

## Table

As previously mentioned, NS Basic doesn't currently support the Palm OS Table object. I'll give you a way to work around the lack of a true Table object, and you can decide if it fits your needs.

Emulating a Table object involves emulating both the function and the appearance of a Table. You might expect otherwise, but it turns out that it's easier to get the function working well than it is to produce a realistic looking table. So, we'll start with the function, then present the visual component later.

If you think of tables as rows and columns (a spreadsheet analogy comes to mind), it's easy to make the transition to a two-dimension array, where the first subscript refers to rows, and the second columns:

```
Dim TableArray(20,50) ' create 20 x 50 table array
```

The rest should be simple – create 20 rows of 50 columns of fields, where each field corresponds to a single array element. There's just a little problem – the screen! A 20x50 array (1000 array elements) on a 160x160 pixel screen (25,600 pixels) results in an average of 25.6 pixels per element. If you make each table cell 5 pixels wide and 5 pixels high, the entire array will fit. You won't be able to display anything *useful*, but it will fit.

So, what we need to do is decide how much information we want to display – that will determine how many rows and columns we will display at one time. Since we may want to allow our user to enter information directly into our table, we'll use fields for each table cell. We'll also want row and column identifiers, so we'll use labels (with AutoShift, Dynamic Size, Editable, Single Line, Left Justified true. Better still, let's use buttons (they're center justified, and they give visual feedback if our users tap them). My Table Demo project uses 6 rows and 4 columns, with an additional row of buttons for column headers, and a column of buttons for row headers. That's already 34 objects on the screen, and we still need to add some navigation controls, so we'll also add buttons to scroll in each of the four directions (up, down, left, and right). I like my users to know what's happening, so I'll add a status field.

You can create this project and screen for yourself, but make things easy for yourself, and load the Table Demo project. If you do, you'll notice that I use a 26x26 element array, but you can use whatever size you wish. I chose 26 elements to make labeling easy (26 letters in the English alphabet). I also use separate arrays for the row and column headers, each array also containing 26 elements.

C	D	E	F
5 Day	Time	Gluc	pH
6 Mon	12	130	7.35
7 Tue	4	78	7.42
8 Wed	8	210	7.38
9 Thu	12	104	7.41
10 Fri	4	96	7.44

Regardless of the size of the arrays, the concept is the same. The screen fields represent a subset of the actual array. When displaying the array, all that's needed is to maintain control of where we are in the array, and display the 6 rows and 4 columns at that array position. We don't need to keep track of each element, only the top left cell's row and column numbers. Each additional row and column is offset from the top left according to its position on the screen. Row and column headers get displayed according to the same numbers. Here's part of my subroutine to display the screen fields from any initial top left (row, column) pairing:

```
Sub DisplayTableInfo()  
fldA1.text=TableDataArray(CurrentRow, CurrentCol)  
fldA2.text=TableDataArray(CurrentRow+1, CurrentCol)  
fldA3.text=TableDataArray(CurrentRow+2, CurrentCol)
```

```

fldA4.text=TableDataArray(CurrentRow+3, CurrentCol)
fldA5.text=TableDataArray(CurrentRow+4, CurrentCol)
fldA6.text=TableDataArray(CurrentRow+5, CurrentCol)
fldB1.text=TableDataArray(CurrentRow, CurrentCol+1)
fldB2.text=TableDataArray(CurrentRow+1, CurrentCol+1)
fldB3.text=TableDataArray(CurrentRow+2, CurrentCol+1)
fldB4.text=TableDataArray(CurrentRow+3, CurrentCol+1)
fldB5.text=TableDataArray(CurrentRow+4, CurrentCol+1)
fldB6.text=TableDataArray(CurrentRow+5, CurrentCol+1)
'... and so on...
End Sub

```

Scrolling the table involves incrementing or decrementing the current top and left (row and column) values, then calling the display routine. Updating the table array with each field's contents is performed when the user leaves the field. The following line, from the code for the fldB3 (3<sup>rd</sup> row, 2<sup>nd</sup> column) is:

```
TableDataArray(CurrentRow+2, CurrentCol+1)=fldB3.text
```

Clearing the table array is performed easily, looping through the rows and columns:

```

Sub ClearTable()
For CurrentRow=1 to MaxRow
  For CurrentCol=1 to MaxCol
    TableDataArray(CurrentRow,CurrentCol)=" "
  Next
Next
CurrentRow=1
CurrentCol=1
End Sub

```

For more ideas on how to construct a functional table, see the Table Demo project, and feel free to modify the code for your own projects.