Letters, Loopers and Some Very Strange Relationships

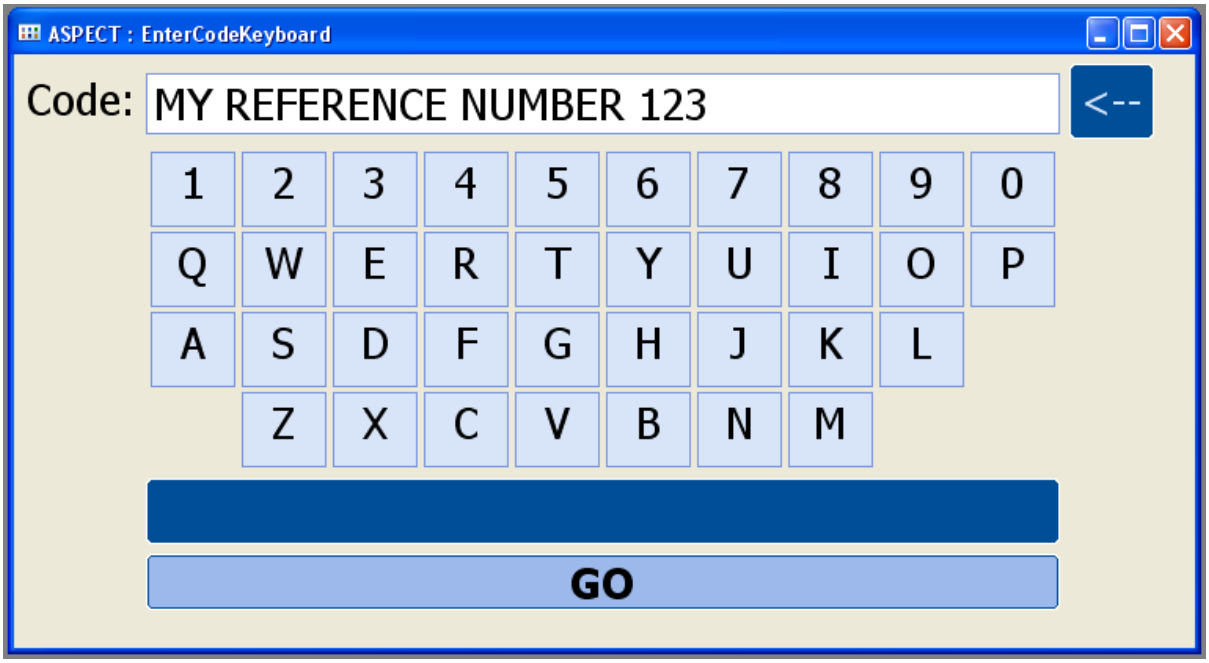
Time for some daft, 'out there' ideas … and a bit of fun.

Here's the goal: let's imagine an environment, such as a busy, messy factory floor, where gloves and mess make it impossible to use a keyboard.

So you have a touch-sensitive screen, covered with a suitable protective layer, and want to give users the ability to look up something by typing in a reference number or keyword.

You need an on-screen keyboard, and because you love doing things in Ffenics or DataEase, you're determined to create that keyboard using your favourite software.

Something like this:



So, we have a label, and field into which a value can be typed. There's a funny arrow button to the right which is masquerading as a backspace button, a rough QWERTY keyboard plus integers, a blue bar representing the space bar, and a button labelled 'Go' which could be used to start a report that returns whatever it is supposed to return.

I'm using a couple of tricks here, including one that I'm surprised I've only just started to use regularly.

If you're looked at other articles by me in which some set of buttons with letters from the alphabet is used, you should know that I have not wasted time and tightened every muscle in my back by laying out 36 buttons, each with their own set of scripts, to represent the keyboard. You should guess that I'm using a subform (just for the QWERTY letters and numbers; everything else is on the main form). And that the subform is build on my swiss-army-knife-of-a-dataset, Looper.

Or maybe you haven't a clue what I'm on about. So let me explain.

