iziBasic v2.0

September 4, 2004

Index

What is iziBasic?

Notice: in French "*izi*" is pronounced like "*easy*" in English

iziBasic stands for easy Basic for Palm. It targets all kinds of developers and should be a very good tool for newbee programmers. Skilled programmers will also find in iziBasic a tool to develop very quickly and easily various types of software.

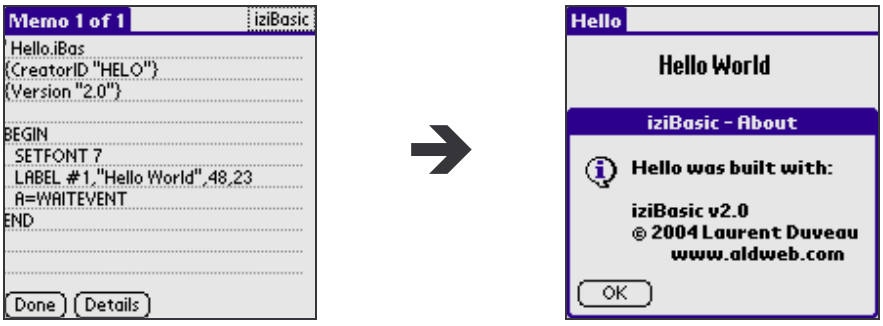
iziBasic is a high level development compiler which builds Stand-alone applications. The great thing is that it does all of that directly on-board of your Palm OS based device. This Stand-alone application builder is very convenient for those wishing to distribute their creations made with iziBasic in a ready to run software.

Source codes are easily written using:
- either the Memo Pad application which is shipped with all Palm OS devices,
- or an on-board third party DOC editor of your choice.

iziBasic reads the source codes, compiles them and builds the applications. The use of the Memo Pad is very convenient as you do not need to install any additional editor software to iziBasic. The 4096 characters limit of a memo is not an issue as iziBasic is able to link (or “chain”) together up to 10 memos to build a bigger application. But, for your convenience, you may as well use a DOC editor which overpasses this 4096 characters limit.

As its name also states it, iziBasic uses the BASIC high level and very easy to learn development language, a customized subset of it to be precise. You will discover how easy and quick it is to develop software with iziBasic when the more common development tools available on the Palm OS platform usually require pretty good development skills.

Easiness of the Basic language and Palm hosting are not gained against execution speed of compiled programs. According to my Bench2 analysis (http://www.aldweb.com/articles.php?lng=en&pg=24), iziBasic is one of the Palm hosted tools which generates the fastest programs. It is in 5th position, behind the PP and OnBoardC compilers, PLua and my LaFac-HELP which are runtime based like iziBasic. And iziBasic is the fastest Basic development language based!



Contact Information

- ✓ World Wide Web main download Site : http://www.aldweb.com
- ✓ Author e-mail : info@aldweb.com

iziBasic is a shareware.

The limitations of the trial version are very light, so as not to stop you from using iziBasic if you like it and cannot afford to buy it.
For instance, there is no time limit and you can very well use iziBasic without any time restriction.
But you are encouraged to support this shareware and buy it.
The limitations are:
- A nag screen to remind you to buy the full version
- Some functionalities are not activated (a few instructions and statements)
- The About box in your programs tells that your software was made with iziBasic

To get a full version of iziBasic, please refer to the iziBasic.txt file that was shipped together with this software or look for iziBasic on my web site (http://www.aldweb.com) and follow instructions.
The cost of iziBasic is just as little as $15.

When you register, you receive a full version of iziBasic that you just need to install on top of this current trial version.

Thanks for purchasing iziBasic.

iziBasic users group

Upon the request of many iziBasic early adopters, I have set up a forum on my web site (**http://www.aldweb.com**) to give you an opportunity to exchange tricks, ask for help & support and share whatever else you would like with the growing community of iziBasic developers.

I will be on the iziBasic forum very often too. So, as to have most people benefit from good ideas, upgrades features requests, beta versions… please use this iziBasic sharing forum rather than sending me e-mails. Then, everybody will get the benefit of your input ☺

How to install iziBasic?

There is nothing special to say here.

iziBasic is a PRC file that is installed like any other Palm file using HotSync.

So, extract **iziBasic.PRC** from the ZIP archive file.

Double-click on it and the Palm install tool will popup.
iziBasic.PRC will be transferred to your Palm device next time you synchronize your Palm with your PC with HotSync.

