Basic4Android 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`*.

Contents table

1 Form generator

Form generator, briefly *FormGen*, is a class module for *Basic4Android* which allows to build and handle forms data; it is sufficiently generalized for a wide use.

1.1 Using the form generator

Before use FormGen the class `fgen` must be instantiate:

```
Sub Globals
...
      Dim fg As fgen    ' instantiate Form Generator class
...
End Sub
...
      fg.Initialize
```

The form is generated by calling the `fg` or `fgw` methods:

instantiatedName`.fg(`*activity*`,`*dataDescription*`,`*callingModule*`,`*subHandleAnswer*`,`*subHandleEvents*`)`

instantiatedName`.fgw(`*activity*`,`*dataDescription*`,`*callingModule*`,`*subHandleEvents*`)`

where *`activity`* is the activity which will contains the form, *`dataDescription`* is a character string containing the form components description, *`callingModule`* is an activity module or a class module, *`subHandleAnswer`* and *`subHandleEvents`* are characters string containing the name of the sub for handle data when the form is closed and the possible sub for handle events or an empty string if you wouldn't handle events.
If the form is only for show data *`subHandleAnswer`* can be an empty string; calling `fgw` allows to wait for the closing of the form and continue in the same function; see the example below:

```
fg.fgw(Activity,parameters,Me,"")
Do While fg.fh_isrunning
      Sleep(100)
Loop
If fg.fh_Data.Get("fh_button") <> "Cancel" Then
...
```

Another example:

```
...
Dim parms As String = "Slide,,5,S,3,10 -10;Rdb,Sexe,,10,,M:Male|F:Female;"
...
fg.fg(Activity,Parms,Me,"handleAnswerTest","handleEvents")
...
fg.fg(Activity,Parms,Me,"handleAnswerTest","")
...
```

☞ In one form it is possible to generate a new form by instantiating a new Form generator class.

1.2 Data description

Every view (or widget) is characterized by a list of attributes, comma separated, in this order: view *Type*, Field *Name*, Field *Label*, *Length*, *Default* Value and *Extra*(s). Views are separated 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 `'` or `#` it is a comment which ☞ must also be terminated by semicolon.

1.2.1 View Type

The Types are indifferent to case.

- Buttons:
  - **B** button;
  - **IB** inline button;2
  - **R** radio button, a set of Radio buttons;
  - **TOGGLE**, **TB** Toggle button.
- **CKB** check box.
- **CKL** check box list.
- **I**, **IMAGE**.
- Spinners (or Combo boxes):
  - **CMB** spinner;
  - **CMT** is a spinner with Text associated for insert values not in spinner;
  - **CMX** is a spinner with text, every choice in spinner is inserted in the text;
  - **F** file and directory.
- Text fields:
  - **COMMENT**, **C** comment;
  - **Date**;
  - **F** file;
  - **H**, **HIDDEN** hidden field;
  - **NS** numeric signed field;
  - **NF** decimal numeric field;
  - **N** absolute integer;
  - **P** password field, the data entered are masked;
  - **QS** qualitative slider;
  - **S** seek bar or slider is an extension of the standard control;
  - **T** text field is the default if the Type is omitted;
  - **Time**;
  - **U**, **UN** not modifiable field i.e. a protected field.

1.2.2 View Name

Is the name of the field that, when the form is closed, is returned with the value associated; the name is case-sensitive and it is used to access data and possibly manage the views both in the `subHandleEvents` both in *callBack* functions associated to buttons (see paragraph Buttons 1.2.7.2).

If *name* is omitted it is set to `fhw_n` where `n` is a progressive number.

1.2.3 View Label

Label of widget or caption of button, if omitted it is used the `FieldName` that it is transformed if it has one of those formats:

- `fieldName` it becomes `Field name`,
- `field_name` it becomes `Field name`.

1.2.4 View Length

The length of the view in characters; if it is omitted it is assumed depending on the *type* of view:

	View type	Value
B	Button	`70 dip`
C	Comment	If present is dip height
CHK	Check box	20 is the length for the possible extra field showed after the check
DATE	Text	12

DN	Decimal numeric text	10
F	File	`30`
IB	Inline Button	`35 dip`
IN	Numeric unsigned text	`7`
N	Numeric text	`8`
QS	Qualitative slider	`200 dip`
P	Password	`16`
R	Radio button	0 the space is proportional the description length
S	Slider	`200 dip`
	Spinners	`20`
T	Text	maximum from `20` and the length of the possibly default value
TIME	Text	10
TIMER		1000 milliseconds
TB	Toggle	6

1.2.5 Default value

Is the value proposed in form; the form is created with the defaults values and it is restored with the defaults values when the `Reset` button is pushed.

1.2.6 Extra(s)

Extra Field is used for add information to the view.

View type	
Buttons	A possible *CallBack* function
Check Box	a possible description after the check box
Date	user format ex. `Dd-MM-yyyy` (it is also the hint)
Radio buttons	an item list separated by `|`
Slider	the slider limits
Qualitative slider	an item list separated by `|`
Spinner	an item list separated by `|`
Text	if the default field is empty, is the hint

The possibly second extra field of text views is a ToolTip.

1.2.7 Summary by type

1.2.7.1 Default and extra field

Type	Length	Default field	Extra field(s)
B	Ignored	possibly `dis[able]` or ~~`cancel`~~	Possibly name of CallBack function, font information
C		the data	
CKB		`1` or `on` or `check[ed]` = checked	Possible Description at right of check box
CMB		*`value`*	An item list separated by `|: [`*`key`*`:]`*`value`*
DATE		`Now` or date	Format (default is `MM/dd/yyyy`)
F		Initial folder	
S		Initial value	Start and end value, default is `0 100`
T,N,DN,P		Initial value	hint, tool-tip
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, instead of button `Ok` there is a button with caption `-->` (is name is `fh_Forward`) and all forms except the first has a button with caption `<--` (is name is `fh_Back`).

The buttons can be used both for take different actions on closing form both for show user caption instead of default `Ok` or `Cancel` or `Reset` this can be obtained with the syntax:

```
B,[name|Ok],caption|fileName,[70|dipLength],[dis[able]],[function|cancel],
[fontInfo]
B,Cancel|Reset],caption|fileName,[70|dipLength],[dis[able]],[message],
[fontInfo]
```

or inline buttons:

```
IB,name,caption|fileName[:description],[35|dipLength],[dis[able]],
[function|message],[fontInfo]
```

The line button **IB** occupies the space of the label, the *caption* can be character(s) or a graphic file; the possible *description* is shown to the right of the button; ☞ *fontInfo* applies only to the caption and not to the *description* that is affected by a possible *Label* pseudo type (Label par. 1.2.8.7).

