InFormBuilder v1.81 by Steen S. Schmidt

November 26, 2006

Contents

Disclaimer & Copyright

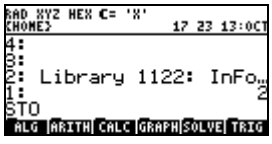
This program is freeware. You are free to distribute this program to anyone, as long as this document is included.

I cannot take responsibility for any damage or data loss caused by this program – it is written in 100% SysRPL and there could be bugs in it. This also means that it will not run on the HP48 series calculator. Because of the adaptation to use the full 131x80 pixels display of the HP49G+ and HP50G, this version will only run on these two calculators. For use on calculators with 131x64 pixels displays (HP49G for instance), please use version 1.7 of InFormBuilder (also included in this package).

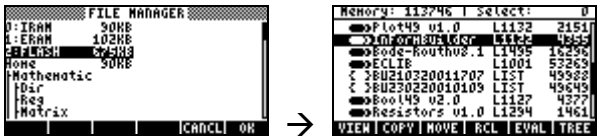
All screenshots in this document are from version 1.7 (and therefore in the 131x64 pixel format), but the changes are mostly cosmetic.

Requirements & Installation

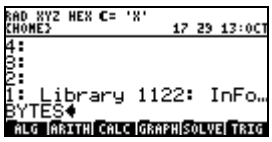
You need to copy the library (library #1122, filename; IFBv181) to the calculator (with the SD card for example) and store it in a port (any port except port 3 should do).
To store it in port 2 for example, you will need to recall the library to the stack, type 2 and STO.

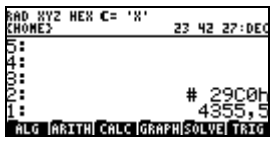


Hold down ON and press F3, release both keys and the calc should warm boot (warning: you'll loose all stack contents, but calc memory will remain intact). When this is done, your port 2 should contain the entry for the library (press left shift APPS to enter the Filer):



Running BYTES on the library on the stack should yield #E108h and 4362.5 bytes.



→


(this screen shot is showing the checksum of v1.7!)

This program works in RPN mode only, since the in- and output heavily depends on stack structure.

The ROM version on the HP49G+/HP50G has to be at least v2.00, and the extable has to be installed too. If these requirements are not met, IFB will not function properly.

The extable library is an external library which contains the mnemonic/address relationships of the Saturn SysRPL and ML languages. This library is necessary for the machine to interpret SysRPL (and ML) source code. You can currently get the extable here:

http://www.hpcalc.org/hp49/programming/entries/extable.zip

In the above .zip file you need to extract the file named 'extable.HP' and rename it to 'extable'. Now transfer it to the calc and store it in any port as you did with the library itself.

InFormBuilder

On the HP graphing calculators it is possible to make InForm boxes using the INFORM command. These boxes can be used to enter data into a running program for example, but are limited to Data Fields (the difference in fields will be explained later). The placement of these fields and their corresponding labels are also very limited – you are at the most able to decide the number of columns your InForm box should contain.

In SysRPL it's possible to do a lot more than that. You have access to 4 types of fields (Data Fields, Extended Data Fields, List Fields and Check Fields) and labels. These fields and labels can be placed anywhere you like, and you don't even have to have the same number of fields and labels.

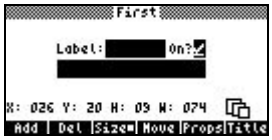
The UserRPL InForm box is probably the command with the most complicated stack syntax, but it's nothing in comparison to the SysRPL one. To make an InForm box with for example 3 labels and 5 fields, you'll have to prepare 78 arguments! This InForm box will then return 1 or 6 arguments – arguments that can contain SysRPL code itself, which is not valid input for UserRPL programs.

IFB is now able to make this easier:

It will let you draw your InForm box in a graphic environment, by adding and deleting fields, changing field properties and moving and resizing the different fields pixel by pixel on the screen.
When you're done with this you'll get the option of either have the program return SysRPL source code for you to incorporate in your own SysRPL project, or have it return compiled code which can be stored and called by UserRPL programs and that will have UserRPL compatible output.

1:

2:



3:

The pictures above show how it could look if you made an InForm box using IFB. Picture **1** shows the environment as it looks when you start IFB, picture **2** shows the environment when you're done

editing and picture **3** shows the final InForm box when it's compiled and running (field #3 is highlighted).