Please uninstall any previously installed version of iziBasic before installing this one.

- ✓ **Minimum Palm OS requirement for iziBasic is version 3.0**
- ✓ **iziBasic is Palm OS version 5 compliant**

Versions history

v2.0 (09/04/04)
- Bug fix: ABOUTBOX now accepts correctly from 1 to 3 string parameters, when it was only working with 3 parameters
- Bug fix: in some weird cases, iziBasic was crashing when being launched
- Enhancement: the database backup flag is now set by files creation statements, so that the databases created by an iziBasic program will be backed up to the desktop computer
- Enhancement: the backup flag was also set for the programs compiled by iziBasic so that your wonderful creations will be sent during the next hotsync to your computer
- Enhancement: the PENX and PENY values are also updated when the pen first touches the digitiser and when it is moved on the screen (previously only when the pen was lifted from the digitizer), also added a PENDOWN function to track the pen status
- Optimization:
    - of iziBasic's compiling engine which builds smaller executable codes
        the generated p-code is ~30% smaller in average
    - and of iziBasic's virtual machine which runs faster
        the Bench1 benchmark returns a 4.7% speed improvement
        the Bench2 benchmark returns a 3.4% speed improvement

    Note: more optimizations can be expected in a future release as I found other areas of improvement
- Added a few new math functions: ACOS, ASIN, DEGREE, LOG, POWER, RADIAN
    Note: other mathematical functions could be made available (like CosH, ArcSinH, DMS2Deg, Rad2DMS, etc…). Please feel free to ask for them ☺
- Added CALLPP$ function, this is a PP Code Segment call feature which opens all possibilities of the Palm development by direct access to the Palm OS APIs (in other words, you can now include "PP applets" in an iziBasic program)
- Added COLOR(v|n) function to capture forecolor and backcolor
- Added WAITEVENT function
- Added GRAFFITISHIFT, PUSHBUTTON and TEXTSELECTOR objects
- Added UPDATEFIELD, UPDATELABEL, UPDATEPOS and UPDATETEXT statements
- Interface improvements:
    - iziBasic now remembers the last compiled source code name, so as to ease the build and try procedure
    - if compilation succeeded, iziBasic now offers to run the just built program
    - added a button to launch your favourite editor (Memo Pad or one of these DOC editors: PIAF, QED, SiEd, SrcEdit)
- Updated and upgraded this documentation file in many areas
- Made various upgrades in the sample programs and added a new iBHelloPP demo source code

v1.0 (07/22/04)
- initial release for Palm OS

How to use iziBasic?

I'll be very quick in these explanations as iziBasic behaves like all Palm OS based applications, so it is very intuitive to use. ☺

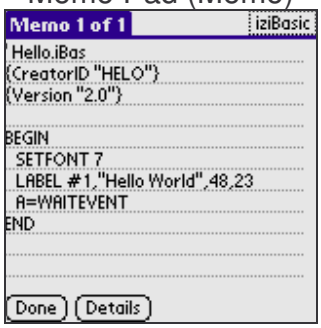
Using the Palm built in Memo Pad or a DOC editor to write programs:

iziBasic searches for the source codes in the Memo Pad database (MemoDB.pdb) and among the DOC files.

On the right side, you see an example of the classical *Hello World* program written in the customized Basic understood by iziBasic.

All Palm OS based devices are provided with an inbuilt Memo Pad application. This lets you write single programs up to 4096 characters. Single programs can be linked (or "chained") to build bigger programs.

Memo Pad (Memo)



Using a third party DOC editor (like PIAF, which is a freeware to be found at http://ppcompiler.free.fr/) is just as easy. You will then be able to overpass the 4096 characters limit of the Memo Pad (some third party extended Memo Pad software do it also but they are not supported by iziBasic).

PIAF (DOC Editor)

ⓘ iziBasic does not recognize the DOC compressed format, only the uncompressed format.

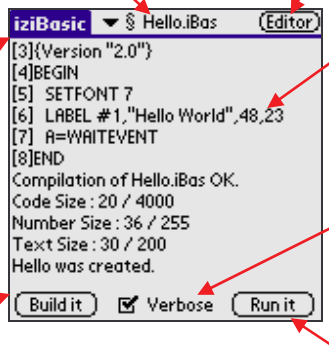
Once your programs are written, you want to run them. This is when iziBasic comes to be useful.

