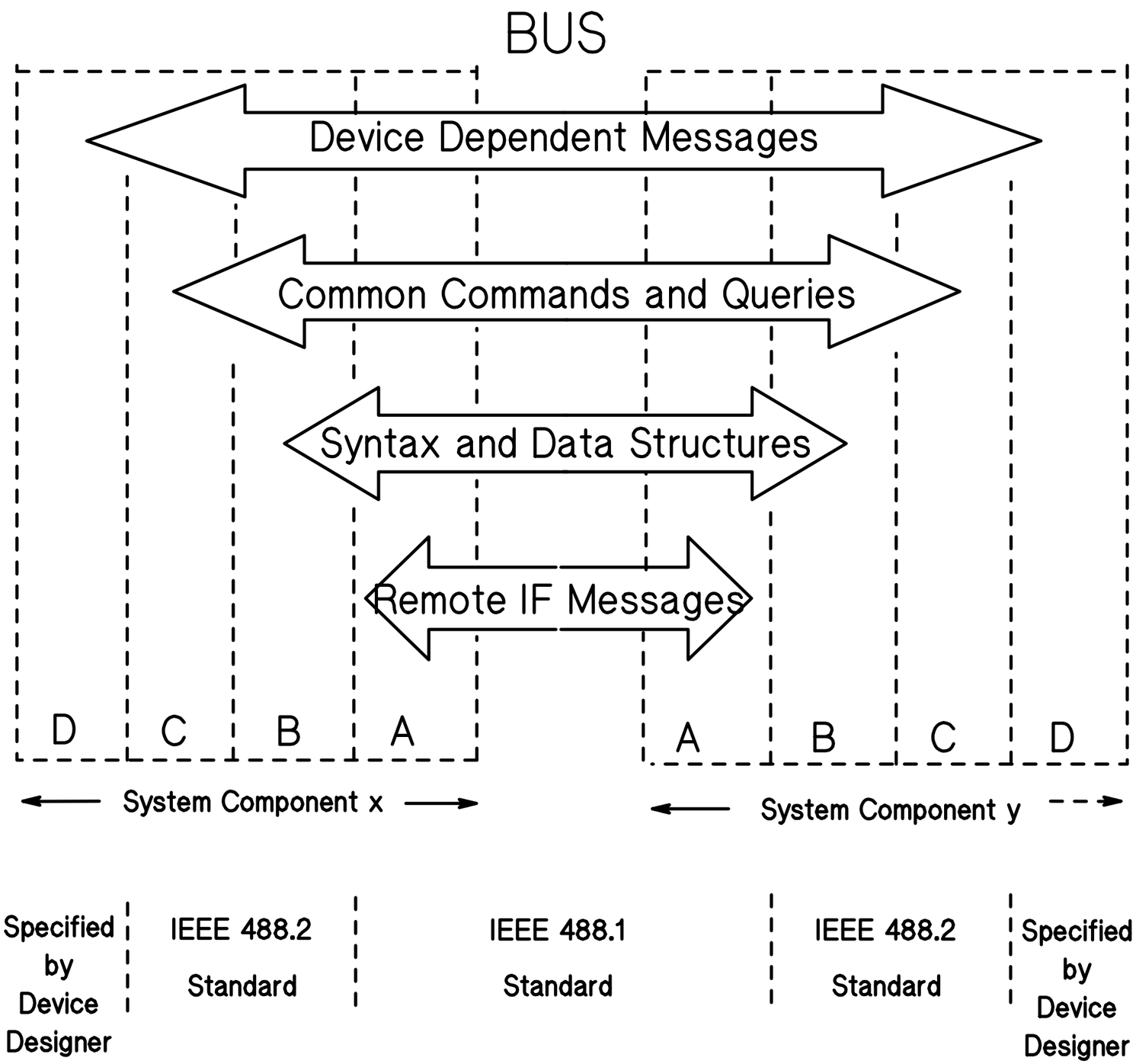
- Data Formats
  (e.g. Numbers all look the same)

- Message Protocol
  (Bus hangups are minimized)

- Common Commands
  (e.g. *IDN? Identifies the instrument)

- Status Model
  (Status byte usuage is consistent)

©1991 Hewlett-Packard Company



A: Interface Functions
B: Message Communication Function
C: Common System Functions
D: Device Functions

©1991 Hewlett-Packard Company

DEVICE COMPLIANCE CRITERIA

- IEEE 488.1 requirements
- Message exchange requirements
- Syntax requirements
- Status reporting requirements
- Common commands
- Synchronization requirements
- System configuration requirements
- Controller requirements
- Documentation requirements

©1991 Hewlett-Packard Company

IEEE 488.2 DATA CODES

- ASCII-Bit Code
- 8-Bit Binary Integer
- Binary Floating Point

©1991 Hewlett-Packard Company

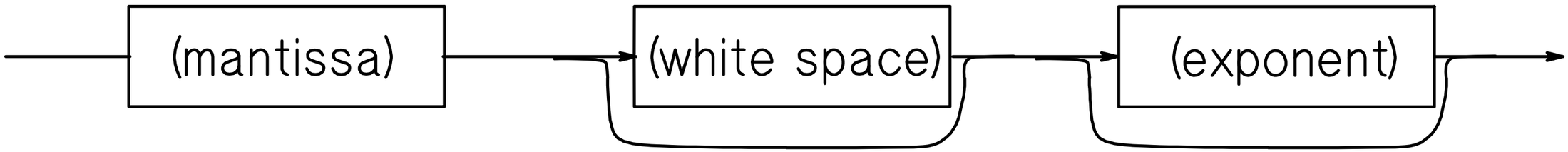
SYNTAX REQUIREMENTS

"Forgiving Listening, Precise Talking"

