Matlab – Octave 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 set of Octave/Matlab script which allows to build and manage forms. *FormGen* is sufficiently generalized for create a set of useful forms from simple message box to complex input forms, based on a list of input controls (some text type, buttons, check boxes, lists, radio buttons...).

1.1 Compatibility

Developed with Octave 6.2.0, MATLAB® R2021a Update 2 May 7,2021.

1.2 Using the form generator

The form builder is contained in `formGen.m` script, which contains the class `formGen`.
Two other scripts `fg_HandleButtons.m` and `fg_HandleEvents.m` contain functions to deal with buttons and events.
The form is generated calling the `fGen` function of `formGen` class:

```
objForm=formGen;
% here we can modify objects properties
fGen(objForm, params, @handleAnswer)
```

or directly:

```
formGen.fGen(formGen, params, @handleAnswer)
```

Where `fGen` parameters are:

- the object itself,
- the list of fields form,
- the function that must be invoked in form closure.

Octave

```
form = ["FORM,,Simulation;T,Name,Simulation name;"...
        "N,Iteration,Iterations number;" ...
        "I,zNumber,Imaginary number;"];
formGen.fGen(formGen, form, @handleAnswer);
```

Matlab

```
form = ["FORM,,Simulation;T,Name,Simulation name;"...
        "N,Iteration,Iterations number;"...
        "%S,s,Comment widget;I,zNumber,Imaginary number;"...
        "L,list,Measure list,,,orko|rino|viale|luino|ferro;"];
formGen.fGen(formGen,join(form,""),@handleAnswer);
```

1.2.1 Properties

The above properties can be set after the object creation.

```
xPos = 10;                  % label position
deltaY = 30;                % control's distance
charWidth = 7;              % medium pixel per character
buttonsDim = 70;            % button's dimension
lineTerminator = newline;   % for text area
checkBoxValues = {'On','Off'};
gfc;                        % figure
```

Example:

```
obj = formGen;
obj.checkBoxValues = {1,0};
fGen(obj, form, @handleAnswer);
```

1.3 Data description

Every control (or widget) is characterized by a list of attributes, separated by comma, in this order: *Type*, *Field Name*, *Field Label*, *Length*, *Default value* and *Extra field(s)*. Controls are separated by semicolons.

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

If *Type* starts with % it is a comment which ☞ must be terminated by semicolon.

☞ The attributes *Field Label*, *Default value* and *Extra field(s)* that contain commas and / or semicolons must be enclosed in ' or " to contain these characters, they can also contain replaceable values identified by the character % (see parag. 1.6.1 Data replacement).

1.3.1 Type

The *Type* is indifferent to the case.

- **Buttons**:
  - **B** button;
  - **R** radio button, a set of Radio buttons;
- **CKB** check box;
- **L** list, Combo box;
- **Text** fields:
  - **S** seek bar or slider;
  - **A** multi line text;
  - **T** text field;
  - **N** integer number, **NS** signed number and **NF** floating number; **I** imaginary number.
  - ~~**U**, **UN** not modifiable field i.e. a protected field, **UN** is numeric for right alignment.~~
  - **H** hidden field.

1.3.2 Field Name

Is the name of the control that, when the form is submitted, it is used to access its value; the name is case sensitive. If no field name is provided the program sets the name in the form: *fieldType_n*.

1.3.3 Field Label

Is the label of control or the caption of button; if omitted it is used the *FieldName*.

1.3.4 Length

Is the length of the control in characters or in pixels for buttons; see the table at right for the defaults length.

A texts area	100	char
B buttons	70	pixels
I imaginary numbers	9	char * 2
N positive integer numbers	7	char
NF floating numbers	9	char
NS signed integer numbers	8	char
R radio buttons	12	char
T texts	20	char

1.3.5 Default

If a numeric field contains two numbers (space separated) the value is a random value inside the two; for imaginary numbers we can have:

One number	random values inside the circle with this radius and centre `(0,0)`
Two numbers	first is the real part, second the imaginary part
Three numbers	random values inside the circle with first value as radius and centre `(second value, third value)`

1.3.6 Extra

Extra field(s) is used for add information to the control, these will be specified in the relative paragraphs.