Using iziBasic to compile source code:

iziBasic lists in this popup list all source codes found in your device.
The glyph character in front of the source code name tells if the file is:
- a DOC file
- a Memo §

If you tick on the iziBasic title, you open the About box from which you can access the Options box

As its name states it, the Build it button launches the compilation of the source code selected in the top-middle popup list



The Editor button lets you launch your favorite editor which you defined in the Options box (accessed through the About box)

Main screen display used to display information about compilation

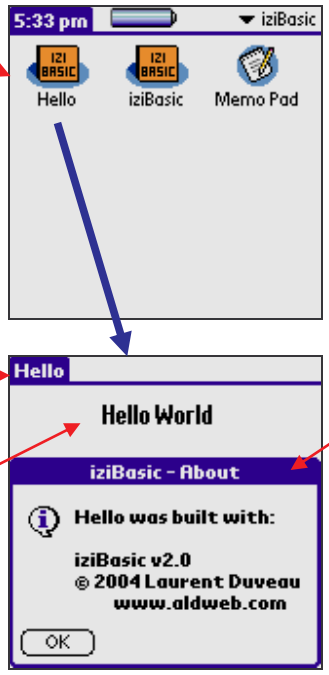
If checked, the Verbose Compiler option will display all processed source code lines when compiling (very useful for debugging!)

If compilation was successful, you may directly run your program from iziBasic by pressing the Run it button (otherwise, this button is hidden)

Running the application:

Once your application was created, it appears as all other Palm OS applications in the launcher

You may now run it…



Trick: you now may want to use RsrcEdit (*free publicity!*) to change this ugly icon and to personalize your application with your own designed icon

The title displayed is the one of the program you built

The Hello World label was created

For this screenshot, I clicked on the title. As a consequence, the default About box was launched

This is the end of this quick tour. iziBasic is just that easy to use. ☺

iziBasic source code skeleton

iziBasic requires you to put these very few lines to detect and compile a minimum source code:

```
' YourProgramName.ibas
{CREATORID "XXXX"}

BEGIN
END
```

Notes:
- replace **YourProgramName** by the name you want for your program. This comment line, with a program name ended by a ".ibas" extension) has to be the very first one of your source code for iziBasic to detect that this is an iziBasic source code
- replace **XXXX** by the Palm Creator ID of your program

iziBasic syntax

Legend:

n is Number
v is NumVar (A-Z)
f is NumFunction
c is TextVar (A$-Z$)
t is Text
s is TextFunction

v|n is either NumVar or Number
c|t|s is either TextVar, Text or TextFunction
and so on...

Notes:
- v|n are 32 bit float IEEE 754 compatible numbers
- c|t are strings with up to 63 characters
- char ¶ is CR/LF within c and t
- iziBasic searches for the iziBasic source codes in the Palm inbuilt Memo Pad
- the first line of a program should be like this: ‘ NameOfProgram.ibas
as iziBasic scans for these types of Memos.
- Statements and functions in **GREEN** are only available in the full version of iziBasic.

Variables assignment and calculation

IziBasic does not have a sophisticated math parser yet. So please note the 2 following important notices:
- calculations are made from left to right
So, in iziBasic, 1+2*3 = (1+2)*3 = 9 and not 1+(2*3) = 7
- in statements, v|n or c|t cannot be an aggregate of calculations.
So, in iziBasic, you should work this way (with an example):

```
A=3*COS(B)+5
C=MAX(A,B)
IF A < C PRINT “OK”
D=MIN(A,B) : D=5*MAX(C,D)
```

When with other Basic compilers or interpreters you could do:

```
IF 3*COS(B)+5 < MAX(A,B) PRINT “OK”
D=5*MAX(C,MIN(A,B))
```

Following is a list of the statements available in iziBasic.
As iziBasic is a subset of the BASIC language, with just a few specificities, I have not included a detailed explanation of what the statements do. Please refer to a BASIC documentation if you are not familiar with the BASIC language.

Compiling directives

{CREATORID t}
Set application CreatorID (4 characters)
Notes:
- This directive is mandatory.
- As all Palm OS software, your application should have a unique 4 characters Creator ID. Please refer to the Palm OS website to get all information about this Creator ID "*thing*" and to register yours.