☞ *fileName* is a name of a graphic file in `assets` folder[1]; the file accepted are `.png`, `.gif` and `.jpg`.

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

The first *extra* field can contains a name of the *function* or a *message* called when a Button is pushed, *function* has the form: *moduleName*.*functionName*. or *functionName* if it is in the module which has required the creation of the form; ☞ the function receives the button name and the form is not canceled and not checked, if you would the form checked the function must starts like the example below where it is used *FormGen* `CheckErrors` function.

```
...
& "T,Name,,25;" _
& "T,Speciality,,30;" _
& "T,Phone,,20;" _
& "T,Mail,,25;" _
& "IB,New,new.png:New receiver,,,updateExpert;" _
& "IB,Delete,delete.png:Delete receiver,,,updateExpert;" _
& "IB,Update,update.png:Update receiver,,,updateExpert;" _
& "Required,Name,Speciality,Mail;" _
& "check,Mail is mail;"
fg.fg(Activity,formParams,Me,"handleParameters","manageDB")
...
Sub updateExpert(Action As String)
      If Not(fg.CheckErrors(Action)) Then Return
      If Action <> "New" Then
...
End Sub
```

☞ `Ok` buttons can have `cancel` instead the function name, in this case no check is performed.

☞ `Cancel` and `Reset` button can have only a message that is shown with a `NO/YES` message where `NO` ignores the request and `YES` executes it (see example below).

The *fontInfo* in the second *extra* field can contains some text properties separated by space:

- *FontSize* a numeric value, the default is `12`.
- *Typeface* the default is `Typeface.DEFAULT` the values accepted are `mi` or `materialicons` and `fa` or `fontawesome` (indifferent to case). ☞ `FontAwesome` font includes the standard Latin characters, ☞ Material Icons only includes the upper case characters.

[1] Since it is not possible to ascertain the presence of files in the assets directory, it is the user's responsibility to ensure the presence of the file.

- *color* see paragraph 1.2.8.2 for the values accepted.
- Gravity information i.e. `left`, `center` or `right` (not applicable to buttons).

Examples:

```
B,Go2,#xF1C0,35,,,fa green 14;
"IB," & Chr(0xF2C6) & ":TELEGRAM,,80,,,fa green 14;"
```

The buttons are inserted at the bottom of the module unless the user wants to insert them after a view.

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

```
...
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 name of an image in the `asset` folder or character that is a simple and efficient way to create buttons with pictures.

Characters can be Unicode character or characters from Fontawesome or from Material Icons, the characters are in the form `#nnnn` or `#xeeee` where *nnnn* is a decimal value of the character and *eeee* is s the hexadecimal value of the character.

```
B,Cancel,#x2718,,,,Exit from visit?;
B,Reset,#x21B6,,,,Clear form?;
B,Start,#9998,,myHandler,Go;
IB,info,#xF05A:Show,,ignore,fa blue 14;
C,cMail,#xE0BE,,,mi blue 20;
```

Table 1: 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
Eye		◉	#x1F441
pound		£	#xA3
right arrow		➜	#x279c
white square		▫	#x25a2
triangle down	#9660	▼	#x25bc

1.2.7.3 Check box

The *extra* field can contains a possible description after the check box; in case of check box set after a view, if *extra* field is not present, the *label* field is used instead.

☞ If checks box is used after a spinner an excessive length can interfere with the spinner.

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

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("Required") = "1","M",""))
```

1.2.7.4 Check box list

`CKL` type generates a set of check boxes.

The extra field contains a list of field separated by , (comma) with syntax: [*key:*]*value*[, [*key=*]*value*[|...; the field name of check box is *key* if present, otherwise is *value*; *value* is the description that appears after the check box.
The*Name* of the check box list is a field that will contain the number of check boxes selected; the default parameter can contain one of the items, possibly others checks must be set by the pseudo type Default, see example:

```
CKL,Lang,Languages,,PHP,'C:C, C++, C#|JS:JavaScript|PHP|PYTHON|RUBY|RUST';
T,Others,,30,,Specify;
Check,Lang < 4,must be from 1 to 3;Check,Lang > 0,must be from 1 to 3;
Default,C:1
```

1.2.7.5 Comment

C|COMMENT,[*fieldName*],*comment*,[*height*],*comment*,[*fontInfo*][,*backgroundColor*]

In the **C** type the comment is contained in the *label* field or *default* field, it occupies the space of label and view. The possible extra field can contain the alignment font and gravity information (see paragraph 1.2.7.2).
height is a fixed dip height of comment, useful for scroll labels.

1.2.7.6 Date

The keyboard contains number and . / and - as separators.

The date inputted is checked when the field loss the focus according to the date format (the default format is MM/dd/yyyy). The default value can be now.

☞ For set another format insert the format in the first *extra* field (is the hint):

```
Title,,Date and Time example;
Ground,Blue 80000;
Date,Date,,,Now,dd-MM-yyyy;
Time,Time;
```

1.2.7.7 File

The initial Folder (and possibly file name) is in *default* field; if it is omitted or if it is not a folder, the File.DirInternal is assumed.

1.2.7.8 Hidden field

Is the type **H** or **HIDDEN**; the syntax is: [H|Hidden],*fieldName*,*value* es:

```
DateTime.DateFormat = "yyyy-MM-dd HH:mm:ss"
...
"Hidden,TimeStamp," & DateTime.Date(DateTime.Now) & ";"
```

The hidden field is returned.

☞ The hidden field can contain an object, in this case it must be set after the creation of the form:

```
...
dim c As Cursor = SQLocal.ExecQuery(cmdSql)
Dim fields As String = "Title,," & table
If c.RowCount = 0 Then
      fgNewForm.fgw(Activity,"C,,No data;",Me,"")
Else
      Dim cursorPosition As Int = 0
      Dim fields As String = "Window,90,,,,shadow,cyan;Title,," & table
      c.Position = cursorPosition
      For j = 0 To c.ColumnCount - 1
            fields = fields & ";U," & c.GetColumnName(j) & ",,30," & c.GetString2(j)
      Next
      fields = fields & ";B,Next,--
>,,,anotherRow;B,Previous,<--,,,anotherRow;H,Cursor"
      fgNewForm.fgw(Activity,fields,Me,"")
      fgNewForm.setValue("Cursor",c)
      Do While fgNewForm.fh_isrunning
            Sleep(100)
```

```
      Loop
End If
...
End Sub
Sub anotherRow(btn As String)
      Dim c As Cursor = fgNewForm.valueOf("Cursor")
      c.Position = (c.Position + IIf(btn = "Next",1,c.RowCount-1)) Mod c.RowCount
      fgNewForm.resetDefaults(genDefaults(c))
End Sub
```

