V6. Data Links

One of the most cumbersome tasks related to managing standard Mac dialogs was that of transferring information between program variables and control values (i.e., getting stuff in and out of a dialog). ViewIt's "data linking" feature simplifies this task by supporting (1) direct links between program variables and control values, plus (2) automatic data type interconversion when variable types and control types differ.

The Old Way

The "old way" of moving information to and from dialogs was to use Control Manager toolbox calls. In cases where the standard control value or title represents the "value" of the control, and only a few such controls in a window need to be managed, it may be simpler to use the old way than to set up data linking. The toolbox call "SetCtlValue" can be used to reset the control value, and "SetCTitle" its title. To get control info, call GetCtl and read the info from fRec variables: cValue = control value, cTitle = control title.

There are several disadvantages, however, to the old way that will usually lead you to use data linking to set and get control values: (1) the Control Manager toolbox calls are not always very smart about redrawing controls, (2) some controls have "values" that can only be accessed through data linking, (3) the Control Manager toolbox calls do not provide any help in converting between data types, and (4) using toolbox calls is very cumbersome when dealing with a large number of controls.

The New Way

The "new way" depends upon a connection or "data link" being established between program variables and controls. Information about such links is stored with each control and consists of the linked variable's memory address, type, and display format. Links can either be established by passing the address of a record containing variables to be linked when calling NewWnd, or by calling LnkCtl one or more times to link isolated variables.

Once data links are established, then the ViewIt commands SetVal and GetVal can be used to update the linked values. SetVal moves values from program variables to controls,

program variable --> control value

and GetVal moves control values to linked variables,

control value --> program variable

where both SetVal and GetVal support setting or getting either a single value or all linked values at once.

The remaining discussion in this topic reviews details of data linking: data types supported, choosing controls to link to variables, establishing links, and using Set/GetVal to update values. This discussion will be easier to follow if you examine example code such as that found in the vDemoXY program which illustrates linking of a record "myRec" to four different controls, and of using LnkCtl to link an isolated variable "myList" to a scrollable list control. A complete description of the vDemoXY source is found in the on-line help within the program's "Modal ViewIt Window".

• Data Types

ViewIt and its control drivers make use of the following integers to identify the "data types" used in data linking:

1 = signedByte (1-byte integer)
2 = integer (2-byte integer)
3 = longint (4-byte integer)
4 = computational (8-byte integer)
5 = real (4-byte real)
6 = double (8-byte real)
7 = extended (10-byte real)
8 = extended96 (12-byte real)
9 = RGBColor (6-byte array of 3 integers)
10 = OSType or ResType (4-byte character array)
11 = Handle to an unlocked, nonpurgeable, relocatable block in memory (4 bytes)
12 = extended96 (12-byte real, THINK C "universal")

-1 to -255 = space for a string that has the same size as a Pascal string of this absolute length ("-1" uses 2 bytes)

IMPORTANT NOTE: A common mistake in data linking is to use the storage size of a string as its data type instead of its length. A string variable "myString", for example, declared as:

```
myString : string[9];   Pascal
char myString[10];      /* C, C++ */
character*10 myString   !FORTRAN
```

would correspond to data type -9, but occupy 10 bytes of memory (bytes used = -data type + 1). If you passed -10 as the data type for this string when setting up data linking, then ViewIt would assume that the string occupied 11 bytes, which means that ViewIt would eventually clobber the byte in memory that follows this string variable.

In Pascal, the extra byte is used as the leading length byte. In C, the extra byte is the terminating null byte. And in FORTRAN, the last byte of the string cannot be used to store information since it is lost when the string is converted to a Pascal string within ViewIt.

• Choosing Controls

Each ViewIt control has a preferred or "native" data type that can be found in cStorType. This native type indicates the data type of the control's associated "value". This value can be of any type in the above table, meaning that it need not be based upon the "contrlValue" field of the standard control record. (This flexibility does not apply to controls based on CDEFs, however, which always use "contrlValue" as their value.)

You can determine a control's native type and other control info from the driver's on-line help which is accessed from within ViewIt's Control dialog or the main ViewIt help window. For typical controls found in dialogs, however, the native type is obvious: strings for text items, and integers for check boxes and radio buttons (where value 0 = unchecked, 1 = checked).

Although it is not necessary for the control's data type to match the variable's data type that it is linked to, some combinations will make more sense than others. It makes sense, for example, to link a real variable to an editable text control, but less sense to link a string variable to a check box. Typical links seen in ViewIt's own dialogs are:

string variable <--> editable or static text control
integer or real <--> editable or static text control
integer (0 or 1) <--> check box or radio button
bit flags <--> menu or list control (multi-selection)
integer <--> menu or list control (single-selection)

Also note that it does not make sense to link button-type controls or static controls that simply display a single icon or picture. Thus windows often contain some controls that are linked and others that are not.

• Linking A Record

Once controls to be linked have been added to a window, then their association with program variables must be established. One way of doing this is to associate a single record with the linked controls in the window. This is done by specifying 1) the data type and memory byte offset of the variable linked to each control, and 2) the memory location of the program record that contains the variables.