Type		
Button	**B**	Possible name of handling function, possible tool tip (second extra field)
Check box	**CKB**	Label at right of check box
Combo box	**L**	An item list separated by `comma`: `[`*`key`*`:]`*`value`*`[|[`*`key:`*`]`*`value`*`[|...`
Radio button	**R**	An item list separated by `comma`: `[`*`key`*`:]`*`value`*`[|[`*`key:`*`]`*`value`*`[|...`
Slider	**S**	Start and end value, default is `1 100`
Text	**A, T, N, NS, NF, I**	Tool tip
~~**Unmodifiable**~~	~~**U.UN**~~	~~The value~~

1.4 Summary by type

1.4.1 Buttons

Buttons can be used both for take different actions on form both for show user caption instead of default `Ok`, `Reset` or `Cancel`.

The syntax is:

`B,`*`name`*`,`*`caption`*`,[`*`length`*`],[`*`function`*`],[`*`afterField`*`],[`*`tooltip`*`]`

The *`length`*, if present, is the button dimension in pixels.

The *`default`* field can contains a name of function which is called when the button is pressed.

`afterField` is a name of the widget where on his right the button is positioned; ☞ in the controls list the button must precede the *`afterField`*.

The second *extra* field can contain a tool tip.

For change the caption of `Ok` or `Reset` or `Cancel` the **`Button`** control must be explicitly entered, the syntax is:

`B,[fg_Cancel|fg_Reset|fg_Ok],`*`newCaption`*`;`

☞ **Matlab only!** The Unicode characters are a simple and efficient means to create buttons with pictures:

```
B,fg_Cancel,%10008;
B,fg_Reset,%8630;
B,fg_Ok,%9998;
```

Name	Symbol	Decimal	Hexadecimal
edit	✎	9998	270E
delete	✘	10008	2718
check	✓	10003	2713
check bold	✔	10004	2714
email	✉	9993	2709
cross	✖	10006	2716
dollar	$	36	24

euro	€	8364	20AC
white square	☐	9634	25A2
Ballot box		9744	2610
Ballot box with check	☑	9745	2611

Table 1: Some UNICODE characters

1.4.2 Check box

The extra field of **CKB** type can contain a possible description displayed to the right of the check box.
To set the check box checked the default value must be `On` (case ignored).
The value returned depends on the property `checkBoxValues`.

```
    obj = formGen;
    obj.checkBoxValues = {1,0};     % values for On/Off
    fGen(obj, form, @handleAnswer)
```

1.4.3 List (Combo boxes)

L type is a Combo box (or Drop Down list) that permits to choice a value from a list; the list box can be linked (see 1.5.4 Link) to a text box to simulate a list box with possible text insertion.
The `extra` field contains the item*s* (see description in Radio button).

If there is only one combo in the form, the form does not have buttons and it is exited when a list item is selected; the form is erased.

```
noButtons = "L,Unit,Measure Unit,,,mm:millimeter|cm:centimeter|m:meter|
km:kilometer;";
...
formGen.fGen(formGen,noButtons,@handleOthers);
...
function handleOthers(data)
    data
end
```

Example 1: One choice without buttons

1.4.4 Comment

The label field is the comment shown:

`C,[`*`fieldName`*`],`*`some comment`*`,[`*`length`*`],[after|center|left|right],[`*`afterField`*`],`
`[error|`*`color`*`]`

The first *extra* field can contain the comment alignment, the second *extra* field is a color, `error` is synonymous of `red`.
`color`(s) valid names include: `red`, `green`, `blue`, `cyan`, `magenta`, `yellow`, `black`, `white`, `none` or the hexadecimal value for RGB in the form `#eee`.
`afterField` is a name of the widget where on his right the comment is positioned; ☞ in the controls list the button must precede the *`afterField`*.

1.4.5 Radio buttons

The *extra* field contains the labels and value of each radio button. To obtain a key instead of the label, the item must have the form: *`key:`*`value`.

```
Rdb,Status,,45,M:Married|S:Single|W:Widow
```

If there is only one radio buttons set in the form, this does not have buttons and it is exited when a button is selected; the form is erased.

1.4.6 Slider

The *extra* field can contains the `start` and `end` values in the form `start end`, i.e. `-5 5`; if it is omitted, the values assumed are `0 1`. The result can have decimals depending on the difference from `start` and `end` value, see table at right.