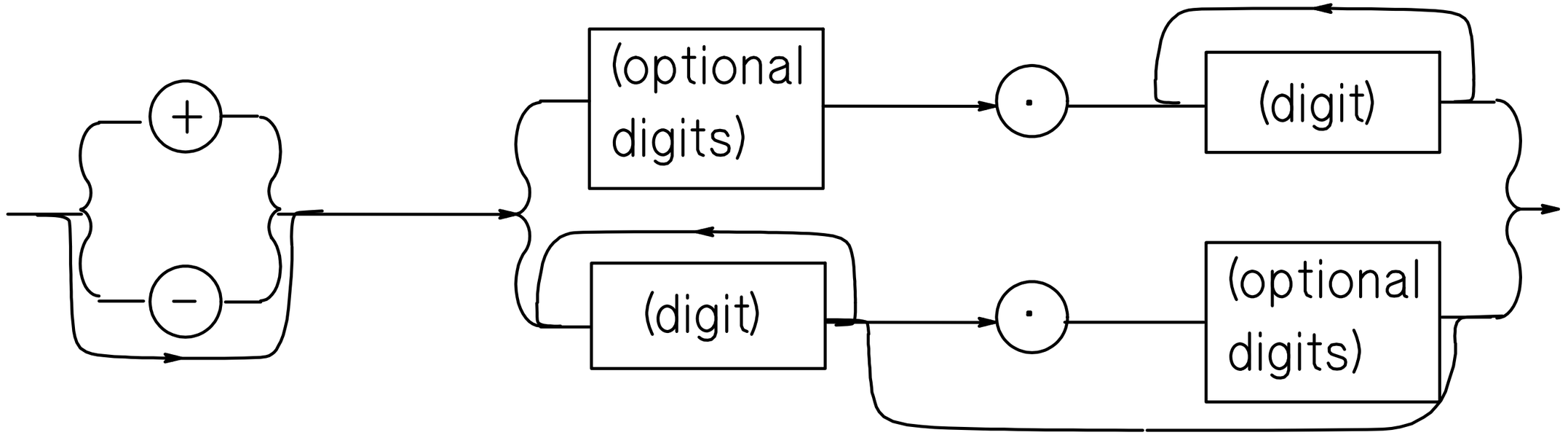
Device Listening Functional Elements

Device Talking Functional Elements

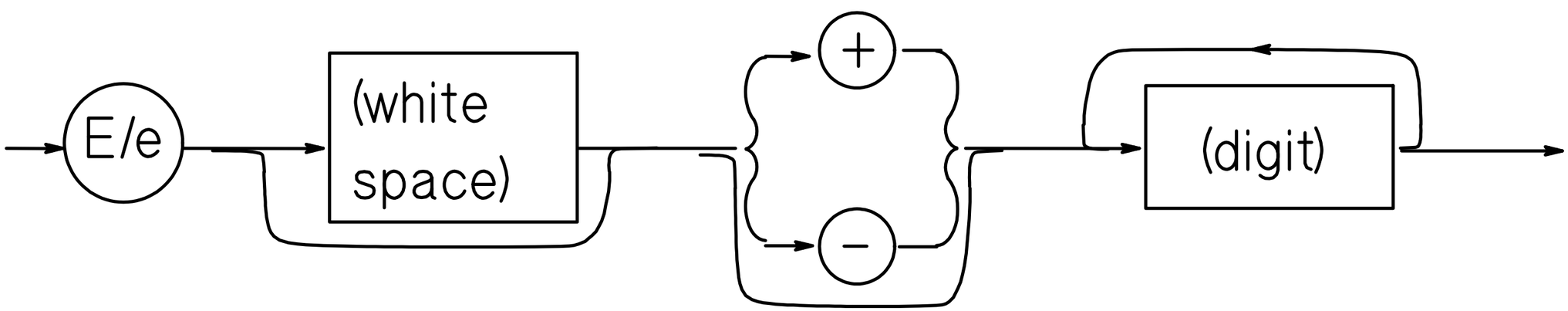
©1991 Hewlett-Packard Company



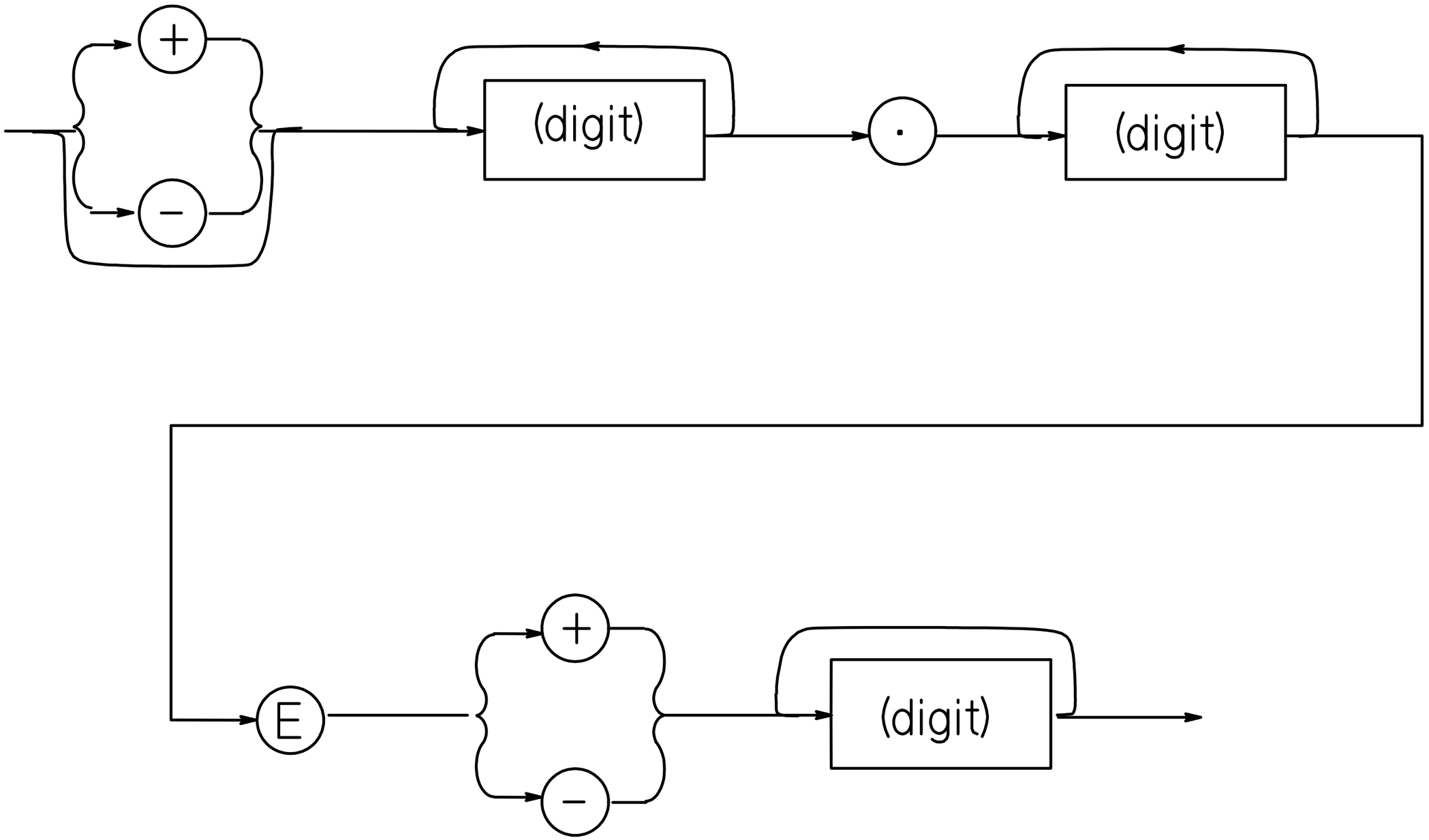
(mantissa) is defined as



(exponent) is defined as



©1991 Hewlett-Packard Company



©1991 Hewlett-Packard Company

STATUS REPORTING MODEL

- IEEE 488 Defined:

  Status byte and service request bit

©1991 Hewlett-Packard Company

STATUS REPORTING MODEL

- IEEE 488.2 Defines:
  - Master summary status bit
  - Event status bit
  - Message available bit

