INPUTE and INPUTN: Input Control In A GUI Environment

By Ed Holling

Ed is a BASIS Software Test Engineer. He has recently been responsible for quality assurance on GUIBuilder.

Input validation is an important aspect of any successful business application. Users are notorious for entering data in every format except the one an application needs. A program that simply accepts just any user input is condemned to waste a great deal of time dealing with bad information. And, as applications move away from the character-based and toward event-driven GUI design, input validation needs become even more complex. BASIS is meeting those validation needs with new visual INPUTE and INPUTN controls in Visual PRO/5® 2.0.

Before the INPUTE and INPUTN verbs were available from BASIS, data entry in Business Basic applications could be arduous. Users were forced to key data in linear fashion, delete back to any mistakes they made, and then retype the section or line over again. This unproductive input method proved so frustrating that many developers designed their own input controls to more creatively handle the entry and editing of keyed data. But, like any complex program subsystem, these custom data entry routines required lots of program maintenance time and tended to slow down the entire application.

INPUTE and INPUTN verbs moved the responsibility for data format validation out of the application and into the language. The result was less design work for applications requiring data entry. Developers could now restrict users to certain kinds of keypresses when entering their data--an ability known as *input masking*. By specifying a character template, a *mask*, for an input control, it was possible to ensure that all user input was consistently laid out before it was handed back to the application. This allows the underlying application to safely assume that all input is in the proper character format. For example, if a user needed to supply a three-digit area code for a North American telephone number, the INPUTE and INPUTN verbs helped save CPU resources and development time by ensuring that only three digits were entered. Masks could also be combined with padding characters to give the user strong visual cues about the expected input format. These verbs have since become widely successful and are now an accepted standard in the Business Basic community.

As GUI has emerged, the issue of sophisticated input control has resurfaced. Suddenly users are in charge of everything that happens on an application's interface. They are no longer content to enter multiple fields in a programmed order; they want to be able to cut, paste, insert, and remove characters from input fields at will. And input validation is just as important for controls that conform to the Windows paradigm as it was in character-based systems. With Visual PRO/5 2.0, BASIS offers new, updated GUI-control versions of the INPUTE and INPUTN verbs that give some of the most consistent, powerful input validation tools available in any of today's development environments.

About INPUTE/INPUTN

INPUTE is an alphanumeric edit control that lets the application specify what letters and digits it will accept as input and where these letters and digits should be placed in the input field. This kind of control has a wide variety of uses. For example, if your database is keyed on a unique product ID consisting of three uppercase letters followed by a dash and four digits, it is possible to ensure that all data returned by the INPUTE control follow this format by using an input mask of the form `AAA-0000`. Once the mask is in place, your application is saved the trouble of checking for this format before acting on the input.

INPUTN provides similar control over numeric data input. This control can be of particular benefit when entering signed decimal values such as monetary amounts. As an example, INPUTN can be instructed to "pour" data, calculator-style, into a field with a fixed decimal position, and then return values with an appropriate international decimal separator.

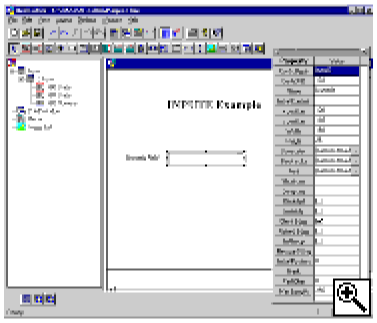
Using INPUTE/INPUTN

Like similar SYSGUI controls, it is possible to generate INPUTE and INPUTN controls by directly adding the appropriate mnemonic to your program, such as:

```
'INPUTE'(id,x,y,w,h,flags$,len,pad${,initpos,restore$},val$)
```

But in Visual PRO/5 2.0, developers can also choose to work with these controls in ResBuilder™, a new graphical resource editor. ResBuilder offers full GUI support for the placement and definition of INPUTE and INPUTN controls, allowing you to interactively design the layout of your input screens without having to type a single line of code. And after specifying the behavior of your INPUTE and INPUTN controls in ResBuilder, you can begin constructing your back-end application by connecting your screen definitions with automatically generated code created in the new GUIBuilder™ visual programming environment. Using ResBuilder, you can incorporate INPUTE and INPUTN controls as you create your Windows resources.

These new controls also interact with your core application in a new way. In a character-based program, the application would send a user to a control, wait for the user to finish working with that control, and then return the value the user entered. But because Windows applications are event-driven, an application using INPUTE and INPUTN controls waits for any of the available controls to tell it what the user is doing, takes whatever action is appropriate, and then informs all other available controls of any relevant changes.



Using ResBuilder, you can incorporate INPUTE and INPUTN controls as you create your Windows resources.

This communication takes place through the SENDMSG() function and the Notify event. SENDMSG() can be thought of as an enhanced version of the CRTL() function, letting an application retrieve and modify parameters associated with a given control. The INPUTE and INPUTN controls use SENDMSG() to dictate their visual attributes (such as size, location, and coloration) and their internal functionality (such as title string, cursor position, and mask settings).

These controls can also pass information back to the application using the Notify event. This event expands the existing event string structure to include information on error codes and special keypresses, such as the function, control, and keypad keys.

For more information on using SENDMSG() and Notify in Visual PRO/5 programs, read Jim Douglas' Spring 1998 *Advantage* article, "Event-Driven GUI Programming With Notify And SENDMSG()," available online at *www.basis.com/advantage/mag-v2n1/eventdriven.html*.

INPUTE and INPUTN controls are also accessible through the standard Windows clipboard commands, making it easy to move information between the controls and other standard Windows applications.

The best of the old and the new come together with the new INPUTE and INPUTN controls. Now you can have the familiar, stable functionality of input validation with INPUTE and INPUTN in your Windows applications with Visual PRO/5 2.0.