Modes

The small icon on the lower right, just above the menu bar, shows the mode IFB is currently in. Two modes are available:

Resize mode:
This mode allows you to resize the current field with the arrow keys. Labels and Check Fields are not resizable, and this mode will not be available when such an object is selected. The upper left corner of the object will remain on the same coordinates of the screen, while the left-arrow key will decrease the length of the field, the up-arrow key will decrease the height of the field, the right-arrow key will increase the length of the field and the down-arrow key will increase the height of the field. It's not possible to increase a field's size beyond the edges of the screen. Row 46 is the lowest pixel row available to fields in an InForm box, so the available screen ends a bit above the menu (I could have allowed fields to extend all the way down to the menu, but they wouldn't show up in the compiled InForm box anyway – there has to be room for the help string). The smallest size a field can have is 1x1 pixel.

Move mode:
This mode allows you to move the current object around on the screen with the arrow keys. It's not possible to move an object beyond the screen, and the limitations regarding positioning on the screen for Resize mode is also valid for Move mode.
The arrow keys move in their respective directions, of course.

Keys

16 keys are active when IFB is running:

ON:
Turns the calculator off (right-shift isn't necessary). ON will turn it on again.

ENTER:
Ends the edit session with a choose box like this:

Warning: Canceling this box will return you to the stack without output. All work done in IFB will be lost!

"**Data → Stack**" will dump the current IFB data on the stack like described in the STO key section. Choosing "**SysRPL Source**" will build a source file (a string), and choosing "**Compiled Code**" will output a compiled code object. The middle option is probably only interesting to SysRPL programmers.

The Output section provides a closer look at this.

STO:
Dumps the current IFB data on the stack in a list. This list must not be changed if you'll retain the possibility of reloading the environment later. Shows the message "Data dumped to stack…" briefly in the top of the screen.

VAR:
Reloads a previously dumped data list from stack level 1. If the stack is empty or the list appears not to be a valid IFB data object, this message appears and the stack is not altered:

Warning: A comprehensive test sequence is executed to assure that the fetched object really is a valid data list, and no object dumped by IFB should cause any trouble. There is, however, a great risk of data loss if this test system is circumvented by manipulated data lists – create your own lists or edit dumped ones at own responsibility!

Compiled code cannot be reloaded into IFB.

TOOL:
This will select the previous object to be the current object. The current object is the one that will receive your commands, it's the one that'll move or get resized when doing that, and it's the one which properties will be edited with the Props key or which will be deleted with the Del key.
Previous and next in this context should be understood as the order in which they were added to the project, not in the order the objects appear on screen – the objects are only sorted in screen-appearing order upon exit to keep speed up during edit. Objects will flash once when selected.

NXT:
This will select the next object to be the current object (see the TOOL key section).

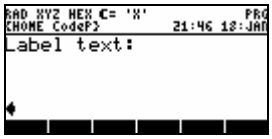
Arrow Keys:
The 4 arrow keys have different actions based upon the mode of IFB. Their different actions depending on mode are discussed in the Modes section.

Menu Keys:
The 6 menu keys have the following function:

Add Add Object:
Lets you add new objects to the InForm box via a choose box:

This choose box let you add any of the 5 possible objects to the InForm box. They will be placed in the top left corner of the edit area (just below the title line) with Move mode selected.

Before this you'll be presented with an appropriate input screen where you can set the object's properties – canceling this will cancel the addition of the object. See the Output section for more detailed description of the fields:



Label:

Here you just enter the label. It will be truncated to a maximum of 32 characters and displayed with the minifont. When the edit line is not empty, CANCEL (ON) will clear the line, else it will exit the entry entirely.

Data Field:

The 12 Check Fields under "Types:" will define what object types are valid to enter into the Data Field. If none are checked, all types are allowed. These fields will correspond to the ***AllowedTypes*** part of the SysRPL definition in the Output section. If all types are allowed, this will have the value **MINUSONE**, else it will be a list including the user types as bints. For example, a checkmark in "Real number" and "Unit object" will result in **{ 0 13 }**.
The "Algebraic" check field lets you decide whether the Data Field should insert algebraic tick marks in the edit line when inputting data.
"Help:" lets you enter a help string to be displayed at the bottom of the InForm box when this data field is highlighted (***"HelpString"***).
"Init:" is the initial object to be displayed at InForm box startup (***InitValue***). It will also be the object displayed after the field has been reset (***ResetValue***).

Extended Data Field:

The description of the normal Data Field also applies to this one, except that you haven't got a choice between Algebraic tick marks or not. The chosen types will limit what variables are visible when browsing in the FILER.