1.2.7.9 Image

`I|Image,`*`name`*`,`*`imageFile`*`,[`*`width`*`][,`*`left`*`][,border [half trasparent|`*`color`*`]|shadow]`

Image or **I** shows an image on form; the *`imageFile`* is a graphic file with path, if there is no path the file must be in the `DirAssets` folder:

example

```
I,/storage/emulated/0/Android/arcadia.jpg;I,sermig.gif;I,galeno.png,300,,border
```

1.2.7.10 Radio buttons

The *length* is the length of the single view, if the *length* is omitted the space is proportional the description length; 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,Single,M:Married|S:Single|W:Widow;
Rdb,AgeType,,10,Y,M:Months|Y:Years;
```

Value can be an image:

```
R,Langue,,13,,FR:frs.png|EN:uss.png|Unknown;
```

☞ The images must be in the folder `DirAssets`; the best height is 24 pixels.

☞ It is possible to obtain the value exposed by means of `valueOf(fh_`*`viewName`*`)`.

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

1.2.7.11 Slider

The *length* is in pixels.

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

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

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

The qualitative seek bar (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
...
```

☞ The possibly view after a spinner (see paragraph After 1.2.8.1) can interfere with the form.

1.2.7.12 Spinners

CMB is a simple spinner, if no member is selected the value returned is an empty string; if the form has only one **CMB** spinner, there are no buttons and the form is exited when a spinner item is selected.

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

The *extra* field can contain:

`%days` is generate a list of days name

`%months` is generate a list of months name

`%#months` is generate a list of months name that returns the number corresponding.

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

The returned value for **CMB** and **CMT** is the key if present. ☞ It is possible to obtain the value of **CMB** exposed by means of `valueOf(fh_viewName)`.

CMX type is a spinner with text, every choice in spinner is inserted in the text, this is useful for example to compile a list of symptoms with a description.

1.2.7.13 Text fields

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

Type **U** and **UN** are fields not modifiable; **UN** data are right aligned.

1.2.7.14 Time

The keyboard contains number and `:` (colon). The value is checked when the field loss the focus; the hour can be entered without the leading zero(es); default can be `now`.

1.2.7.15 Timer

`Timer,`*`name`*`,,[1000,`*`tick`*`],[dis[able],enable],`*`function`*

The extra field can contains a name of the function called when *`tick`* occurs, in the form: *`functionName`* or *`moduleName.functionName`* if it is not in the module which has required the creation of the form.

For default the timer is disabled; it can be started if default is `enable` or by instructions like below:

```
Title,,Timer whith progress bar;
ground,red green;
Window,95,,,20,shadow;
S,pBar,Progress bar;
Timer,Timer,,50,,handleTimer;
B,Start,,,,handleTimer;
```

```
Sub handleTimer(Name As String)
      Select Name
            Case "Timer"
                  Dim seekBar As SeekBar = fg.getHandle("pBar")
                  Dim pBar As Int = seekBar.Value
                  If pBar < 100 Then
                        seekBar.Value = pBar + 1
                  Else
                        fg.disable("Start")
                        fg.disable("Timer")
                  End If
            Case "Start"
                  Dim btn As Button = fg.getHandle("Start")
                  btn.Text = fg.iIf(btn.Text = "Start","Stop","Start")
                  Dim t As Timer = fg.getHandle("Timer")
                  t.enabled = Not(t.Enabled)
            Case Else
      End Select
End Sub
```

1.2.7.16 Toggle

`Toggle` or `TB` is an extension of the toggle view which can have more than two values and supports also images.

The *Extra* field contains a list of toggle elements in the form like spinners items; if it lacks it is assumed `On|Off`.

```
Title,,Toggle example;
Window,95,,,10,shadow;ground,red green;
DN,n2,Decimal number;
B,Function,#x221A;
After,Function,n2;
Toggle,Langue,,,,FR:frs.png|EN:uss.png|
Unknown;
TB,Switch,,,Off;
B,Cancel,#x2718;
B,Reset,#x21B6;
```

1.2.8 Pseudo types

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

1.2.8.1 After

The pseudo field `after` is useful for insert a view at right of another view:

`after,`*`viewNameA`*`,`*`viewNameL`*

where *viewNameA* is the view to be placed at right of the view *viewNameL*.

The possible *viewNameA* views are:

- buttons,
- check boxes,
- radio buttons,
- not modifiable fields,
- text fields,
- toggle.

Buttons can be after a Title.


```
Window,98,,,,,teal;
Title,idTitle,'After, label and icons
example #xF17B',,Orange 18 fa;
Label,,orange fa 11 Right, :;
T,Mail,Mail #xF003,25,ross@lib.it,,Insert
a valid mail;
B,Go,see.png;
check,mail Is Mail;
After,Go,Mail;
N,Age,,5,75,age,Patient age;
B,Go2,#xF1C0,35,,,fa orange 14;
After,Go2,Go;
R,AgeType,,12,Years,Y:Years|M:Months;
After,AgeType,Age;
T,fileName,File name,25;
U,Extension,,,*.png;
After,Extension,fileName;
T,Phone,Phone #xF095;
Check,Id is \s*|\d\d,Nothing or two
digits;
T,Name;T,Id,,5;
After,Id,Name;
B,Cancel,Close,35;
After,Cancel,idTitle;
```

☞ In the list of wiews *viewNameL* must appears before *viewNameA*. and both must precede the `After` pseudo type.

☞ We can also set *viewNameA* after another view aftered (see the example above).

☞ A button aftered works like buttons at form bottom if it doesn't contains a name of callback function.

☞ If *viewNameA* is used after a spinner an excessive length can interfere with the spinner.

1.2.8.2 Background

The pseudo field `Ground` or `Background` create a background in the form or a gradient with a possible effect of transparency; the syntax is:

`[Ground|Background][,`*`fromColor [toColor [direction]]`*`]`

The colors are in hexadecimal notation with two digit for every component: RRGGBB, in this case the color is opaque; the transparency is a hexadecimal value from 00 to FF appended before the color, see the examples below:

```
"Ground;"                 ' default FF404040
"Ground,FF;"              ' blue
"Ground,7F0000FF;"        ' half transparent blue
"Ground,FF 8000 LEFT_RIGHT;"    ' gradient blue to green
```

direction can be:

- TOP_BOTTOM or TR_BL (Top-Right to Bottom-Left),
- RIGHT_LEFT or BR_TL (Bottom-Right to Top-Left),
- BOTTOM_TOP or BL_TR (Bottom-Left to Top-Right),
- LEFT_RIGHT or TL_BR (Top-Left to Bottom-Right).

The default *direction* is TOP_BOTTOM.
Color can be also one of this (non transparent): aqua, black, blue, crimson, cyan, gray, green, lightgray, magenta, olive, orange, purple, red, silver, teal, white, yellow.