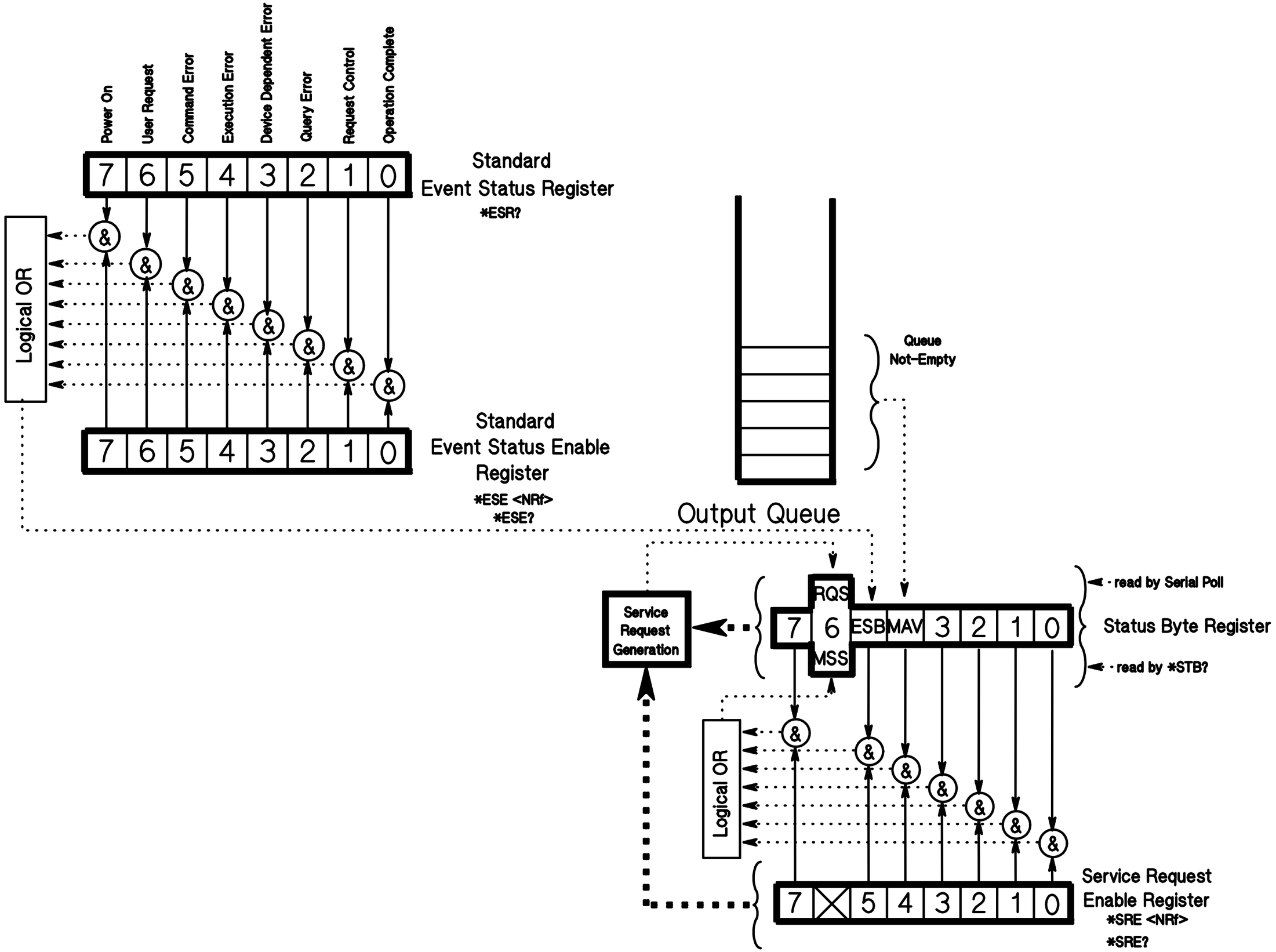
- Standard Event Status Register

Power on	Device dependent error
User request	Query error
Command error	Request control
Execution error	Operation complete

Standard Event Status Enable Register

Output Queue

©1991 Hewlett-Packard Company



©1991 Hewlett-Packard Company

IEEE 488.2
COMMON COMMANDS

Defined using ASCII characters

Preceeded by an "*"

Queries end with "?"

©1991 Hewlett-Packard Company

IEEE 488.2 REQUIRED COMMANDS

(ATN FALSE)

SYSTEM DATA

*IDN	Identification

INTERNAL OPERATIONS

*RST	Reset
*TST?	Self-Test Query

SYNCHRONIZATION

*OPC	Operation complete
*OPC?	Operations complete query
*WAI	Wait to continue

©1991 Hewlett-Packard Company

IEEE 488.2 REQUIRED COMMANDS

(ATN FALSE)

STATUS & EVENT

*CLS	Clear status
*ESE	Event status enable
*ESE?	Event status enable query
*ESR?	Event status register query
*SRE	Service request enable
*SRE?	Service request enable query
*STB?	Read status byte query

©1991 Hewlett-Packard Company

IEEE 488.2

- Required IEEE 488.1 Capabilities
  (Everyone can talk, listen and be serial polled)

- Data Formats
  (e.g. Numbers all look the same)

- Message Protocol
  (Bus hangups are minimized)

- Common Commands
  (e.g. *IDN? Identifies the instrument)

- Status Model
  (Status byte usuage is consistent)

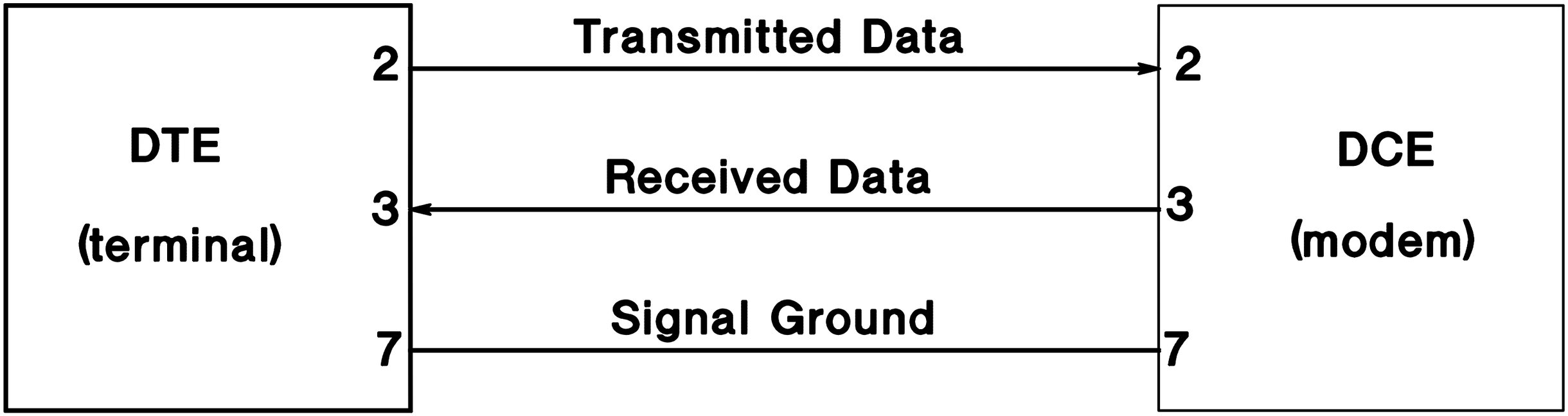
©1991 Hewlett-Packard Company

EIA DEFINED:

- Mechanical Characteristics
- Electrical Characteristics
- Interchange Circuits & Functions
- Relationship to Standard Interface Types
- Similar to CCITT V.24 & V.28

