Re BASIC! Form Generator

Disclaimer

This SOFTWARE PRODUCT is provided by El Condor "as is" and "with all faults." El Condor makes no representations or warranties of any kind concerning the safety, suitability, lack of viruses, inaccuracies, typographical errors, or other harmful components of this SOFTWARE PRODUCT. There are inherent dangers in the use of any software, and you are solely responsible for determining whether this SOFTWARE PRODUCT is compatible with your equipment and other software installed on your equipment. You are also solely responsible for the protection of your equipment and backup of your data, and El Condor will not be liable for any damages you may suffer in connection with using, modifying, or distributing this SOFTWARE PRODUCT.

You can use this **SOFTWARE PRODUCT freely, if you would you can credit me in program comment:**

El Condor – CONDOR INFORMATIQUE – Turin

Comments, suggestions and criticisms are welcomed: mail to rossati@libero.it

Conventions

Commands syntax, instructions in programming language and examples are with font `COURIER NEW`. The optional parties of syntactic explanation are contained between `[square parentheses]`, alternatives are separated by `|` and the variable parties are in *`italics`*.

Table of Contents

1 Form generator

The script fgen is an RFO Basic script (version v 1.91), which allows to build and handle forms; it is sufficiently generalised for a wide use.

1.1 Calling the form generator

The function fgen has one parameters a list of views; it returns a map of data.

```
sample$ = "TITLE,RFO Basic Form Generator,silver,red;" ~
                  + "S,Urgency,,,W:White,G:Green,Y:Yellow,Red;" ~
                  + "T,Mail,E-Mail,18,insert mail;" ~
                  + "N,Number,,10,Insert Integer Number;" ~
                  + "P,psw,Password,15,Insert password,Almost 6 characters;" ~
                  + "U,UnMod, ,25;" ~
                  + "T,T1;" ~
                  + "CKB,Consent,,,to keep data;" ~
                  + "Defaults,Number:3.1415,UnMod:-------------------------;"

include fgen.bas
fh_data = fgen(sample$)
```

1.2 Data description

Every view is a list of attributes which are comma separated in this order: *Type* of view, *Field Name*, *Field Label*, *Length*, and *Extra*(s). The views are separate by semicolon.

In addition to the views there can be some others information (*Pseudo types*) with different semantics, that will be detailed in the paragraphs dedicated to them.

If the view list starts with `%` it is a comment which ☞ must also be terminated by semicolon.

The `Ok`, `Reset` and `Cancel` buttons are automatically added to the form.

1.2.1 Type

- Buttons:
- **B** button for change the caption of `Ok`, `Reset` and `Cancel` buttons;
- **CKB** check box, values are `On` for selected check boxes, `Off` otherwise; if the default value is `On` then the check box appears checked.
- Combos:
  - **CMB** combo box (spinner)
  - **CMT** combo box with modifiable text
- Text fields:
  - ~~**C comment;**~~
  - ~~**F file;**~~
  - ~~**D folder;**~~
  - **N** integer numeric field (right aligned);
  - ~~**S slider is an extension of the standard control: it works also on float number range and inverted direction e.g. the start value can be greater than the end value;**~~
  - **P** password field;
  - **T** text field;
  - **U** not modifiable field.

The Types are indifferent to the case.

~~For comments, **C** type, this value is displayed instead of Label Field, which is ignored.~~

1.2.2 Field Name

Is the name of the view, which is returned, when the form is closed, with the value associated.

1.2.3 Field Label

Label of view or caption of button, if omitted it is used the Field Name.

1.2.4 Length

The length, in characters, of the view. If the length is omitted it is assumed to be `20`.

1.2.5 Extra

For `spinners` (CMB type) ~~and `Button List`~~ is an item list separated by `,` (`comma`); for text files is a hint.

~~For Type **S** Slider, contains the start and end values in the form `start-end`, e.g. `-5 5`; if omitted the range is `0 100`.~~

~~For Type **F** can be a filter file e.g. `Images (*.jpg;*.bmp)|Videos (*.avi;*.mpg)`~~

For combos ~~or button List~~ if one would return a code associated to the description, the item has the form:

`key:value`.

Example:

```
CMB,Unit,Measure Unit,5,Kg:Kilos,Lt:Litres,Mc:CubicMeters,Wh:Watt/hour
```

1.2.6 Pseudo types

Pseudo fields are flavours for show form; they have a type and the syntax is different from the normal views.