☞ By default the form has a gray background; for a transparent background use a 00 transparency, for example: Ground,**00**000000;.

1.2.8.3 Check

This pseudo view is used for execute controls on fields:

```
check,fieldName operator (value|fieldName)[,errorMessage]
```

where operator can be one of =, >, <, <>, >=, <= or is.
After the operator is we can have:

```
required|mail|regularExpression
...
& "T,email,My EMail,20;" _
& "P,password,type password;" _
& "P,repeatPassword,retype password;" _
& "check,email is mail,Incorrect mail form;" _
& "check,password=repeatPassword;" _
& "Check,psw is .{6#44},Password too short;" _
...
```

☞ if *value* contains comma or semicolon they must be masked (see paragraph 1.4.1).

1.2.8.4 Defaults

The syntax is: default[s],*viewName*:*viewValue*[,...]

default[s] is useful for populate the form.

☞ if *viewValue* contains comma or semicolon they must be masked.

1.2.8.5 Dictionary

```
Dict[ionary],function[,param]
```

This pseudo permits translation. function is a custom function that return a map where the key is the English word or sentence and the value is the translation. param normally indicates the target language.

1.2.8.6 Random defaults

```
rdefault[s],ViewName[:ViewRandomValue][,...]
```

The rdefault pseudo type can be used for testing purpose. *ViewRandomValue* can have the syntax:

- n_1 n_2 a random value from integers n_1 n_2 extremes included, ☞ must be $n_1 < n_2$;
- a list of items separated by |.

☞ In case of combo box, qualitative slider and radio buttons only *ViewName* is required.

```
Title,,Random defaults example;
N,Integer,Integer number,7,;
DN,Real,Decimal number,10;
R,Status,,12,,M:Married|S:Single|W:Widow;
QS,Urgency,,9,Green,White|Green|Yellow|Red;
CMB,cmb,Combo box,20,,Alpha|Beta|Delta|Gamma;
S,Slider,Seek 1,6,,10 -10;
RDEFAULT,Integer:-100 +100,Real:17 97,Status,Urgency,cmb,Slider:-10 10
```

1.2.8.7 Label

For manage the view's labels:

`label,,`*`fontInfo`*`,`*`afterLabel`*

afterLabel are the possibly characters inserted after the label.

fontInfo see paragraph 1.2.7.2.

☞ The second value is intentionally empty and it is ignored.

Example:

```
Ground,FF 8000 LEFT_RIGHT;Title,,Label example;
Label,,Right magenta fontawesome, :,;T,Name,,25;
```

☞ For a correct determination of space for labels `label` must precedes the views.

1.2.8.8 Required

A list of fields that mus be inserted:

`required,`*`field`*$_1$`[,`*`field`*$_2$`...]]`

1.2.8.9 Section

`section,`*`sectionName`*`[,`*`condition`*`[,`*`condition`*`[,...]]]`

This pseudo type is for multi-forms, he must precede its views.

The *condition* has the form: *fieldName operator* `[`*value*`|`*fieldName*`]` where operator can be one of =, >, <, <>, >=.

Sections are displayed in the order in which they are present, a `Section` whit condition is displayed only if all condition are verified.

In case of multi section are added two navigation button: `fh_Forward` and `fh_Back` with caption respectively `-->` and `<--`; the `Ok` button is present only on the last section.

☞ the views before the first `section` are repeated on all form.

sectionName is the title of the form section; if it lacks is used a possible `Title` in the section or the possible `Title` before the first section.
See the example at right.

```
Title,,Section's Default title;
Section,;
S,Seekbar1,Seek 1,6,-3,10 -10
Section,Section title;
S,Seekbar2,,6;
Section,Sec 2;
Title,,Inner section title;
QS,Urgency,,9,Green,White|Green|Yellow|Red
```

1.2.8.10 Tab

`tab,`*`name`*`,`*`tabTitle`*

`tab` allows to organize the form through a simulation of `tabHost` view; it can be used in sections; if *tabTitle* is omitted is used the *name* instead; *name* can be omitted, in this case it is set to `fh_tabn`.
Every tab contains the `Reset` button and possibly buttons declared in the same tab; `Cancel` and `Ok` buttons are inserted outside the tabs.

☞ before the first `tab` there can be `ground`, `title` and `window` pseudo type, any other view is ignored, for now, except for `Ok` or `Cancel` button (if you want to change the caption).

1.2.8.11 Title

```
Title,name,formTitle,[backgroundColor,[fontInfo],[hide|back|cancel]
TITLE,Title,Send mail parameters,FF202020,Red 18;
```

The default of *backgroundColor* is transparent, the default `FontSize` property is `20`.

fontInfo see paragraph 1.2.7.2.

☞ The title can be omitted.

- `hide` and `back` insert at right of the title  that permits to hide or send in background the form;
  when the form is sent in background appears at top right of the screen  to which allows you to return the form to the front.
- `cancel` inserts at right of the title  the cancel button.

1.2.8.12 Window

```
Window,[width|100%x],[height|automatic],[left|0],[top|0]
    [,border [half trasparent|color]|shadow][,backgroundColor]
```

Ex. `Window,98,,,,shadow,teal;`

width, *height*, *left* and *top* are in `%`.

If *width* or *height* are lesser than `100%` and *left* or *top* aren't present the form is centered.

if *heigth* is omitted the `height` is chosen to include the whole form.

☞ `border` default is black half transparent.

backgroundColor is *fromColor* [*toColor direction*]

For *direction* see 1.2.8.2 Background.

1.2.9 Returned Values

If the form has been generated by `fg` function the data are accessible in the Sub indicated as third parameter; this sub receives a map of data that are accessible via `Get` or `GetDefault` methods using the field name as key. If the form has been generated by `fgw` function the data are accessible in the same calling function by the function `valueOf(Name)`.

Added fields	
`fh_button`	contains the name of the button pushed (`Ok` or `Cancel` or the name of possibly custom button(s))
`fh_fieldName`	For spinners, qualitative seek and radio buttons contains the value exposed

```
' example of manage data in sub receiver
...
      fg.fg(Activity,prms,Me,"handleAnswer","handleEvents")
End Sub
Sub handleAnswer(fData As Map)
            If fData.Get("fh_button") = "Cancel" Then
...
```

```
' example of manage data in the same sub
      fgNewForm.fgw(Activity,fields,Me,"")
      Do While fgNewForm.fh_isrunning
            Sleep(100)
      Loop
      button = fgNewForm.valueOf("fh_button")
      If button = "Cancel" Then Exit
...
```

☞If Cancel button was pressed there is only `fh_button` field.

1.3 CallBack

FormGen can handle events by the fourth parameters of the `fg` call, or by a sub associated to a button.