{MINOSVERSION t}
Set the minimum Palm OS version (format "M.m", where M is Major and m minor) for your application to run. It cannot be smaller than "3.0".
If it is set, at runtime your application will check if the target device meets this minimum OS requirement and quit smoothly with a message if it is not met.
Notes:
- This directive is facultative. If not set, iziBasic will assume that it is worth "3.0".
- Warning: I cannot guarantee that all YOUR iziBasic developments will run smoothly from Palm OS 3.0 on. Indeed, iziBasic uses many Palm OS API calls (which are embedded for your convenience) that were introduced progressively over the Palm OS versions. I have identified the very few instructions which require a minimum Palm OS version with informative panels:

The iziBasic runtime checks the device's Palm OS version and these instructions are executed only if the test is positive, otherwise the instructions are not executed.
- So, you should test that your software made with iziBasic will work on all targeted devices, just like with all other development tools.
- My personal tests, running all sample programs shipped with iziBasic, were made on real devices running Palm OS 3.5, 4.1 and 5.2, in the Palm Emulator 3.0, 3.1, 3.3 and 4.0, and in the PalmSimulator 6.0. The good news is that all the sample applications shipped with iziBasic proved to run smoothly in all these cases ☺
- So, if your source code is well built, your application should run from version 3.0 of Palm OS, just like iziBasic itself and my sample programs.
- Please refer to the DESTROY #v|n GUI statement for further information.

{RESOURCEFILE [+] t}
Add a resource file to your program. This is, for instance, very useful to add images (of Tbmp type) which will then be used by the IMAGE and IMAGEBUTTON instructions.
Notes:
- This directive is facultative.
- In your PRC file, the default images shipped with iziBasic are of Tbmp type and are numbered from 1 to 40. Please refer to the IMAGEBUTTON instruction to see the list of these 40 available images.
- The [+] facultative parameter lets you decide if you want to include your resources in addition to the 40 images shipped with iziBasic. If not set, the 40 default images will not be included to your program.
- You may build your resource files with a multi-purpose resource file management tool like RsrcEdit (www.quartus.net/products/rsrcedit/) or with a bitmap dedicated tool like Icon Manager (www.palmgear.com/index.cfm?fuseaction=software.showsoftware&prodid=47054).

{VERSION t}
Set application version.
Note:
- This directive is facultative.

Math Operators + - * / \ ^ MOD NOT AND OR XOR
Note: \ is integer division) and ^ is exponentiation

Test Operators = <> > >= < <=

Labels label:
Notes:
- only the 20 first characters of a label are taken into account by iziBasic
- labels are case sensitive (they are the only ones in this case!), therefore LABEL is not the same thing as Label or label
- you can put up to 70 labels in one source code

Statements SingleStatement [: SingleStatement] [...]
Notes:
- a SingleStatement MUST be < 62 characters long
- Statements are not case sensitive

BASIC Statements

BEEP [v|n] [, v|n]
Notes:
- the first v|n parameter is the number of beeps to play. If not set, beep once
- the second v|n parameter is the type of sound to play (if not set, the Info beep is played):

1 = Info	2 = Warning	3 = Error	4 = StartUp
5 = Alarm	6 = Confirmation	7 = Click	

BEGIN
Statement
END
Note: entry point and last instruction of the program
Both BEGIN and END must be defined
Only the last BEGIN is taken into account by iziBasic if you input several of them. But you can have several END instructions.

BREAK
Note: break program execution (for debugging purposes...)

CALL v|n
Note: not to be used until I provide a documentation of how to build assembler-like routines in the Code Stack. Meanwhile, you can use PEEK and POKE to store [0..255] values, making sure you do not override the used Code Stack (see FRE function).

CHAIN t
Notes:
- the program execution will chain to the “NameOfProgram.ibas” given in t
- the Code Stack will be emptied and refilled with the new program code which will start at its BEGIN point
- Numbers and Texts Stacks are not emptied, so values are passed to the new code. Would you wish to empty them too, just use the CLEAR statement.