☞ The slider has always a value.

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

1.4.7 Text fields

Numeric fields are **N** (unsigned integer), **NS** (signed integer), **NF** (signed number with possibly decimals) and **I** (imaginary number). The type **A** is a multi line text; if the `length` exceeds 50 characters is generated a multi line text.

☞ In Matlab the possibly spaces at right of the rows of multi line text are trimmed.

The *extra* field, if present, contains a tool tip.

1.5 Pseudo types

1.5.1 Event

`Event,`*`fieldName`*`,`*`eventName`*`,`*`function`*

`eventName` is a user choice name,

`function` receive the control handle, an empty parameter, the object and the *`eventName`*.

```
...
"Event,Blue,changeColor,changeGround;"...
"Event,Green,changeColor,changeGround;"...
"Event,Red,changeColor,changeGround;"...
...
function changeGround(src,~,this,eventName)
    Red = this.valueOf(this,"Red");
    Green = this.valueOf(this,"Green");
    Blue = this.valueOf(this,"Blue");
    h = guidata(src);
    set(h.fg_pnl,"color",[Red Green Blue]);
end
```

Example 2: Function called by event

1.5.2 Form

The type `Form` is used create a panel container:

`Form,`*`name`*`,[`*`caption`*`],[`*`background`*`],[modal|normal],[static],[`*`left`*`],[`*`bottom`*`],[`*`width`*`],[`*`height`*`]`

`caption` can be omitted

`background` default `[0.94 0.94 0.94]`

The fourth field sets the `windowStyle` property.

`static` (case ignored) a `Cancel` button is not generated, ☞ `docked` is not supported.

The form is erased by Cancel button and Ok button(s) unless `static` is present.

The possibly *`left`*, *`bottom`*, *`width`* and *`height`* parameters they are used to impose custom dimensions to the form (figure); in the absence of these `left` is default of the generated figure, `width` and `height` are

changed to the form size and `bottom` is changed if the `height` if the form would appear partially on the screen.

1.5.3 Hidden field

`H,`*`fieldName`*`,`*`value`*

The value of the hidden field is returned on form closed by `Ok` button.

1.5.4 Link

`Link,`*`comboName fieldName`*

This command allows to insert a value taken from a combo into a text field.

1.6 Data presentation

The data are presented in the order they appears in the parameters list, except for the buttons that appears at right of a control.

The buttons inserted automatically (*standard buttons*) are `Ok`, `Cancel` and `Reset` button, they have the name respectively `fg_Ok`, `fg_Cancel` and `fg_Reset`; their presence depends on the controls contained in the form:

- there are no buttons if there is only one Combo box or one Radio buttons set (**`CMB`** and **`R`** types),

otherwise:

- the `Cancel` button is present if the form is not declared `static`,
- the `Reset` button is present if there are data fields (e.g. Type *`texts`*, **`R`**, **`CHK`**, **`L`**, **`S`**, etc.),

1.6.1 Data replacement

`%`*`nnnnn`* ☞ **Matlab only!** *`nnnnn`* is a *Unicode* value (from 2 to 5 decimal digits).

1.7 Button action

1.7.1 Ok button

The *`handleAnswer`* function (the third parameter of the form generator) receive a structure where the key is the widget name and the value is the widget contents; there is also a field `fg_Button` with value `fg_Ok`.

1.7.2 Cancel button

The *`handleAnswer`* function receive a structure containing only the `fg_Button` with value `fg_Cancel`.

1.7.3 Reset button

This button restore the initial values, in case of random numbers those are recalculated.

1.7.4 Other buttons

The function called by those buttons receive the parameters: *buttonHandle*,~,*formObject*.

```
...
     "I,Imaginary,Imaginary number,,5 3 -7;"...
     "B,bSqrt,Sqrt,40,Sqrt,Imaginary,Square root;"...
...
function Sqrt(src,~,this)
    buttonName = get(src,"tag")
    iNumber = this.valueOf(this,"Imaginary")
    sqrti = sqrt(iNumber);
    sprintf("Square root of %f%+fi is %f%+fi",real(iNumber),imag(iNumber),...
         real(sqrti),imag(sqrti))
end
```