1.3.1 Handle Events

This function can be used to personalize the form e.g. modify the state of the view, perform controls end so on. The functions receives an array which contains view name and event, besides, for views, contain the view handle, the value and possibly extra field, see below.

Parameters				
View or state	**Event Name**	**Handle**	**Value**	**Extra - Note**
`fh_start`	`Start`		*`TitleName`*	*`Section`* number
`fh_cancel`	`End`			*`Section`* number
`fh_end`	`End`			Ok button(s)
`fh_reset`	`Reset`			*`Section`* number
`fh_panel`	`LONGCLICK`		`bitMap`	Is the image of the form
`viewName`	`SETVALUE`	*`viewHandle`*	`value`	`setValue` function
`buttonName`	`CLICK`	*`viewHandle`*		
`viewName`	`CHANGED`	*`viewHandle`*	`newValue`	Text fields `oldValue` in fourth cell
`viewName`	`ENTER`	*`viewHandle`*	`Value`	Text fields
`viewName`	`FOCUS`, `LFOCUS`	*`viewHandle`*	`value`	Text fields Focus and lost focus.
`viewName`	`VREND`	*`viewHandle`*	`value`	At end of Voice recognition (if it is active)
`checkboxName`	`CKB`	*`viewHandle`*	`value`	Value = 1 if checked, else 0
`seekbarName`	`SLIDE`	*`viewHandle`*	`value`	Extra is the handle of label that contains the value.
`spinnerName`	`CLICK`	*`viewHandle`*	`value`	Handle to spinner for CMT and CMX type)
`newTab`	`OPEN`	*`oldTab`*		
`Timer`	`TICK`	*`timerHandle`*		
`Toggle`	`CLICK`	*`viewHandle`*	`value`	

The button `CLICK` event precedes the closing of the form, however, it is possible to inhibit the closing by setting the property `fh_yesToExit` to `False`.

```
Sub handleEvents(parm() As Object)  ' widget events handler
  Dim value As String = ""
  If parm.Length > 2 Then
      If parm.Length > 3 Then value = parm(3)
```

```
  End If
  Log("Handle event " & parm(0) & " event: " & parm(1) & " Value: " & value)
      Select parm(0)
        Case "fh_start","fh_reset"
        Case "Message"
            If parm(1) = "VREND" Then
              Dim txt As String = parm(3)
              parm(3) = txt.SubString2(0,1).ToUpperCase & txt.substring(1) &
"."
            End If
        Case "Parms"
            fg.fg(Activity,"Prova,,,t",Me,"handleAnswer","handleEvents")
        Case "Send"
            sendMail(parm(0))
            fg.fh_yesToExit = False
      End Select
End Sub
```

Example of sub for handle events

1.3.2 Handle CallBack

The function is called when a button with call back function (in the *extra* field) is pushed or a timer tick occurs; the function receive the name of the button. The values of view can be accessed by the function `valueOf(viewName)`.

```
Sub Globals
...
   Dim fg As fgen    ' instantiate Form Generator class
...
   Dim Finder As String = $"T,Name;CMB,list,List;B,Find,-->,6,,loadNames;"$  _
             & $"B,New;B,Ok,See,,disabled;After,Find,Name;H,app,listProd"$
...
End Sub
...
fg.fg(Activity,"Title,title,Find Products;" & Finder,Me,"handleAnswer","")
...
Sub handleAnswer(fh_Data As Map)
   If fh_Data.Get("fh_button") <> "Cancel" Then
...
    End If
End Sub
Sub loadNames(btnName As String)
   Msgbox(fg.valueOf("app"),"")
End Sub
```

Sample of Callback function

☞ We can also manage the *CallBack* in the event management function (the fourth parameters of the `fg` call), in this case it is not necessary to specify a dedicated function.

1.4 Remarks

1.4.1 Masking comma and semicolon and insert Unicode characters

If *label* or *default* or *extra* or condition contains commas or semicolons, the function `mask(data)` must be used, commas are replaced by `#44` and semicolons by `#59`, ☞ alternatively the fields can be enclosed in `'` or `"`.

```
Dim cmd As StringBuilder
...
instantiatedName.mask(data_to_be_masked)
cmd.Initialize
```

```
cmd.Append(fh_Data.Get("Name") & ",")
cmd.Append(fg.mask(fh_Data.Get("Label")) & ",")
cmd.Append(fh_Data.Get("Length") & ",")
cmd.Append(fh_Data.Get("Type"))
cmd.Append("," & fg.mask(fh_Data.Get("Default")) & ",")
cmd.Append(fg.mask(fh_Data.Get("Extra")))
```

☞ The fields containing comma, colon or semicolon can be enclosed in ' or ".

1.4.2 Manage Buttons

Form Generator inserts the `Ok` button, the `Cancel` button and the `Reset` button depending on the views contained in the form:

- the `Cancel` button is always present, unless there is only one spinner (type **`CMB`**) or it is stated on title pseudo type.
- the `Reset` button is present if there are data fields,
- the `Ok` button is not present if there is only one spinner (**CMB** type) or there are some others buttons without function associated.

This is controlled by a flag (`lonely`) that is the sum of value assigned to every widget:

- 0 Comment, images and not modifiable widgets,
- 1 simple combo box (spinners),
- 2 widgets that can be restored to initial value,
- 2 buttons that can be enabled/disabled,
- 97 buttons (**B** and **IB** type).

If the form contains only not modifiable views (*type* **U** or **Image or** Buttons without defaults value), there is only the `Cancel` button.

The Forward (`-->`) and Back (`<--`) buttons are present in case of multi sections.

1.4.3 Data presentation

The data are presented in the order they appears in the parameters list, except for the buttons not aftered that appears together the buttons inserted by *FormGen*, at the bottom of the form.

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

With the pseudo type `after` some views are be placed at right of another view.

