User Versions of Interface Tools

Although the primary focus of the new user interface for the HP 48G/GX was to enhance our built-in applications, it became apparent as the project progressed that calculator owners who program would want access to the same capabilities to enhance their efforts. For the choose box, message box, and especially the input form tools, the biggest challenge involved scaling back the numerous features to produce simple user commands that still offer customization potential.

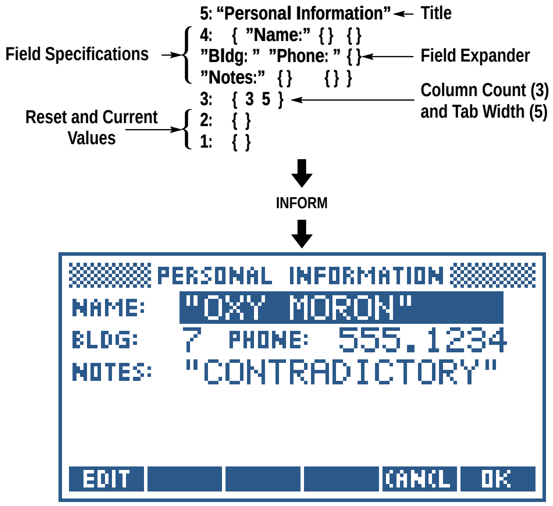
The message box command, MSGBOX, was designed to display pop-up messages with a minimum of fuss. Thus, it takes just one argument—the message string—and produces a word-wrapped normal-sized message box.

The choose box command, CHOOSE, is slightly more complicated. To enable but not require the same object-oriented use of choose boxes as the built-in applications, the CHOOSE command accepts a list of items in two formats. In the simplest format, an item is specified by a single object, which is displayed and returned if chosen. In the alternate format, an item is specified by a two-element list object. The first element is displayed in the choose box, and the second element is returned if the item is chosen.

For simplicity of the user interface, CHOOSE displays a normal-sized choose box without the multiple-choice capability used by some built-in applications.

The MSGBOX and CHOOSE commands largely follow the same interface specification methods as their system-level counterparts. This differs markedly from the input form user command, INFORM. To maintain complete flexibility over all elements of form layout and behavior, the input forms engine takes three arguments for each label and thirteen arguments for each field, specifying such details as exact location and size, display format, and so on. Added to that are global arguments for the form procedure and form title and some other details. All together, an input form with four labeled fields requires 68 arguments. While this amount of information is justified for the varied needs of built-in applications, it is an unnecessary burden for programmers just wanting to get some simple input from the user.

For the INFORM command, therefore, we developed an automatic form layout scheme that serves most needs, with options for further detailing. Basically, the INFORM input form is viewed as a grid that is filled with fields starting in the upper-left corner and proceeding from left to right and top to bottom. The number of columns in the grid is specified as one of INFORM's arguments, and each field's width is determined by the width of its label and by the user-supplied tab width, which places invisible tab stops within each column to help align fields vertically. A field can span multiple columns with a special field-expander specification. Help text and object type restrictions can be included for any field, but aren't required.



```
5: “Personal Information”
4:  { ”Name:” { } { }
”Bldg: ” ”Phone: ” { }
”Notes:” { }   { } }
3:  { 3 5 }
2:  { }
1:  { }
```

```
PERSONAL INFORMATION
NAME:  "OXY MORON"
BLDG:  7  PHONE:  555.1234
NOTES: "CONTRADICTORY"
EDIT                 CANCL  OK
```

Fig. 1. A custom input form created by INFORM.

Fig. 1 shows an example of a custom input form created by INFORM. Notice that, despite the relative simplicity of the input arguments, an input form with aligned fields of varying widths is presented. This technique for building input forms proved so valuable that it was used to create the Solve Equation input form, which changes according to the number and names of variables in the equation to be solved.