CLEAR [v-v | c-c]
Note: clear all A-Z NumVars (set to 0) and A$-Z$ CharVars (set to empty string “”), a range of Numvars only, or a range of CharVars only

CONST c = t

CONST v = n

DEC v
Note: equivalent to v = v - 1

GOTO label

GOSUB label

label:
Statement
RETURN

FOR v = v|n [DOWN]TO v|n [STEP v|n]
Statement
NEXT

IF v|n TestOper v|n Statement
IF c|t TestOper c|t Statement
Note: the Statement can be any statement but a IF THEN [ELSE] ENDIF statement

IF v|n TestOper v|n THEN Statement or **IF c|t TestOper c|t THEN Statement**
[ELSE Statement]
END IF
Note: statements can be put on the next line after THEN AND ELSE. Example:

```
IF A=1 THEN B=2 : C=3
ELSE B=3
ENDIF
```

```
IF A=1 THEN
  B=2 : C=3
ELSE
  B=3
ENDIF
```

INC v
Note: equivalent to v = v + 1

[LET] c = c|t|s [+ c|t|s] [...]
Note: s can also be a A$(v|n), please read the Arrays paragraph

[LET] v = v|n|f [MathOper v|n|f] [...]
Notes:
- calculations are made from left to right
So, in iziBasic, 1+2*3 = (1+2)*3 = 9 and not 1+(2*3) = 7
- f can also be a A(v|n), please read the Arrays paragraph

POKE v|n , v|n
Notes:
- 1st v|n should be in the [FRE(0)..4000] range. Be careful not to POKE under the address returned by the FRE(0).function as it will corrupt the compiled code by iziBasic and your program will behave in a strange manner if you do so!
- 2nd v|n: has to be a [0...255] value.

PUSH c|t|v|n

REM or '
Note: one line comment

REPEAT
** Statement**
UNTIL v|n TestOper v|n or **UNTIL c|t TestOper c|t**

SLEEP v|n

SWAP v , v

SWAP c , c

WHILE v|n TestOper v|n or **WHILE c|t TestOper c|t**
** Statement**
WEND

Functions

NumFunctions:

ABS(v|n)
ACOS(v|n)
ASC(c|t)
ASIN(v|n)
ATAN(v|n)
COS(v|n)
DEGREE(v|n)
Note: v|n has to be in RADIAN
EXP(v|n)
FRE(v|n)
Notes:
- returns the first free address in the Code Stack (v|n=0), in the Number Stack (v|n=1) or in the Text Stack (v|n=2). The free space is the difference between the returned address and the maximum address available (4000 for the Code Stack, 255 for the Number Stack and 200 for the Text Stack).
- FRE(0) returns the first free address not used by your program in the Code Stack. You may then POKE [0..255] values to the Code Stack from this address on up to 4000, that is in the [FRE(0)..4000] range.
INT(v|n)
LEN(c|t)
LOG(v|n)
LN(v|n)
MAX(v|n , v|n)
MIN(v|n , v|n)
NOT(v|n)
PEEK(v|n)
Note: returns the [0..255] value at address v|n which has to be in the range [1..4000]
POP
POWER(v|n , v|n)
RADIAN(v|n)
Note: v|n has to be in DEGREE
RND(v|n)
ROUND(v|n)
SGN(v|n)
SIN(v|n)
SQRT(v|n)
TAN(v|n)
TICKS
TICKSPERSEC
VAL(c|t)

TextFunctions:

BIN$(v|n)	
CHR$(v|n)	
DATE$	Note: date format DD/MM/YYYY
GETOSVER$	
HEX$(v|n)	
LCASE$(c|t)	
LEFT$(c|t , v|n)	
LTRIM$(c|t)	
MID$(c|t , v|n , v|n)	
OCT$(v|n)	
POP$	
RIGHT$(c|t , v|n)	
RTRIM$(c|t)	
SPACE$(v|n)	
STR$(v|n , v|n)	Note for 2nd argument: gives the number of decimals to display If v|n<0 then display number in exponential notation If v|n=0 then display integer part of number
TIME$	Note: time format HH:mm:ss
TRIM$(c|t)	
UCASE$(c|t)	

Constants

PI	Note: is 3.141592654
VERSION	Note: iziBasic version, format is M.m (Major.minor)

Console

CLS

INPUT c|v

PRINT
PRINT c|t [;]
PRINT v|n [USING v|n] [;]
Note for USING v|n: gives the number of decimals to display
If v|n<0 then display number in exponential notation
If v|n=0 then display integer part of number
If USING v|n is not set, then will display the number in exponential notation (equivalent to set it < 0)

WAIT
Note: wait for [Enter] button to be pressed

TextFunctions:

INKEY$

Note: returns an empty string if no key was pressed. You will then track key pressed within a WHILE WEND or a REPEAT UNTIL loop.
Example: REPEAT : K$=INKEY$: UNTIL K$<>””

Graphics

Legend:
x is X coordinate and y is Y coordinate, both are v|n

COLOR v|n
Notes:
- in black & white screen mode, v|n is 0 (white) or 1 (black), in other screen modes, v|n = Red x 65536 + Green x 256 + Blue

GOTOXY x , y

IMAGE v|n, x, y
Note: v|n is the image ID which automatically adapts to the color displaying capabilities of the device: high resolution color; low resolution color, gray scale or black & white. Please refer to the IMAGEBUTTON instruction to see the list of available images shipped with iziBasic and to the RESOURCEFILE compiling directive to see how to add your own customized images to your programs.