1.4.4 Work with views

We can modify the view properties accessing the view by the function `getHandle`; therefore for some properties there are specific functions:

- enable view: **`enable`**`(`*`viewName`*`)`
- disable view: **`disable`**`(`*`viewName`*`)`
- get the handle of the view: **`getHandle`**`(`*`viewName`*`)` In case of radio button is the handle of radio button checked,
- get the handle of the view label: **`getLabelHandle`**`(`*`viewName`*`)`
- Change the value: **`setValue`**`(`*`viewName`*`,`*`value`*`)`
- Get the view value: **`valueOf`**`(`*`viewName`*`)`

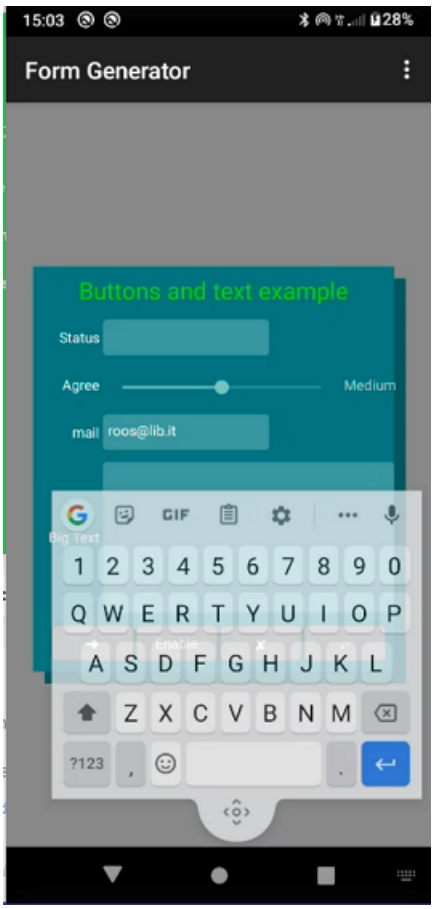
Examples:

```
fg.disable("btnGo")
Dim lbl As Label = fg.getHandle("title")
lbl.TextColor = Colors.Green
If fg.valueOf("Consent") = 1 Then fg.enable("btnGo")
fg.SetValue("Slider",0.2)
fg.setValue("Number",400)
fg.setValue("Spinner","Delta")
fg.setValue("UnMod","**********")
fg.setValue("Rdb","Married")
```

1.4.5 Input text and voice recognition

If the device has installed the Gboard app it is possible insert into a text fields data from voice.
If Gboard it isn't present it can be downloaded from Play.

The board appears with the first line of icons when the text field is empty (however the phone is always present). The three dots icon shows a set of options among which the one that allows to have the floating keyboard (as in the image).



1.4.6 Save form image

The form panel respond to double click, if the form has a handle events function it receives the bit map in the value (see example below) otherwise the form image is written on `File.DirRootExternal` with name `YYYY-MM-DD HH mm ss.png`.

```
...
fg.fg(Activity,formParams,Me,"handleParameters","manageDB")
...
Sub manageDB(parm() As Object)
      If parm(1) = "SETVALUE" Then Return
      Select parm(0)
...
            Case "fh_panel"    ' value contains bitmap
                  Dim bmp As Bitmap = parm(3)
                  saveImage(bmp)
            Case Else
      End Select
End Sub
```

1.5 Others functions and utilities

FormGen module contains, may be, useful functions; some are just seen above at paragraph 1.4.4 Examples.

1.5.1 Add values to spinner

The function `setComboValues(viewName,data)` replace the values of spinners (**CMB**, **CMT**, and **CMX**). `data` is a string of items in the form: `[key:]value`. separated by `|`.
The function returns the number of items.

1.5.2 Dialog method

A dialog can be emulated:

```
Dim ask As String = "Title,,Picture comment;Window,80,35,,,Shadow;T,Comment"
fg.fgw(Activity,translator.translate(ask),Me,"")
Do While fg.fh_isrunning
      Sleep(100)
Loop
If fg.fh_Data.Get("fh_button") <> "Cancel" Then
      ...
End If
```

1.5.3 Right and Left string functions

Right and Left functions returns a pieces of string stripped by some extent:

```
Log(fg.Left(fg.Right("Condor Informatique",12),6))    ' Inform
```

1.5.4 Get filename components

`pathinfo`[2] returns a map containing information about a file ex:

```
Dim fileInfo As Map = pathinfo("/www/htdocs/index.html")
```

- `dirname` `/www/htdocs`
- `basename` `index.html`
- `extension` `html`
- `filename` `index`

1.5.5 Create Image

```
createImage(destinationPanel,imgeFile,Width,Left,Top)
```

show the *`imgeFile`* and returns the `imageView`.
`Width`, *`Left`* and *`Top`* can be `-1` in this case the image is not resized (i.e. the width and height are those of the image) and centered in *`destinationPanel`*; if there is the width the image is created with the height correctly.

1.5.6 Get the widget Handle

The method `getHandle` returns the widget handle (or `Null` if it not exists).

```
Dim cmb As Spinner = fg.getHandle("listProp")
cmb.clear
cmb.addall(fg.splitKeyValue("listProp",fg.implode("|",listExploiteurs)))
```

☞ the handle of a not modifiable field (**U**) is a label handle.

☞ For Combo Text (**CMT**) and Combo Extended Text (**CMX**) the handle is relative to the text associated with the spinner; for access to the spinner:

```
Dim cmbTxt As View = fg.getHandle("Commune")
Dim aWdg() As Object = fg.widgetRef.Get(cmbTxt.Tag)
Dim cmb As Spinner = aWdg(3)
...
```

1.5.7 Extend Log function

`toText` is a function for speedy logs; it has two parameters, the first is a string containing some text and a formatting command (`%` or `%`*`n`* or `%`*`n`*`.`*`d`*), the second is an array of object. Every `%` is replaced by an element of the second parameter. The possible *`n`* is a length of output, the possible *`d`* is the number of decimals to show.

```
Dim aaa()  As Object = Array(3,3.14,"vintun",False,True, 18723)
Dim t As String = "% %6.3 %11 %1 aaaa %3 %12.2"
Log(fg.toText(t,aaa))
```

The result is:

[2] This is similar to PHP `pathinfo` function.

```
3  3.140 vintun      f aaaa tru    18,723.00
```

1.5.8 Implode

`implode(`*`separator`* `As String,` *`obj`* `As List) As String`
`implodeFrom(`*`separator`* `As String,` *`obj`* `As List,` *`From`* `As Int,` *`At`* `As Int) As String`

`implode` function returns a string containing all elements of an array or list with separator between each element:

```
Log(fg.implode("<br>",aaa))
```

The result is:

```
set<br>3<br>3.14<br>vintun<br>false<br>true<br>18723
```

1.5.9 Kill the form

The function `fh_Clear` kills the form.

```
If fgh.fh_isRunning Then
      fgh.fh_Clear
End If
```

1.5.10 Object type

`typeOf` is a function which return a type of object, stripping `java.lang` from what is returned by `GetType` Basic4Android function.

```
Select TypeOf(values(j))
   Case "Integer","Double"
      ...
```

1.5.11 Sand box

The Sand box is a demo of *FormGen* and a tool for testing forms. The initial form, not created by *FormGen,* has a text box and some buttons:

- Add button generate a form for create a view.
- Test button generate a form starting from what is contained in the text box.
- Button for generate samples of selected type.
- Sample button generate a sample form with some views and a CallBack button.

The source of samples is shown in the text box.