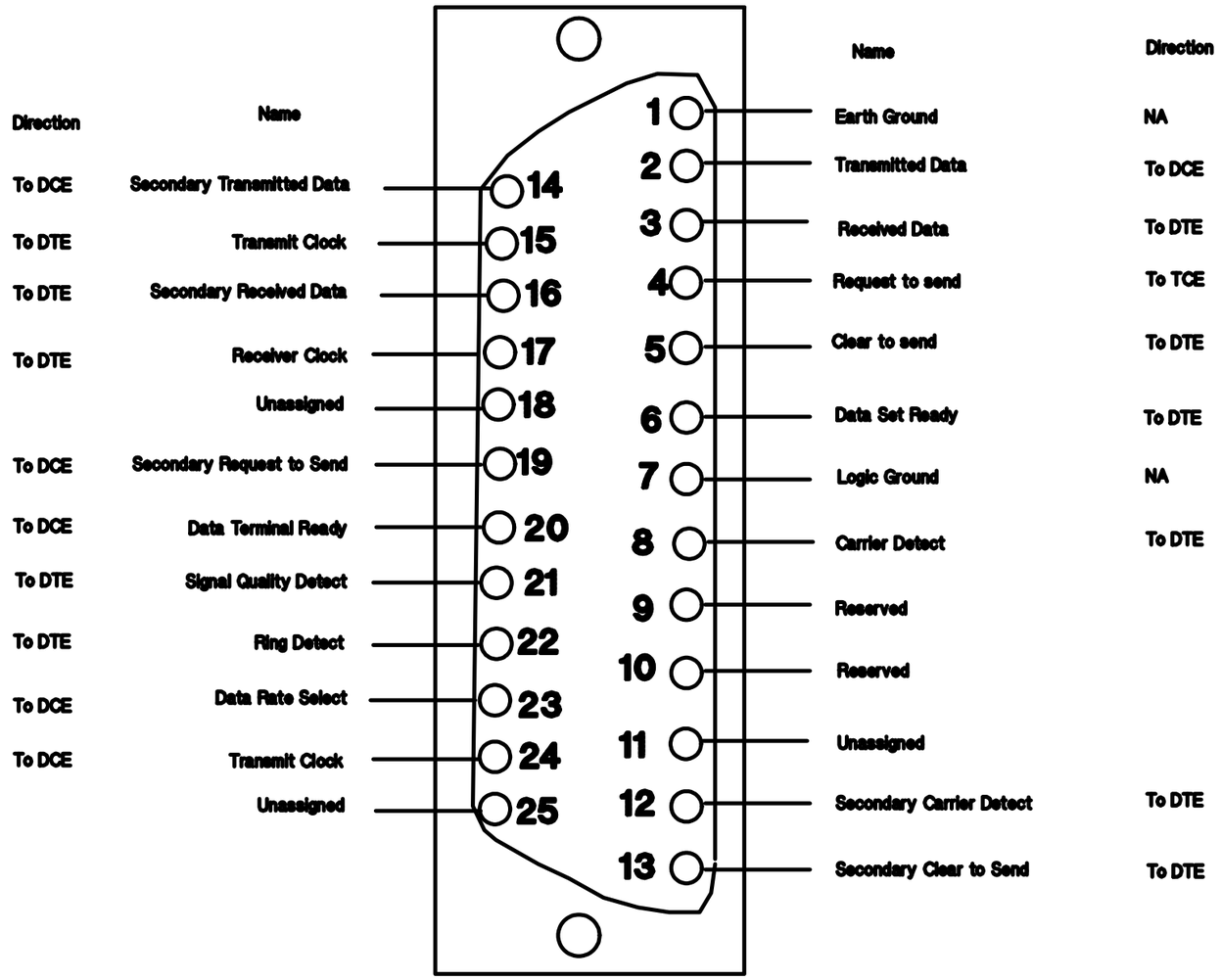
©1991 Hewlett-Packard Company

RS 232

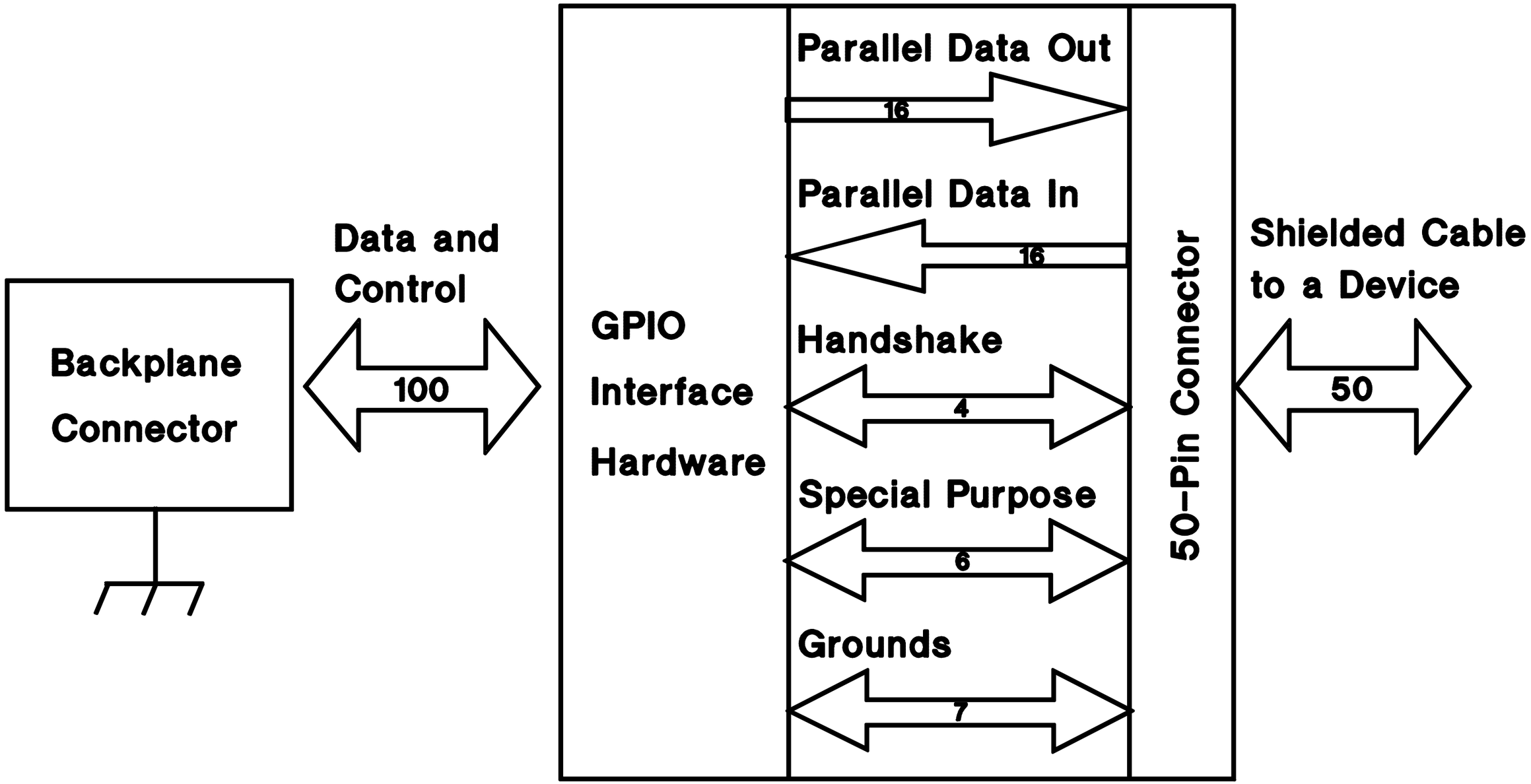


©1991 Hewlett-Packard Company

RS 232 Pin Assignments



©1991 Hewlett-Packard Company



Block Diagram of the GPIO Interface

©1991 Hewlett-Packard Company

HP VEE Device I/O

- Instrument "State" Drivers
  - Developed by HP for 170 instruments
  - Easiest HP VEE instrument control
  - Most interactive
- Instrument "Component" Drivers
  - Allow efficient access to instrument driver
- Direct I/O
  - For devices and instruments with no pre-developed drivers
  - Fast, flexible, and powerful
  - Transaction interface consistent with other HP VEE I/O objects

©1991 Hewlett-Packard Company

HP Instrument Drivers

- Used in ITG and HP VEE-Test
- Text file that defines:
  1. Instrument components (or functions)
  2. Bus mnemonics to set components
  3. User interface for front panel interaction