LINE [x , y] TO x , y

BOX [x , y] TO x , y

BOXFILLED [x , y] TO x , y

PSET x , y

SCREEN v|n
Notes:
- sets screen mode: 0 for black & white, 1 for 4 grays, 2 for 16 grays, 3 for 256 colors and 4 for 65536 colors
- 

- for devices equipped with Palm OS versions prior to 3.5, screen mode is set to 0 (black & white)
- be careful to set your screen mode before drawing any graphic or GUI object on the screen

NumFunctions:

COLOR(v|n)

Note: returns backcolor if v|n = 0 and frontcolor if v|n = 1

COLORRGB(v|n , v|n , v|n)

Note: returns a color given the Red, Blue and Green gradients (therefore v|n have to be in the [0..255] range

HIGHRES(v|n)

Notes:

- attempts to set the screen size to 320x320 pixels (high resolution) if v|n = 1 and to 160x160 pixels (standard low resolution) if v|n = 0.
- returns 1 if the function succeeded, 0 otherwise; so, if HIGHRES(1) returns 1, the device has a high resolution screen.

SCREENMODE

Note: returns the current screen mode (0 for black & white, 1 for 4 grays, 2 for 16 grays, 3 for 256 colors and 4 for 65536 colors)

SCREENMODES

Note: returns 0 for black & white devices, 1 for 4 grays devices, 2 for 16 grays devices, 3 for 256 colors devices and 4 for 65536 colors devices.

POSX

POSY

GUI

Legend:

x is X coordinate, y is Y coordinate, w is Width and h is Height, all are v|n
#v|n is the Object ID and is in the range [1..999]

ABOUTBOX c|t [+ c|t] [+ c|t]
Note: when the user clicks on the Application name, opens your customized AboutBox, otherwise displays the default AboutBox (which says that your application was made with iziBasic)

BUTTON #v|n , c|t , x , y , w , h

CHECKBOX #v|n , c|t , 0|1 , x , y , w , h
Note: 0|1 = unchecked | checked

DESTROY #v|n
Notes:
- because of a bug in Palm OS prior to version 4.0 (so in versions 3.x), you should destroy an object before using again the same Object ID. To know which version of Palm OS is on the client device, use the GETOSVER$ function and code consequently (i.e. create even empty objects at the program start, then destroy them just before reusing them), or limit your application with the {MINOSVERSION "4.0"} compiling directive which was added almost for this sole purpose!

example for Palm OS prior to version 4.0:

```
LABEL #1,"Hello World",50,50
…
DESTROY #1 : LABEL #1,"Bye, Bye",70,70
```

- for later Palm OS versions (version 4.0 and later), iziBasic handles automatically the reuse of an Object ID by first deleting the previous object and then building the new one, so that you do not have to destroy an object before using again the same Object ID.

example for Palm OS version 4.0 or later:

```
LABEL #1,"Hello World",50,50
…
LABEL #1,"Bye, Bye",70,70
```

- for maximum compatibility, you might want to always proceed like explained for versions prior to version 4.0, as this way of doing will work for all Palm OS versions starting with version 3.0.

- handspring the Handspring Visor series do not support the DESTROY statement because of what I suspect to be a bug in their Palm OS 3.x special versions

GRAFFITISHIFT 0|1 , x , y
Notes:
- 0|1 = unset | set the Graffiti Shift indicator
- in the case of unsetting the Graffiti Shift, the x and y coordinates are fake

IMAGEBUTTON #v|n , v|n , x , y , w , h
Notes:
- v|n is the image ID with the following images available (which automatically adapts to the color displaying capabilities of the device: high resolution color; low resolution color, gray scale or black & white):

- palm 3.5

Warning: there is no check on this image ID number as you can very well add your own customized images to your program. You may also remove or replace this set of images, please refer to the RESOURCEFILE compiling directive for further information.