2 Technical notes

2.1 Software versions

- B4A Version 11.80

2.2 Errors messages and signals

MsgboxAsync	View *viewName* not exists! After ignored	
	viewName is not a wiew	random defaults
	Errors of check control	
	folder folder not accessible	view type File
ToastMessageShow	*tip*	tip of *texts* view
	This field *viewName* doesn't exits	pseudo type Default
	Session *sessionName* not activated for condition '*condition*'	

2.3 ☞ On device

FormGen has been developed and tested most on a device with:

Name	Screen dimension	dip	note
Alcatel 1SE	720x1256	2	Android 10
Blackview A60	600x1077	1	Android Go
Galaxy A50	1080x1984	2	Good Android 9
Wiko U FEEL	720x1024	2	Good Android 6
Lenovo A396	480x729	1	poor
Mate20 Plus	480x778	1	insufficient Android 6

2.3.1 Tuning

Border width	borderLineWidth	3dip	
Base height for multi line texts	baseHeight	24/1dip	
right position of spinner selector	sSelector	27dip	
Top of tabs	tabsTop	40dip, 0	0 if no title
Button dimension	btnWidth	70dip	
fixed widget distance	deltaY	40dip	
fixed widget height	widgetHeight	deltaY*0.75	
Shadow width	shadowWidth	10dip	
Widgets height	deltaY	40dip	

2.4 Assigned names

fh_section*n*	Section title name, *n* is counter
fh_widget*n*	If there is no widget name, *n* is counter
fh_tab*n*	Tab tag and name if not exists, *n* is counter
fh_*widgetName*	For access a value exposed by radio buttons and spinners

2.5 Maps

Name	Key	Value	Note
allButtons	*ButtonName*	Array(*Caption*,*Event*,*tab*,*buttonWidth*,*CallBack*)	String array
after	*viewAfter*	*viewBefore*	
checks	Field name	control on view([1])	array of comma separated checks
Defaults	Field name	value	For all widgets, empty string for widgets without default
Elem	Field name	*array field description*	
fh_Data	Field name	value	map of data entered
fh_FieldsType	Field name	Field type	
handleButtonsMap	Field name	Button handle	
limits	*fieldName*Min, *fieldName*Max, *fieldName*Delta, *fieldName* (*QS type*)	value items (*QS type*)	Slider limits; for qualitative slider are list of values
mapValues	Field name & value	key	For Radio buttons and Combo box (spinner)
	Field name & key	value	
timersMap	Id	Name	
tipsMap	Field name	tip or label	label after file view for show folder
toggleMap	Toggle name	Values list	
widgetRef	Field name	Id, widgetType, label, extraField,tab	(2) For CMT and CMX extraField is the combo ID

1. checks contains arrays:
   - *operator*
   - (*value*|*fieldName*)
   - possibly error message.

~~1.~~

2. widgetRef contains a reference to all view and **TITLE** pseudo field

   key = *fieldname*, value = array(*Id*, *widgetType*, *label*, *extraField* (see below), *tabPosition*)

   Id is a widget ID, for RadioButtons is a panel container

   - *extraField* is:
   - button a possibly sub for CallBack,
   - seekbar (type **S**) is the ID of the label containing the value,
   - file (type **F**) is the file path,
   - spinner text (type **CMT** and **CMX**) is the ID of the spinner associated,
   - radio button is a radiobutton selected or Null,
   - check box of **CKL** (check box list) is the **CKL** field name.

- Comment is handle to label.

2.6 Lists and arrays

- `rdbList` the data array contains normalized RadioButtons data description.
- `sections` two dimension array (number of sections, 3):
  - section widgets,
  - section name,
  - check condition(s).
- `label` contains label, gravity, pixel dimension of eventual label to add, text color
- `tabAttribs` array of type `tabAttrib`: label, caption, ground color or gradient, height of tab, `lonely` flag.
- `title` array contains:
  - name,
  - caption,
  - background color,
  - font info,
  - possible `hide`, `cancel`, or `button` (this last is inserted by *FormGen* when there is a button after title),
  - title width,
  - title height.
- `LblTitle(5)` item 0 is the form title, possible others four for title Tabs.

3 History

Version 0.3.1.1

- Button with `cancel` in default field for do not show `Cancel` button

Version 0.3.1

- function `implode` accept in addition to the `Arrays` also `Lists`,
- a form with only a single spinner exits when an element is selected.
- pseudo field `GROUND` added

Version 0.4.x

- the ground is set to black, if no ground is present,
- added `checks` pseudo field for add controls,
- add comments,
- the fields describing the view are rearranged, the type is the first item, this permits a neatest evolution of the software,
- eliminated the mandatory (M) character for the fields that must exists, this is replaced by the pseudo type `required`,
- added the pseudo type `defaults` and `after`,
- choice of the caption for buttons `Ok`, `Cancel` and `Reset`,
- pseudo type `Hidden`.

Version 0.5.x

- Dialog method,
- qualitative slider,
- check improved by comparison operators,
- multi form management,
- Graphic button and button with Unicode characters.

Version 0.5.3

- Eliminated BC type

Version 0.6.2

- added the C (comment) type

Version 0.7.0

- Error messaging has been improved,
- added Tab pseudo type,
- minor corrections.

Version 0.7.3

- Improvement of **After** pseudo type: the presence of views involved is controlled and also the not modifiable type (**U**) can be aftered.
- Support of `android:minSdkVersion="5" android:targetSdkVersion="19"` instead of: `android:minSdkVersion="4"`

`Version 0.7.6`

- Added pseudo type `RDefault`

`Version 0.8.0`

- Added `Date` and `Time` wiew,
- fields are checked when lost focus,
- text fields are sentence capitalized,
- some colors are accepted by name,

- added the type **`IN`** integer positive number,
- Sand Box improved.

`Version 0.8.1`

- CMT view can return a key if exists.

`Version 0.8.3`

- Correct placement of button with `after` pseudo type in tabbed form.

`Version 0.8.4`

- Possibility of obtaining the exposed value of a combo or radio button through `valueOf(fh_`*`widgetName`*`)`.
- Esposed the `createImage` function.

`Version 0.8.5`

- the presentation of widgets, in particular buttons, has been improved.
- Radio buttons can accept an image.

`Version 0.8.7`

- `title` can be omitted.
- `tab` has been made more flexible and the possible `title` has been placed above the `tab`.
- `Image` can accept `border` or `shadow`.

`Version 0.8.9`

- Added the `Timer`.
- The Seek Bar (`S` and `QS` view) length can be set in pixels.
- In manifest added: `CreateResourceFromFile(Macro, Themes.DarkTheme)`

`Version 0.9.4`

- eliminated error generated by a more stringent compiler's type check,
- fixed regular expression bug for email checking,
- added background color in `window` pseudo command,
- the *extra* field of type `Comment` can contain the alignment `right` or `center`,
- the attributes can be enclosed in `'` or `"` if they contain comma or semicolon.