- Also contain function interrelation (coupling)

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

Coupling and the proper order of components allows incremental state programming

©1991 Hewlett-Packard Company

Instrument Drivers

- All drivers are compiled to allow fast loading
- All HP ITG drivers are supported except 3852 (user subs)
- HP VEE drivers default to Incremental On which only works for state programmable (supplied) drivers
  - Incremental Off may work for some "homemade" drivers

©1991 Hewlett-Packard Company

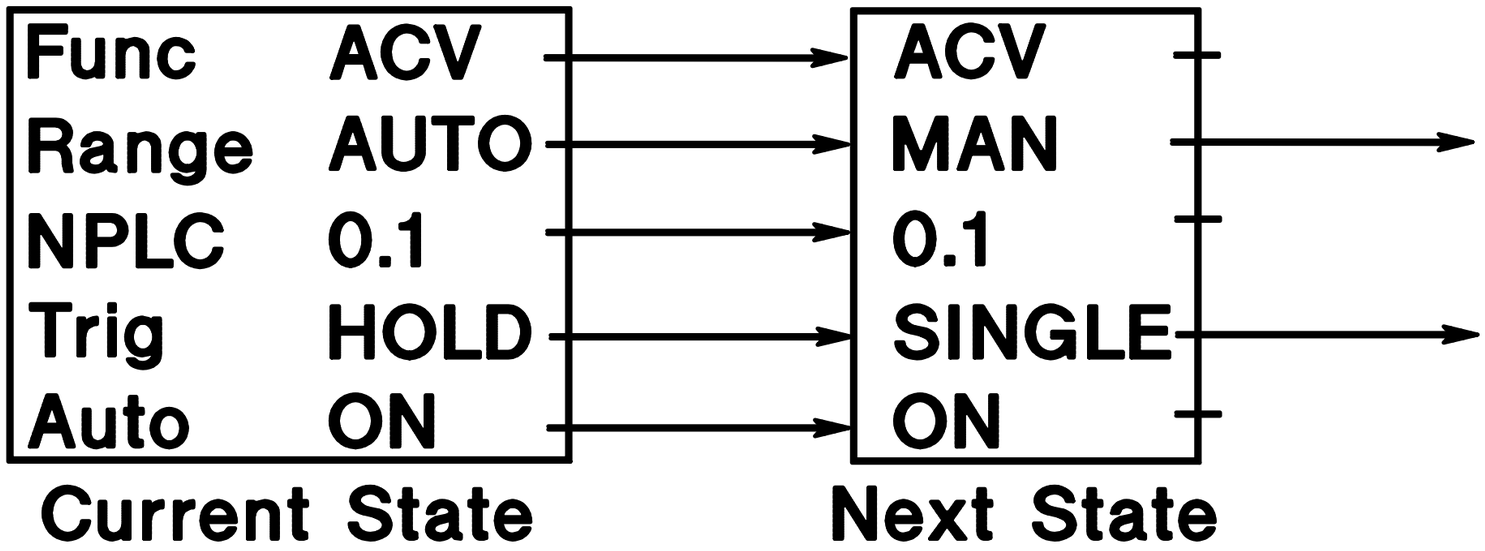
State vs. Component Drivers

- "State" Drivers
  - Show complete graphical panel
  - Use when working with full instrument states
- "Comp" Drivers
  - Don't show full panel
  - Use for setting specific components
  - Efficient, yet still isolate instrument knowledge to driver
  - Only maintain state of specific components