LABEL #v|n , c|t , x , y

LISTCHOICE #v|n , c|t , c|t , x , y , w , h
Note: the first c|t is the initial selection, the second c|t is the list of selections separated by a ¶ character (example: “choice #1¶choice #2¶choice #3”) with a maximum of 7 items

NUMFIELD #v|n , c|t , 0|1 , x , y , w , h
Notes:
- 0|1 = single line | multiple lines
- only numbers can be keyed in by the user, but the result is returned in a TextVar

PUSHBUTTON #v|n , c|t , 0|1 , x , y , w , h
Note: 0|1 = not pushed | pushed

SETFONT v|n
Note: 0=stdFont 1=boldFont 2=largeFont 3=symbolFont
4=symbol11Font 5=symbol7Font 6=ledFont 7=largeBoldFont

TEXTFIELD #v|n , c|t , 0|1 , x , y , w , h
Note: 0|1 = single line | multiple lines

TEXTSELECTOR #v|n , c|t , x , y, w, h

UPDATEFIELD #v|n , c|t
Notes:
- updates the text of a NUMFIELD or a TEXTFIELD
- this is a way to avoid deleting (DESTROY) and rebuilding a field from scratch (this was the only way in iziBasic version 1.0)

UPDATELABEL #v|n , c|t
Notes:
- updates a label's text created with LABEL
- this is a way to avoid deleting (DESTROY) and rebuilding a LABEL from scratch (this was the only way in iziBasic version 1.0)

UPDATEPOS #v|n , x , y

UPDATETEXT #v|n , c|t
Notes:
- updates the text of the following GUI objects: BUTTON, CHECKBOX, LISTCHOICE, PUSHBUTTON, TEXTSELECTOR
- this is a way to avoid deleting (DESTROY) and rebuilding an object from scratch (this was the only way in iziBasic version 1.0)

NumFunctions:

CHECKBOX(#v|n)

Note: returns 0=unchecked or 1=checked

COLORSELECT(v|n)

Notes:

- v|n is the initial proposed color
- 

DOEVENTS

Note: returns -1=ExitAppRequest or 0=no event or #v|n that was clicked (Button, CheckBox, ListChoice, PushButton or TextSelector)

FONTSELECT(v|n)

Notes:

- the v|n parameter is the default highlighted font in the dialog box and it has to be one of these three values: 0=stdFont, 1=boldFont, 7=largeBoldFont
- returns the selected font

MESSAGEBOX(c|t [+ c|t] [+ c|t] , v|n)

Notes:

- v|n is the type of MessageBox

0= Done	1= OK	2= OK|Cancel	3= Yes|No
4= 1|2	5= 1|2|3	6= 1|2|3|4	7= 1|2|3|4|5
8= 1|2|3|4|5|6		9= 1|2|3|4|5|6|7	

- returns the button pressed (1=first , 2=second …)

PENDOWN

Note: returns 1 when pen is down and 0 when pen is up

PENX

PENY

Note: PENX and PENY are trapped during a DOLOOP or a WAITEVENT call even though they do not return an event themselves

PUSHBUTTON(#v|n)

Note: returns 0=not pushed or 1=pushed

SELECTEDCHOICE

Note: returns the selected item (1=first , 2=second …) of the last LISTCHOICE event => you should capture the ListChoice event with a DOEVENTS and get the selected item just afterwards

WAITEVENT

Notes:

- returns -1=ExitAppRequest or #v|n that was clicked (Button, CheckBox, ListChoice, PushButton or TextSelector)
- be aware that control is not returned to your program until one event occurs, use DOEVENTS if you need to get control back between events. In other words, [A=WAITEVENT] is equivalent to: [REPEAT : A=DOEVENTS : UNTIL A<>0]

TextFunctions:

DATESELECT$(c|t)

Notes:
- c|t is the default date sent to the Date Selector, format is DD/MM/YYYY. Pass an empty string if you wish to send the current date.
- date format returned is DD/MM/YYYY or empty if the user cancelled the input

FIELD$(#v|n)

Notes:
- retrieves the value stored in a NumField or a TextField
- to convert the returned value in a NumField to a number, use the VAL function

TIMESELECT$(c|t)

Notes:
- c|t is the default time sent to the Time Selector, format is HH:mm. Pass an empty string if you wish to send the current date.
- time format returned is HH:mm or empty if the user cancelled the input
- be careful with this time format as it differs from the HH:mm:ss format used in the TIME$ function
- 

Preferences

Legend:
- #v|n is the Pref Number and is v|n in the range [1..999]

DELETEPREF #v|n

SAVEPREF #v|n , v|n|c|t

NumFunctions:

LOADPREF(#v|n)

TextFunctions:

LOADPREF$(#v|n)

Arrays

There are 2 arrays defined in iziBasic: A() and A$().
At runtime, both can address all Numbers and Strings stacks.