Script 1: Function associated to a button

1.8 Errors

1.8.1 Console logged errors

`Default of fieldName: 'defaultField' is not numeric`	Numeric fields
`Event in non existent field: fieldName`	Event pseudo type
`Link widget fieldName doesn't exists`	Link pseudo type
`Slider default fieldName outside range`	Slider
`Unknown color: color`	Form and comments
`Unknown type: type`	

1.9 Some functions

`formGen` contains some static functions:

formGen.functionName(*parameters*)

1.9.1 Left part of string

string = fg_Left(*string*, *quantity*)

1.9.2 Control value

value = *objForm*.valueOf(*this*,*fieldName*)

```
function Sqrt(src,~,this)
    iNumber = this.valueOf(this,"Imaginary");
    sqrti = sqrt(iNumber);
    t = sprintf("Square root of %f% +fi:\n",real(iNumber),imag(iNumber))
    r = sprintf("%f %+fi",real(sqrti),imag(sqrti))
    msgbox(strcat(t,r));
end
```

Example 3: Square root of imaginary: use of valueOf function

1.9.3 Number inside

***boolean* = fg_Inside(*a*, n_1, n_2)**

returns `1` (`true`) if *a* is inside n_1 and n_2 included.

1.10 Data Control and form submission

Before the submission the numeric data are controlled, in case of error(s) the form is not submitted and the field(s) in error are bordered in red and it is generated an alert.

Form data are sent when the `Ok` button is pressed and there aren't errors.

The form has also these fields:

- `fg_Button` contains the name of the button which has submitted the form or, in case of single combo, list or radio, the name of the field.
- `fg_Tag` in case of form cancelled it contains `fg_Cancel` or `fg_Form` depending on how the form has been closed.

2 History

0.1.0	August 2021	First version.

3 Technical notes

3.1 Differences between Matlab and Octave

This paragraph describes the activities to make the product usable in the two environments.

Matlab Functions must be in file *functionName*.m

Octave The char function accepts only integers less than 256

Octave warning: set: allowing tooltip to match uicontrol property tooltipstring

Octave warning: set: allowing unit to match uibuttongroup property units

3.2 Controls

Control	Tag	Userdata
Buttons	field name	
Check box	field name	cell with value returned, default is {'On','Off'}, native values are {1,0}
Popup list	field name	handle to possibly text linked
Radio button group	field name	array of radio buttons handle
Radio button	Field name	progressive from 1
Slider	field name	struct text handle, min, Max, decimals (handle, m, M, decimals)
Others	field name	

3.3 Structures and variables

name	content	key	value
buttons	Buttons properties	*buttonName*	1. Associated function 2. String (Caption)
gData.Events	Type and function handler	*fieldName*	.handle .eventName
Links		*fieldName*	Text linked
pnl	Panel handle		
Gdata.itemsList	Lists, radio buttons	*fieldName*	Array of keys
gData.data	Widget info	*fieldName*	See 1.
Gdata.fg_pnl	Figure handle		
Gdata.staticForm			S for static form
plcHolders	Replacement for quoted items	Phn *n* is progressive	replacement

1) controls data:

1. *Type*
2. *Field Name*
3. *Field Label*
4. *Length*
5. *Default*
6. *Extra*
7.

8. label handle
9. widget handle
10. II widget handle (imaginary number part)
11. error flag
12.

4 Annexes

4.1 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 few examples.
The regular expression is contained between `/ /` and can be followed 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.

`(...)` Parentheses are used to "extract" part of the content and / or to isolate sub-parts of the content. If the sub-part is not to be extracted the syntax is: `(?:…)` see in the examples below **Floating values**.
`?=`*`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).

4.1.1 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 20
`^([-+]?\d{1,2}(?:\.\d{1,2})?)$`	**Floating values** `[-+]?` the sign is possible, **`\d{1,2}`**, one or two digits `(\.\d{1,2})?`, possible decimal point followed by one or two digits. The number is captured, the decimals aren't captured for the presence of `?:`.

`(?=.*\d)(?=.*[A-Z])(?=.*[a-z]).{6,12}`	At most one digit, one capital letter, one minuscule and from 6 to 12 characters
`^#[0-9a-fA-F]{3}$`	Check if there is exactly 3 hexadecimal digits

5 Indexes

5.1 List of Examples