The first step is done on page 2 of ViewIt's Control dialog by setting the control's "Variable Type" and "Byte Offset" ("page 2" refers to the fact that the Control dialog has two pages or views that can be switched by hitting an icon that appears as a page corner at the bottom, right of each view). Alternatively, this information can be set in ViewIt's Links dialog which has the advantage of showing all of the data links for all controls in the window (but no other control information).

"Variable Type" is the data type of the linked variable (see above list), and "Byte Offset" gives the memory location of the variable as a byte offset into the program record. Note that many compilers "byte pad" odd-sized variables in records, so keep this in mind when counting byte offsets. A string, for example, that has a storage size of 9 bytes (data type -8),

```
myString : string[8];   Pascal
char myString[9];       /* C, C++ */
character*9 myString    !FORTRAN
```

would occupy 10 bytes of a record due to byte padding. On the other hand, two 1-byte booleans or integers can be put together in a record to occupy just 2 bytes, but a 1-byte variable followed by a multi-

byte variable will occupy 2 bytes due to byte padding.

Also set the "Number Format" and "Decimals/Digits" if the link requires conversion of numbers to strings. The format and digits fields have the same meaning as parameters c and d in the Num2S command (since ViewIt uses Num2S to do its internal number-to-string conversions).

The second part of establishing data linking is to provide ViewIt with the address of the linked program record. This address is passed in parameter d when calling NewWnd (as described under NewWnd in the "Window Commands" topic). ViewIt then uses this record address and the variable's byte offset into the record to calculate the memory location of each linked variable.

Parameter c of NewWnd can be optionally used to pass the size of the linked record. If c > 0, then any controls associated with memory addresses that are not within the program record are not considered linked. This prevents ViewIt from poking memory outside of the record.

• Linking A Variable

In some cases you will not find it convenient to use a single record to establish links with program variables. An alternative approach is to make one or more calls to the LnkCtl command (described under LnkCtl in the "Control Commands" topic) to establish links between controls and isolated program variables.

Calling LnkCtl results in the same private control variables being set as when linking an entire record, and the two schemes can be combined (i.e., some controls in a window might be linked to a record, and others to isolated variables). Note that if you use LnkCtl, then you do not need to worry about setting the control's "Variable Type", "Byte Offset", etc., fields in the Control dialog since this information is passed directly to ViewIt as part of the LnkCtl call.

• Using Links

The link between a program variable and a control is not used until GetVal or SetVal is called (see "Control Commands" topic for a complete description of these commands). GetVal gets the current control value and moves it to the linked program variable, doing any necessary data type conversions before updating the variable. SetVal gets the current value of the program variable and uses it to update the linked control's value, again doing any data type conversions that may be necessary. The type conversions done by ViewIt include number <-> string, integer <-> real, and C <-> Pascal or FORTRAN <-> Pascal when the program's native string type differs from Pascal.

ViewIt uses the SToNum command to convert strings to numbers. If the string being converted is empty or not a properly formatted number (such as a number with more than one decimal point), then SToNum sets the number to an error value equal to the the corresponding fRec variable fI1Err, fI2Err, fI4Err, etc. Default values for fI1Err...fR12Err are zero, but these can be reset by the program to special values that indicate when an error has occurred. This feature is particularly useful when using GetVal to read a large number of values all at once from a window. After GetVal, the linked variables can be quickly scanned to check for bad values.

Use of data linking can dramatically reduce the amount of code necessary to manage a ViewIt window. A good example of this can be found in the "vDemoXY" program. Also note that many of ViewIt's built-in dialogs use data linking to move values between variables and controls. Simply enter editing mode and examine the links to see how this was done.

Finally, note that the links between program variables and controls do not depend upon the position of the controls in the control list. Controls can always be reordered without affecting data links. Controls can even be replaced with "improved" versions as long as links are reestablished.

WARNING: If more than one control is linked to the same variable when GetVal is called to obtain all linked values in a window, then that variable will be assigned the current value of the last control in the control list that is linked to the variable (even if such a control is hidden or inactive).

• Handle-To-Handle

Data type 11 refers to the handle of a relocatable block. In this case, the entire contents of the block (not the 4-byte handle) get moved in response to GetVal and SetVal. If a BaseCt static PICT control is linked to a handle, for example, then each time SetVal is called ViewIt copies the contents of the block to the PICT resource in memory and redraws the control (by calling SetHandleSize, BlockMove, and DrwCtl). Thus "flipping" from one program-created picture to another can be done by simply resetting the linked handle to the next PicHandle and calling SetVal.

WARNING: When setting up data linking with data type 11, pass the address of a program variable containing the handle (not the handle itself!) in parameter b when calling LnkCtl.

• Custom Links

Some control drivers that support more complex controls have their own "custom" way of doing data linking. In this case the driver is in complete control of what happens when GetVal or SetVal is called, and you may not even need to set up a data link for these commands to work. The CommCt communications control driver, for example, uses GetVal to return an entire record of information to the program. Such custom linking will be described in the driver's documentation.