1.2.6.1 After

The pseudo field `after` is useful for insert ~~a button or~~ a check box ~~or a radio buttons at~~ right of a view:

`after,`*`viewNameBefore`*`,`*`viewNameAfter`*

where *`viewNameAfter`* is the view to be placed at right of the view *`viewName`**Before***

☞ In the list of widgets *`viewName`**After*** must appears before *`viewName`**Before***.

```
...
& "N,Age,,5,,age;" _
& "R,Sex,,10,Masculin,M:Man,F:Féminin;" _
& "After,Sex,Age;" _
...
```

1.2.6.2 Check

This pseudo field is used for some controls on fields:

`check,`*`fieldName`*`,[mail[,required]]`

☞ if `value` contains comma or semicolon they must be masked (see par. 1.3 Masking data and Insert Unicode characters).

1.2.6.3 Defaults

The syntax is:

`defaults,`*`viewName`*`:`*`viewValue`*`[,...]`

it is useful for populate the form.

☞ if *`viewValue`* contains comma or semicolon it must be masked (see par. 1.3 Masking data and Insert Unicode characters).

1.2.6.4 Ground Background

The pseudo field `Ground` or `Background` create a background in the form; the syntax is:

`[ Ground|Background],`*`Color`*`[,landscape]`

The default is a gray color `silver`, the default orientation is `portrait`.

1.2.6.5 Title

`TITLE,`*`formTitle`*`,`*`titleColor`*`,`*`titleBackgroundColor`*`;`

example: `TITLE,RFO Basic Form Generator,silver,red;`

1.2.7 Summary by type

1.2.7.1 Default and extra field

Type	Length	Default field	Extra field
B	Ignored	~~possibly `dis[able]` or `cancel`~~	~~Possibly name of CallBack function~~
CKB		`1` or `check[ed]` = checked	Possible Description at right of check box
CMB,CMT		*`value`*	An item list separated by `|` : `[`*`key`*`:]`*`value`*
~~**F**~~		~~Initial folder~~	
~~**S**~~		~~Initial value~~	~~Start and end value, default is `0 100`~~
T,N,~~DN~~,P		Initial value	hint,
U		Not modifiable text	

1.2.7.2 Buttons

The package adds the standard buttons `Ok`, `Cancel` and `Reset`, ~~if there are sections on all forms except the last the `Ok` button has caption --> and all forms except the first has a button with caption <-- and name `fh_Back`.~~

The buttons can be used both for take different actions on closing form both for show user caption instead of default `Ok`, this last can be obtained not only for `Ok` button with this syntax:

```
B,[Ok|Cancel|Reset],caption
```

The value of *default* field `dis[able]` is used to start the form with the button disabled.

The *extra* field contains a name of the function called when a CallBack Button is pushed, in the form: *`moduleName.functionName`*. or *`functionName`* if it is in the module which required the creation of the form.

The Ok button is replaced if there is almost one type **B** control in the list not associated, by `AFTER` pseudo type, to some control.

```
...
Dim fg As fgen      ' instantiate Form Generator class
...
Dim frm as String = "N,n1,Integer,10,;n2,DN,Decimal,10;B,Multiply,,10,,Main.CallBack"
fg.fg(Activity,frm,"Main.handleAnswerTest","")
...
Sub CallBack(btnName As String)  ' CallBack event handler
      Dim n1 As Float = fg.iIF(IsNumber(fg.valueOf("n1")),fg.valueOf("n1"),0)
      Dim n2 As Float = fg.iIF(IsNumber(fg.valueOf("n2")),fg.valueOf("n2"),0)
      Msgbox(n1*n2,btnName)
End Sub
...
```

The label of button can be a Unicode character which is a simple and efficient way to create buttons with pictures: the Unicode characters is in the form `#`*`nnnn`* where *`nnnn`* is a decimal value of the Unicode character.

The button can also have an image that must be in the assets folder; the name of image is in the label field and are accepted `png`, `jpg`, `jpeg`, `gif` and `ico` images.

Name	Decimal value	Symbol
edit	`#9998`	✎
delete	`#10008`	✘
check	`#10003`	✓
check bold	`#10004`	✔
email	`#9993`	✉

```
B,Cancel,#10008;
B,Reset,#8630;
B,Start,#9998,,myHandler,Go;
```

Table 1: Some UNICODE characters

cross	#10006	✖
	#8630	↶
euro		€
pound		£
white square		☐

1.2.7.3 Check box

For checked box insert into **default** field `check[ed]` or `1`.