`Version 0.10.0`

- the form presentation is done by only one task,
- the event `fh_start` is correctly managed,
- `button` caption, `label`s and `title` can have font `Fontawesome` and `MaterialIcon` by the modification of the *`color`* (now *`fontInfo`*) field which also accepts the font size and font type.

`Version 0.10.2`

- added flavor for build the label from *fieldName*,
- label with newline are accepted,
- correct error on aftering comment,
- correct error on checking items number of check list (CKL) ,
- add `hide` and `cancel` to title,
- added the function `fh_clear` for kill the form.

`Version 0.10.4`

- The check box can be checked also by default value `on`.
- `Ok` buttons can have `cancel` instead the function name, in this case no check is performed.
- ☞ The RDB Length default is now 0 for to make space proportional to the description length
- ☞ added `Back` to title and changed the behavior of hide (now it really hides)

3.1 Differences of version 0.4.x from previous version

3.1.1 Buttons

- Default field of type **B** buttons can contain the disable command, instead of the value returned when the button is clicked; the value returned is the button name.
- Removed the option `cancel` in default field of button for do not show `Cancel` button.

3.1.2 View list

The fields are rearranged, the type is the first item, this permits a neatest evolution of the software.

3.1.3 Others

- Removed functions `stripFile`. it has substituted by `pathinfo` that returns directory, file name and extension.
- Eliminated the mandatory (M) character for the fields that must exists, this is replaced by the pseudo type `required`.

3.2 Differences of version 0.5.3 from previous version

Eliminated the BC button, the call back is implicit if the *extra* field is not empty.

The Tip *pseudo type* has been eliminated, the tip is on the second *extra* field of text views.

3.3 Differences of version 0.6.x from previous version

The form of calling has changed:

instantiatedName`.fg(`*activity*`,`*dataDescription*`,`*callingModule*`,`*subHandleAnswer*`,`*subHandleEvents*`)`

The *callingModule* is an activity module or a class module *subHandleAnswer* and *subHandleEvents* are the name of subs concerned. This change has become necessary to be able to call between modules.

3.4 Differences of version 0.8.3 from previous version

- The mask for comma and semicolon c`hr(1)` and `chr(2)` they are no longer supported.
- The syntax of pseudo type `Tab` is now:
  `tab,`*name*`,`*tabTitle*`[,`*defaultIcon*`[,`*selectedIcon*`]]`.

3.5 Differences of version 0.8.7 from previous version

There is not a default if the `title` is omitted; the `tab` and window management has been made more flexible.

3.6 Differences of version 0.8.9 from previous version

The length of Seek bar (`S` and `QS` view) now is the length in pixels of the slider and not of the text where the value is shown.

4 Known issues

- When the data fields are only not modifiable (**U** type) there is only the `Cancel` button.
- The qualitative seek bar (**QS** type) can't be empty.
- Section must have almost a field tough empty.
- The default of Android spinner is to show an existing item also when it isn't selected.

5 Annexes

5.1 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).

5.1.1 Examples

^\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})	(?=.*\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 12
^[-+]?\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

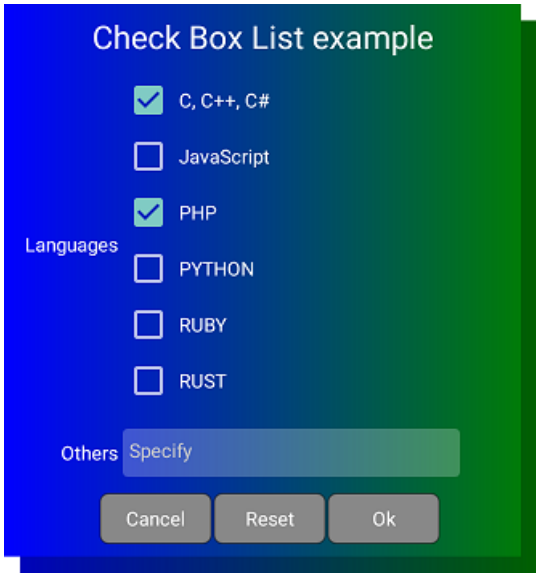
5.2 Gallery

This section shows some particular use of the program.

5.2.1 Check Box List

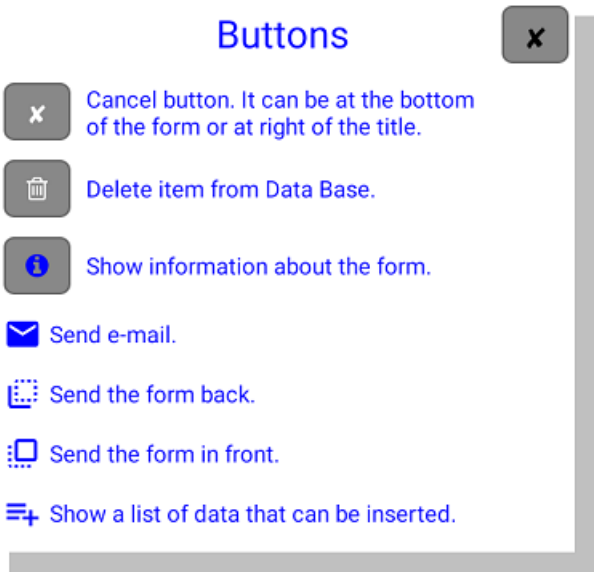
Check box list with control on number of selected items.

```
Title,,Check Box List example;
Window,95,,,10,shadow,FF 8000 LEFT_RIGHT;
CKL,Lang,Languages,,PHP,'C:C, C++, C#|
JS:JavaScript|PHP|PYTHON|RUBY|RUST';
T,Others,,30,,Specify;
Check,Lang < 4,must be from 1 to 3;
Check,Lang > 0,must be from 1 to 3;
Default,C:1
```



5.2.2 Show icons

```
Title,Title,Buttons,,Blue fa,Cancel;
Window,90,,,10,Shadow,white;
Label,,Blue;
IB,canc,#x2718:Cancel button. It can be
at the bottom#10of the form or at right
of the title.,35,,ignore;
IB,Delete,#xF014:Delete item from Data
Base.,35,,,fa 14;
IB,info,#xF05A:Show information about the
form.,35,,ignore,fa blue 14;
C,cMail,#xE0BE,,,mi blue 20;
U,uMail,Send e-mail.;
After,uMail,cMail;
C,cBack,#xE882,,,mi blue 20;
U,uBack,Send the form back.;
After,uBack,cBack;
C,cFront,,,#xE883,mi blue 20;
C,cListSymptom,,,#xE03B,mi blue 20;
U,uListSymptom,Show a list of data that
can be inserted.;
After,uListSymptom,cListSymptom;
U,uFront,Send the form in front.;
After,uFront,cFront;
```



C