©1991 Hewlett-Packard Company

Incremental State Programming

- The software package 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 are sent to instrument



©1991 Hewlett-Packard Company

State Drivers

- Error Checking ON
  - For instruments that allow it, after an action list is sent, the driver requests the instrument error status
- Error Checking OFF
  - Speeds execution some, but leaves the user exposed to unreported errors

There are better ways to optimize than turning off error checking

©1991 Hewlett-Packard Company

HP VEE Comp Drivers

- Only required functions are added to object
- Only added functions have state maintained
- Because State Lookup is NOT done for every function, comp drivers execute much faster than state drivers
- Assumes incremental OFF, error checking OFF

©1991 Hewlett-Packard Company

HP VEE Instrument Drivers Summary

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

©1991 Hewlett-Packard Company

Using Instrument Drivers

- First step is creating HP-UX device files
  - Performed by vee-config
- Next step is to Configure I/O
  - Allows user to specify which instruments are available, the instrument address, timeout, etc.
  - All instrument configuration must be accessed under Configure I/O
  - Two menus under CONFIG
    1. Direct I/O Configuration
    2. Instrument Driver Configuration

©1991 Hewlett-Packard Company

Using Instrument Drivers
Device Configuration

- **NAME** – Must be unique to this config
  – What will appear on the object title
- **INTERFACE** – Choose interface type – HP-IB, Serial, GPIO
- **ADDRESS** – 0 – if no instrument is present
  – 714 – Bus Address (7) and Instrument Address (14)
- **DEVICE TYPE** – Descriptive name
  – Defaults to Driver File Name
- **TIMEOUT** – Instrument Timeout in seconds
- **LIVE MODE** – ON – Instrument is present
  – OFF – No instrument attached
  (If address is 0, live mode is OFF)

©1991 Hewlett-Packard Company

Using Instrument Drivers
Instrument Driver Configuration

- **ID FILENAME** – Select ID to associate with instrument
- **SUB ADDRESS**
  - Defaults to -1
  - Only required for some instruments
  - Not HP-IB secondary address
  - Slot or card address
- **INCREMENTAL MODE**
  - ON - Only send required commands
  - OFF - Send all commands
- **ERROR CHECKING**
  - ON - Ask the instrument for any errors
  - OFF - Don't check for instrument errors

©1991 Hewlett-Packard Company

Using Instrument Drivers Within Models

- Unconnected ID panel used interactively
- Terminals allow ID functions to be controlled
  - All components known within ID available via data input/output terminals
  - Field input/output allows selection of any feature present on panel

©1991 Hewlett-Packard Company

Direct I/O

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

©1991 Hewlett-Packard Company

Trade Offs With Direct I/O Transactions

- **Benefits**
  - Highest performance I/O
  - Consistent usage with other I/O transactions
  - Use to access instrument functionality unavailable through ID
- **Disadvantages**
  - Requires familiarity with instrument programming
  - Not interactive (no live mode)

©1991 Hewlett-Packard Company

Configuring Direct I/O

- Single instrument has separate configuration for Driver and Direct I/O
- Direct I/O configuration specifies
  - Terminators and EOL sequence
  - Formatting of array data
  - Conformance to IEEE 488 or 488.2
  - Information needed to save and restore instrument learn string

©1991 Hewlett-Packard Company

Transactions for Direct I/O

- **EXECUTE**
  - Sends addressed commands
  - CLEAR, TRIGGER, LOCAL, REMOTE, RESET
- **WAIT UNTIL SPOLL MASK** – Allows SPOLL operation and result compare within an instrument object
- **READ IOSTATUS** – Returns 2-bit value of GPIO, STSØ and STS1
- **WRITE**
  - STATE – Allows uploaded learn string to be written
  - IOCONTROL – Allows control of PCTL, CTLØ, CTL1 lines

©1991 Hewlett-Packard Company

Using Advanced HP-IB Functions

- EXECUTE - Sends non-addressed (global) bus commands
  - ABORT, CLEAR, TRIGGER, REMOTE, LOCAL, LOCAL LOCKOUT
- SEND - Allows custom command/data transactions to be created
  - COMMAND - Data sent with ATN TRUE (command)
  - MESSAGE

    IEEE-488 defined mnemonic commands
    Sent with ATN TRUE
    DCL, TCT, etc.
  - DATA - Data sent with ATN FALSE (data)
- TALK, LISTEN, UNLISTEN, UNTALK, MTA, MLA, SECONDARY

©1991 Hewlett-Packard Company

Using Advanced HP-IB Functions

- **HP-IB Serial Poll** – Command sequence initiated by controller which causes addressed instrument to return status byte
- **Wait For SRQ**
  - Suspends operation of current thread until SRQ line is asserted
  - SRQ shared by all instruments on bus
  - Must usually follow with SPOLL to determine source
  - Operation of independent threads continues