The **extra** field can contain a possibly description at right of the check box.

The value returned of check box is a string containing 0 or 1, they must be compared as string:

```
cmd.Append(fg.iIF(fh_Data.Get("Mandatory") = "1","M",""))
```

☞ If the check box is set after another view (see 1.2.6.1 After) the label is ignored.

1.2.7.4 File

The first *extra* field can contains a hint that is used also as description of the file type, the subsequent extras contain file extensions :

```
F,File,Image File,30,,Image,*.jpg,*.png;
F,psFile,PDF and PS files,50,,PDF and PS files,*.pdf,*.ps;
```

1.2.7.5 Radio buttons

It is possible to have more than one set of radio buttons.

The **length** is the length of the single Node; the **extra** field contains the item list separated by `|`. For get a key instead the description, the item must have the form: `key:value`.

The **default** value can be the data showed or the key:

```
Rdb,Status,10,R,Single,M:Married|S:Single|W:Widow;
Rdb,AgeType,10,R,Y,M:Months|Y:Years;
```

1.2.7.6 Slider

The *length* is the length of the text which shows the slider value.

The *extra* field of the type **S** can contains the start and end values in the form `start end`, e.g. `-5 5`; the range is `0 100 if` omitted, if only one value is present, the default value for the second is `100`; the result can have decimals depending on the difference from `start` and `end` value, see table at right.

abs(start - end)	n. decimals
> 99	0
<100 and > 10	1
<10 and > 1	2
<1 and > 0.1	3
...	...

`start` can be greater of `end` e.g.:

```
S,Slider,,5,-3,10 -10
```

☞If there isn't a default value and the slide is not moved the value returned is empty.

The qualitative slider (type **QS**) returns qualitative values taken from the *extra* field where they have the same syntax of the values of radio buttons:

```
...
QS,Urgency,,8,Green,White|Green|Yellow|Red
...
```

1.2.7.7 Combo box

CMB is a simple combo box, if no member is selected the value returned is an empty string; if the form has only one **CMB** combo box, there is only a cancel button and the form is exited when a combo box item is selected.

The *extra* field of combo box contain the item list separated by `|` (see description in Radio button).

CMT type is a combo box with text associated for insert a possibly value not in combo box.

CMX type is a combo box with text, every choice in combo box is recorded in the text, this is useful for example to compile a list of symptoms.

1.2.7.8 Text fields

For text type (**T**, **P**, **N**, **DN**) the possibly *extra* field is the *hint*; the possibly second *extra* field is the ToolTip.

1.3 Masking data and Insert Unicode characters

In some fields (*labels*, *extras*,...) the program replace the sequence `#nn...n` with the corresponding ASCII character, comma is `#44` and semicolon is `#59`.

1.4 Returned Values

The function return a map where the key is the field Names and the value is the value of the view, furthermore the bundle contains also the key `fh_button` with the name of the button pressed; if the form is cancelled there is only the `fh_button` with value `Cancel`.

1.5 Controls

Numeric fields, if the type is **N (**comma and sign are accepted**)**.

1.6 Remarks

1.6.1 Handling Buttons

fgen inserts the `Ok` button, the `Cancel` button and the `Reset` button.~~, this is function of the widgets contained in the form:~~

- ~~the `Cancel` button is always present,~~
- ~~the `Reset` button is present if there are data fields (e.g. Type F, D, N, P, R, CKB, CMB, CMT, S) ,~~
- ~~the `Ok` button is present if there are only data fields, in other words Buttons and Buttons List replaces the button `Ok`.~~

2 Data presentation

The data are presented in the order they appears in the parameters list~~, except for the Type **B** buttons that appears together buttons inserted by `fgen`, at the bottom of the form~~.

~~For view of Type Text, if the length exceed the maximum characters allowed for the line, the view is multi lined; this maximum characters for line depends from the labels width.~~

~~With the pseudo type `after` buttons or check box can be placed at right of another view.~~

The label of button can be ~~a name of an image that must be in the asset folder or~~ a Unicode character which is a simple and efficient way to create buttons with pictures: the Unicode characters is in the form `#nnnn`.

```
B,Cancel,#10008;
B,Reset,#x21B6;
B,Start,#9998,,myHandler,Go;
```

Table 2: Some UNICODE characters

Name	Decimal value	Symbol	Hexadecimal value
edit	#9998	✎	#x270E
delete	#10008	✘	#x2718
check	#10003	✓	#x2713
check bold	#10004	✔	#x2714
email	#9993	✉	#x2709
cross	#10006	✖	#x2716
dollar		$	#x24
euro		€	#x20AC
pound		£	#xA3

white square	▢	#x25a2

3 Others functions and utilities

3.1 The Sand Box

The script `sbFgen.bas` is an example of use `fgen`.