List Field:

The "Data:" field contains the ***ChooseData***, and must be of the form {{"Label" Obj}…}. "Help:" lets you enter a help string to be displayed at the bottom of the InForm box when this data field is highlighted (***"HelpString"***), just as in the other input screens.
"Init:" has to be an object contained within the Data list, and will be the initial object to be displayed at InForm box startup (***InitValue***). It will also be the object displayed after the field has been reset (***ResetValue***). If it's left blank, it'll default to the first object in ***ChooseData***.

A List Field will return a complete list as output. That is, if one enters {{“Flag 1” << 1 SF >>}{“Flag 2” << 2 SF >>}} in the “Data:” field, and later on chooses option 1 (“Flag 1”) in this field in the compiled InForm box, you’ll receive the whole list of option 1 as output from that field: {“Flag 1” << 1 SF >>}.

Check Field:

This fields almost explains itself, doesn’t it ☺

DEL Delete Object:
Will delete the current object. Warning: IFB doesn’t prompt for confirmation!

SIZE Toggle Resize mode:
Sets Resize mode (SIZE) if Move mode was active, otherwise it clears Resize mode (SIZE) and sets Move mode. Labels and Check Fields cannot be resized.

MOVE Toggle Move mode:
Sets Move mode (MOVE) if Resize mode was active, otherwise it clears Move mode (MOVE) and sets Resize mode.

PROPS Edit Object Properties:
Allows you to edit the current object’s properties – you’ll be presented with the same input form (depending on current object type) as when adding a new object (reflecting the objects current properties though). Canceling this input form (or exiting it with an error – see the Error Messages section) will leave the object unchanged.

TITLE Edit InForm Title:
Allows you to edit the InForm box title – an empty title (“”) is allowed. A maximum of 31 characters can be displayed – if the title exceeds this, it’ll be truncated to 30 characters and an ellipsis (…) will be added.

Output

When working in IFB, you’ll cycle through the added objects in the order in which they were entered. Upon exit the order is sorted (order of the labels isn’t touched) in an uppermost/leftmost order. This is so the final code will initialize the cursor at the first field, and the arrow keys will cycle the InForm box in a ‘normal’ way.

The “**SysRPL Source**” exit option will build a source file in this form:

!NO CODE
!RPL
::
n label definitions
m field definitions
***#n #m* ‘DROPFALSE**
“Title string”
^IfMain

;
@

The italic parts are built from your choices in IFB (this is also the case in the definitions below). There has to be at least one label and one field to create a successful source file. As long as this requirement is fulfilled, there's no limitation on the number or mix of fields and labels in one InForm box.

Definitions and descriptions of the different fields (the different properties are explained in greater detail in the Keys section - ADD):

Labels:
These are text strings that are displayed with the minifont. Their definition within the SysRPL code looks like this:
"Label" #x #y

Data Fields:
These are fields that can contain any type of data entered into them, as long as the entered data obey the type properties of the particular field. Their definition within the SysRPL code looks like this:
'DROPFALSE *#x #y #length #height ONE/THREE* AllowedTypes* FOUR *"HelpString"* MINUSONE MINUSONE *ResetValue InitValue*
*This value will be **THREE** if the "Algebraic" property is checked, otherwise **ONE** (see the Keys section - ADD).

Extended Data Fields:
These fields are similar to ordinary Data Fields – you can enter data into them as usual, limited by the allowed types. On top of this you can also choose data from the FILER (also among allowed types). Their definition within the SysRPL code looks like this:
'DROPFALSE *#x #y #length #height* TWENTYTHREE *AllowedTypes* FOUR *"HelpString"* MINUSONE MINUSONE *ResetValue InitValue*

List Fields:
List Fields allows you to choose among a predefined set of options, each one outputting a list – for example containing a program that can be run afterwards to reflect the chosen value. Their definition within the SysRPL code looks like this:
'DROPFALSE *#x #y #length #height* TWELVE MINUSONE SEVENTEEN *"HelpString" ChooseData* SEVENTEEN *ResetValue InitValue*

Check Fields:
These fields will offer a checkmark option. Their definition within the SysRPL code looks like this:
'DROPFALSE *#x #y* SIX NINE THIRTYTWO *AllowedTypes* MINUSONE *"HelpString"* MINUSONE MINUSONE *ResetValue InitValue*

The "**Compiled Code**" option will output a compiled code object. This object can be saved in a variable and for example be used by a UserRPL program. The code, when executed, will launch an InForm box similar to the one drawn in IFB, which takes no arguments and has 2 possible output, depending on how it was exited:

If exited with CANCEL it'll leave a real zero (**0.**) on stack level 1, and nothing else.