Looper is a table of integers. To be precise, it is a table of ordinal numbers -- that is, integers in order. Why I call it Looper is my own business. [1]

The area on screen is a 10 x 4 subform. Each record in that subform is a number from 1 to 40, the first 40 records in Looper.

The object with the appropriate letter displayed is a 'layout-only virtual', that is, a field that only exists on this document. It happens to be called vLetter, is one character long, and is derived:

```
midc ( "1234567890QWERTYUIOPASDFGHJKL  ZXCVBNM   " , LoopNo , 1 )
```

That odd string is the set of characters from your QWERTY keyboard, starting with the numbers across the top, then each row of upper case letters. The spaces towards the end help with the indenting of the keyboard.

LoopNo is our actual field in the form, the one containing the ordinal integer. The derivation basically finds the corresponding letter for the key. For example, the letter 'R' is in the second row and is four columns over on the keyboard; the ordinal number behind that record will therefore be 14, and the 14th letter in my string is the letter 'R'.

As I said, these are records in a subform. But what's the nature of the relationship? Well, this is one of my new tricks: the main form and Looper match based on … NO match fields.

Yes, they match on nothing. Nada. Zilch. And as a result, ALL records in the subform display. Try it and see with any old forms.

Now, Looper probably contains several thousand integers. But because I have removed the scrollbar from the subform, we only see the first 40 records. (Actually, if I had used a button instead of a text object for the letters, we'd have to deal with a minor issue about scrolling because the button can get focus, but let's worry about that some other time.)

The letter field has two scripts: a clicked event for adding to the string of text in the main form's 'code' field, and a ValueLoaded event to hide the field if its value is a space (to make the layout look a bit more like a keyboard).

The clicked event script is:

```
define "t" number .

Customer.FirstName.Value :=
  concat ( Customer.FirstName.Value ,
    if ( getarray ( 1 ) = "SpacePending" , " " , "" ) ,
    vLetter.Value ) .

t := setarray ( 1 , "" ) .

Customer.FirstName.Show () .
```

[1] Well, ok, since you ask. My first use was so I could repeat something x number of times in a complicated report. So complicated (it was a Bank of England statutory report that commercial banks are required to submit on a regular basis, and had the most ad-hoc layout) that it started to drive me Loopy. The name stuck.

In a recent blog post I covered how to update values between sub and main forms using scripts. Basically, we are adding the value of the letter to the end of the value already in our main form field.

Spaced Out

The getarray bit obviously needs explaining. It turns out that when you concat a space to the end of a field, the value in the field seems to immediately be trimmed of the trailing space.

The dark blue bar, as I mentioned above, is the space key. When this is clicked, this script:

```
define "t" number .
t := setarray ( 1 , "SpacePending" ) .
```

justs sets a flag, using array #1, to indicate that we are expecting a space. When a letter is clicked, we can then first insert such a pending space before the letter, and get around the trimming of spaces described.

Each letter also resets array #1 to blank, effectively meaning that the space is no longer pending.

The valueloaded script on the letter field is:

```
if vLetter.Value = " " then
  vLetter.Hide () .
else
  vLetter.Show () .
end
```

Two other things need mentioning. The first is that I found issues with Ffenics if the record contained no actual fields from the form. So Looper's LoopNo is actually on the record, scrunched up in the top left corner. It has its new display option 'hide' checked, which means that it is never actually made visible. DataEase for Windows scripts didn't need underlying fields present.

The other item is the backspace button. This deletes one character at a time from the string. Its script is:

```
Customer.FirstName.Value := firstc ( Customer.FirstName.Value ,
  length ( Customer.FirstName.Value ) - 1 ) .
Customer.FirstName.Show () .
```

Of course, you might want to have more characters than I've chosen. I'm sure someone could have some fun working out how to put the display into upper or lower case, or to imitate the keyboards found on some PDAs and smart phones. You could also add a 'Clear All' button. And another intriguing problem would be to have more than one field for data-entry. That's your homework.

Time to implement this from scratch: less than one hour.