V6. Data Links

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

Note that data linking need not be used in cases where the standard control value or title represents the "value" of the control, and there are only a few such controls in a window. In this case, simply use the ViewIt command GetCtl to get control values (the standard value and title are returned in cValue and cTitle), and the toolbox calls "SetCtlValue" or "SetCTitle" to set values. In many cases, however, the "value" of a control will not be its standard value or title, and data linking will be the best way to get and set values.

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)

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. 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). "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. Also set the control's "Number Format" and "Decimals/Digits" if the link requires conversion of numbers to strings. The format & digits fields have the same meaning as parameters c and d in UtilIt's Num2S command (ViewIt uses Num2S to do its 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 (see the "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 (see "Commands") 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).

Using Links

The link between a program variable and a control is not used until GetVal or SetVal is called (see the "Commands" topic). 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 UtilIt's 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. The 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).

Special Cases

There are two useful exceptions to the normal data linking scheme supported by ViewIt. First, 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 it is easy to "flip" from one program-created picture to another by simply resetting the linked handle to the next PicHandle and calling SetVal.

The second exception occurs when the control driver sets a control's native type (cStorType) equal to its baseID (the resource ID of the driver). In this case, ViewIt expects the driver to do all of the work when responding to GetVal and SetVal, making it possible for the driver to link entire records of information to a single control. Since this type of data linking is completely managed by the driver, you'll have to read its on-line help for programming details.