4 Technical notes

4.1 Arrays

- `field$` array of parameters view
- `widgets$` array of views

4.2 Bundles

Name	Key	Value	Note
`1 (GLOBAL)`	*bundleName*	*bundleReference*	Contains the reference to the bundles
`After`	*fieldNameAfter*	*fieldNameBefore*	ex. `After,Mail,Consent;`
`colors`	*colorName*	*alfa,red,green,blue*	
`defaults`	*fieldName*	*fieldValue*	
`fh_data`	*fieldName*	*fieldValue*	The data of form
`fields`	*fieldName*	*index*	index on fields array
`fieldsHandler`	*fieldName*	*handle*	
`handlers`	*viewReference*	*fieldName*	
`listValues`	*fieldName+value*	*key*	For Radio buttons and Combo box (spinner)
`touches`	*fieldName*	*touch rectangle*	
`tTexts`	*fieldName*	*text handler*	

4.3 Variables for personalisation

In the script `fgenCustom.bas`.

```
! Custom values
offsetX = 0.05 * screenW
offsetY = 100     % distance from the top
btnWidth = 70     % button width
deltaY = 55       % distance from views
charCheck$ = chr$(10004)      % check box character
tPause = 400      % pause change color
! title
title$ = "RFO Basic Form Generator"
bgColor$ = "silver"
titleColor$ = "black"
titleBgColor$ = "gray"
```

5 History

6 My be in future

- Radio buttons,
- seek bar,
- add Hidden Fields,

- handle CR and LF for Type **U**,
- extend field controls (e.g. range numeric, date, regular expression, etc.),
- date fields.

7 Annexes

7.1 Introduction to regular expressions

A regular expression is a string of characters used to search, check, extract part of text in a text; it has a cryptic syntax and here there is a sketch with a few examples.
The regular expression can be prefixed by modifiers such as **(?i)** to ignore the case.
The expression is formed with the characters to search in the text and control characters, among the latter there is a \ said *escape* used to introduce the control characters or categories of characters:

- **\ escape character**, for special characters (for example asterisk) or categories of characters:
  - **\w** any alphabetical and numerical character, **\W** any non alphabetical and numerical character,
  - **\s** *white space* namely. tabulation, line feed, form feed, carriage return, and space,
  - **\d** any numeric digits, **\D** any non digit,
- **.** any character,
- **quantifiers**, they apply to the character(s) that precede:
  - ***** zero or more characters
  - **+** one or more characters
  - **?** zero or one character (means possibly)
  - **{*n*}, {*n*,}** and **{*n*,*m*}** respective exactly *n* characters, almost *n* characters and from *n* to *m* characters.

(...) what is between parentheses is memorized,
?=*pattern* checks if *pattern* exists,
[a-z] any letter from a to z included,
[*a*|*b*] a or b,
\b word boundary,
$ (at the bottom),
^ (at start).

7.1.1 Examples

☞ The \ character in RE! BASIC string must be escaped for example: `re$ = "[aAbBcCdDeEfF\\d]{8}"`.

`^\s*$`	Empty set or white spaces
`aa+`	Find a sequence of two or more `a`, like `aa`, `aaa`,....
`(\w+)\s+(\w+)\s+(\w+)`	Find and memorize three words
`(\-[a-z])`	Find and memorize `minus` followed by one alphabetic character
`.(jpg|jpeg)$`	**Controls file type jpg or jpeg**
`^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$`	**Control of mail address**
`^\d+$`	**Only integers**
`((?=.*\d)(?=.*[a-z]+)(?=.*[\W]).{6,12})`	**Conditions for password** `(?=.*\d)` almost a digit from 0-9 `(?=.*[a-z])` almost one lowercase character `(?=.*[\W]+)` almost one special character `.` match anything with previous condition checking `{6,12}` length at least 8 characters and maximum 20
`^[-+]?\d{1,2}(\.\d{1,2})?$`	**Numeric values** `[-+]?` the sign is possible `\d{1,2}` one or two digits

	`(\.\d{1,2})?` It is possible to have a decimal point followed by one or two digits
`[aAbBcCdDeEfF\d]{8}`	8 hexadecimal digits