©1991 Hewlett-Packard Company

Maintaining Instrument State

- Direct I/O instrument objects can upload the state (learn string) of instruments
  - State is maintained with the object
    Multiple objects share same learn string
  - Does not interact with driver state
- Typical use:
  - Set up instrument with State Driver or front panel
  - Upload learn string
  - Use learn string to preset instrument state
  - Use Direct I/O to make incremental changes

©1991 Hewlett-Packard Company

Bus Monitor

- Works with HP-IB, GPIO, RS-232
- Records all traffic
  - Generated by Driver or Direct I/O
  - Received by controller
- Data is timestamped, displayed in text or hex, I/O direction indicated, and command bytes interpreted

©1991 Hewlett-Packard Company

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
  - Operating system facilities used instead of being reinvented
    - Data buffering
    - Process priority
    - Virtual memory
    - Concurrency of operation
- Benefit: Complex systems built from less-complex modules
  Less coupling means easier maintenance

©1991 Hewlett-Packard Company

Why IPC?

- Data exchange or sharing between processes
- Synchronization of concurrent processes
- IPC trades added complexity of data handling for reduced complexity of program structure
  - Synchronizing data exchange between programs easier than handling asynchronous control events in single process

©1991 Hewlett-Packard Company

IPC Facilities

- HP VEE implements HP-UX file system IPC only
  - Ordinary files
  - Pipes
- Other methods are very specialized - Access via HP-UX Escape object if required
  - Shared memory
  - Semaphores
  - Message queue
  - Signals

©1991 Hewlett-Packard Company

Using Files for IPC

- "Unlimited" capacity
- Unlimited number of processes may access
- Programs must agree on arbitrary conventions for format and synchronization
  - Auxiliary files often used as lock files
- Excellent performance if reader and writer can share file buffer
  - Lightly loaded system
  - Moderate amounts of data
- Poor performance if physical file system involved

©1991 Hewlett-Packard Company

Using Pipes for IPC

- Pipes enforce FIFO message order
  - Multiple processes may write or read
  - Data can be read once ONLY
- Named pipes are created and accessed as part of file system
  - mknod mypipe p (sysadmin only)
  - mkfifo mypipe (user)
  - Exist independently of any processes
- Arbitration for multiple readers on pipe
- Pipes must exist locally (not NFS mounted)

©1991 Hewlett-Packard Company

Using Named Pipes

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

©1991 Hewlett-Packard Company

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 model
  - Not after each object deactivates
  - Never deleted
- Pipes opened as "blocking"
  - Needed for synchronization
  - Can hang waiting for data or space available

©1991 Hewlett-Packard Company

Effective Use of Named Pipes for IPC

- User can initiate separately
  - Reliable user required!
- HP VEE can use HP-UX Escape
  - No wait for child exit
- Other process can invoke HP VEE

©1991 Hewlett-Packard Company

Requirements for Cooperative IPC

The Non-VEE Process Should:

- Open pipes read-only or write-only
- Look for EOF conditions on each read
- Trap SIGPIPE to help diagnose mysterious failures
  - Issued by kernel if write attempted after reader closes
- Use UNbuffered write operations, or flush buffers prior to any READ after WRITE
- Use single reader/single writer model
  - Ideally, interleave read/write operation

©1991 Hewlett-Packard Company

Escapes to HP BASIC/UX

- Performed as two steps to avoid multiple BASIC bootup
  - Initialize HP BASIC/UX
    - Invokes BASIC
    - Loads and runs requested program
  - To/From BASIC/UX
    - Equivalent to To/From Named Pipe

©1991 Hewlett-Packard Company

Effective Use of To/From BASIC/UX

- BASIC/UX ASSIGNS all I/O paths to have read-write capability
  - Always a writer for every read and vice versa
  - No EOF or SIGPIPE possible
  - OUTPUT will only block on full pipe
  - ENTER will block on empty (or closed!) pipe
- Well-designed cooperative processes a MUST

©1991 Hewlett-Packard Company

Using TRANSFER with Named Pipe

- Creates subprocess (rmbxfr) to read/write pipe
- Main BASIC/UX process remains unblocked
- Use EOR/EOT interrupts to signal data availability
- Use ON DELAY to act as watchdog timer
- Advantages:
  - No blocking
  - Multiple sets of TRANSFER and pipe, can connect to single RMB
- Disadvantages:
  - Not as easy to implement as OUTPUT/ENTER

©1991 Hewlett-Packard Company

Application Development: Building Complex Models Visually

Benefits of Using HP VEE

- Time spent solving the problem – no time spent remembering syntax
- Development time is decreased
  - No edit, compile cycle
  - Changes made quickly
- Multifunctionality of objects based on data types and shapes
- Inherent user interface
  - Visual orientation
- Automatic Data Typing

© 1991 Hewlett-Packard Company

Top Down Design

- Define the problem and its constraints
- Identify and define logical order and sequence
- Define subtasks
- Further define each subtask into manageable units
- Implement units
  - UserObjects
- Structured programming
  - Exactly same principles apply as in languages

© 1991 Hewlett-Packard Company

Levels of Complexity: HP VEE-Test

- Test and measurement: data flow
- Sequential flow along data path
  - Sequence determined by data type, shape
- Few objects; mostly I/O and display
- Usually more emphasis on instrument, particularly with direct I/O
- Data acquisition: high sample rates
  - Optimization
- More analogous to the Basic world

© 1991 Hewlett-Packard Company