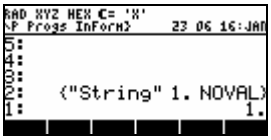
If exited with OK or ENTER, it'll leave the real number 1 (**1.**) in stack level 1, and a list of values from the fields in the InForm box in stack level 2. This list will be in sorted order as discussed in the section above.
Empty fields will leave **NOVAL** in the list, Check Fields will result in **1.** representing a checked field or **0.** representing an unchecked field.

An example of output from an IFB box:

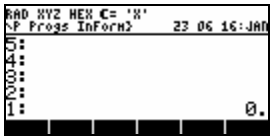
1:

2:

3a:



3b:

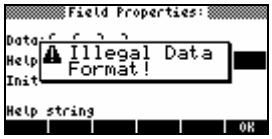


In picture **1** we see an InForm box made with IFB, picture **2** shows values entered in two of the fields; the top Data Field and the Check Field. The third field is empty. Picture **3a** shows the output if one presses the OK key. Notice the order in which the data is sorted in the list, that the Check Field yields **1.** and that the empty field yields **NOVAL**. Picture **3b** shows the InForm box exited with CANCEL. This behavior is just like the built-in INFORM command.

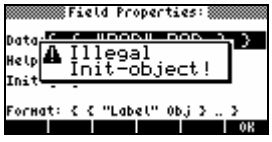
The compiled code has to be accessed through a variable, and cannot be included in a normal UserRPL program, since the built-in UserRPL compiler can't handle the program correctly. A shrewd programmer will of course be able to use this code directly in both User- and SysRPL code ☺.

Error Messages

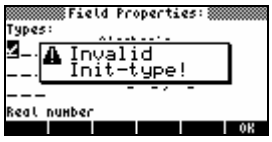
Here's a listing of the possible error messages you can encounter while working with IFB. For the 5 messages accessible through both Add and Props ("Illegal Data-format", "Illegal Init-object", "Invalid Init-type", "Null Data" and "Zero-length Label"), one of two things will happen:
If the shown input screen was accessed through Add, you'll be returned to the edit environment without any object being added. If the shown input screen was accessed through Props, you'll be returned to the edit environment without any changes to the object's properties taking effect.



Illegal Data Format:
The Data property of the List Field has to have this format: {{"Label" Obj}{"Label" Obj}…}. If it hasn't, this error will occur.

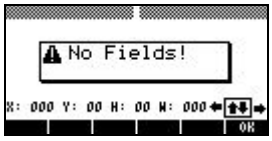


Illegal Init-object:
In a List Field the Init-object has to be represented somewhere inside the Data property list (or be blank). If it isn't in the list, this error will occur.



Invalid Init-type:

In a Data Field or Extended Data Field you can choose valid input types. If the Init-object isn't of a valid type, this error will occur.



No Fields:
IFB was exited without at least one field.

No Labels:
IFB was exited without at least one label.

Not a valid IFB data object:
There was no valid data list on stack level 1 to reload, when VAR was pressed.

Null Data:
A List Field has to have at least one Data element to choose from. An empty list in the Data property will result in this error.

Truncating Label:
When editing a Label (through PROG), you will get this error message if you enter text of a length that will exceed the boundaries of the screen when returned to the edit environment. If so, the text is updated, but truncated to the maximum length that will fit inside the window.

Zero-length Label:
This error occurs if you try to add a Label with no characters in it (or try to change a Label to "" with PROG). You can use the SPC key to create blank labels.

Revision History

v1.0: Initial public release.

v1.1: Fixed bug when Init field of List Field was empty.
Fixed compilation bug of UserRPL types entered in Fields.
Added STO/RELOAD functionality.
Changed program so that it requires ROM v1.16 to work at all.

v1.2: Fixed bug when the “current” object wasn’t the last object when reloading data with VAR.

v1.3: Fixed bug with ***InitValue*** in List Field.
Changed library number to 1122.

v1.4: Changed all internal references to lib ID 1122 too ☺

v1.5: Moved Select mode to its own keys (TOOL & NXT).
Now shows position and dimensions of current object on screen.
Added “Data → Stack” exit option.

Thanks to Carsten Dominik for these suggestions.

v1.6: Reduced size of library with more than 55%.
Reduced UserRPL output with more than 60 bytes.
Removed ROM version and extable presence tests.

Thanks to Wolfgang Rautenberg for great help with this.

v1.7: Fixed bug when resizing objects.

v1.8: Adapted software to use entire 131x80 display on HP49G+.

v1.81: Changed library to use the new inform engine which enables labels to be drawn on the lower part of the 131x80 display too.