A-Z variables are also addressed with A(1)-A(26)
A$-Z$ variables are also addressed with A$(1)-A$(26)
This means, for instance, that A$(2) and B$ are the same thing, A(4) and D are also the same.

DIM A(n)
Notes:
- At design time DIM A(n) reserves some space in the Numbers stack in addition to the A-Z variables
- n>26 and n<=250. You will have to leave some space in the upper stack for all other numerical assignments
- DIM A(n) must be defined before the iziBasic compiler reserves some space in the Numbers stack for its use, so you should place your DIM A(n) at the top of your program, just after the Compiling Directives

DIM A$(n)
Notes:
- At design timeDIM A$(n) reserves some space in the Text stack in addition to the A$-Z$ variables
- n>26 and n<=198. You will have to leave some space in the upper stack for all other text assignments
- DIM A$(n) must be defined before the iziBasic compiler reserves some space in the Text stack for its use, so you should place your DIM A$(n) at the top of your program, just after the Compiling Directives

CONST A$(n) = t

CONST A(n) = n

[LET] A$(v|n) = c|t|s [+ c|t|s] [...]
Note: s can also be a A$(v|n)

[LET] A(v|n) = v|n|f [MathOper v|n|f] [...]
Note: f can also be a A(v|n)

Files

Note:

- iziBasic is limited to work on database files (dtb), it does not work on program files (prc). It is even limited to work on specific database files (“DATA”, “Data” and “data” types) in most instructions. This was made on purpose, to avoid any risk of performing dangerous actions on files or deleting programs on the devices.

Legend:

- #v|n is the File Handle and is in the range [0..9]; this means that you can work with up to 10 files simultaneously
- c|t is the name of the database file; do not put the “.dtb” extension in this name

CLOSE #v|n

COPY c|t , c|t , [c|t]
Note for [c|t]: facultatively set Creator ID of destination file to a new 4 characters value
Warning: you are allowed to copy databases with any Creator ID, this means that you can copy almost any database file (with “DATA”, “Data” or “data” type) on your device.

INPUT #v|n , v|c

KILL c|t
Warning: you are allowed to delete databases with any Creator ID, this means that you can delete almost any database file (with “DATA”, “Data” or “data” type) on your device.

OPEN c|t FOR INPUT|OUTPUT|APPEND|RANDOM AS #v|n
Notes:
- to avoid any risk of deleting important files on your Palm device, the OUTPUT, APPEND and RANDOM file modes only work with databases having a “LDIB” Creator ID
- be careful: if you erase iziBasic from your device, which has “LDIB” as a Creator ID, the standard Palm OS deletion mode will also erase all databases having a “LDIB” Creator ID
- would you wish to work on external files anyway, you may do it in 3 steps by using the COPY instruction with “LDIB” as Creator ID for the destination file before using the OPEN instruction with this destination file. After you finish working on this file, CLOSE it, then you may KILL the original file and COPY the destination back to the original one.

PRINT #v|n , v|n|c|t

RENAME c|t , c|t
Warning: you are allowed to rename databases with any Creator ID, this means that you can rename almost any database file (with “DATA”, “Data” or “data” type) on your device.

RUN c|t
Notes:
- exits from the current program and launches the new c|t program
- if the program passed in parameter does not exist, remains in the current program and you might then want to handle the error in the lines following the RUN instruction…

SEEK #v|n , v|n
Note: SEEK is available for files opened in INPUT or RANDOM mode.

NumFunctions:

EOF(#v|n)
Note: End Of File reached (1=true / 0=false)

FILEERROR
Note: last file operation error (1=true / 0=false)

FILEEXISTS(c|t)
Note: returns 1 = file exists or 0 = file does not exist

LOC(#v|n)
Note: LOcation in File = current record number

LOF(#v|n)
Note: Length Of File = number of records

PP Code Segment Call

Palm Pascal onboard Compiler (nicknamed “PP”) is a great, free and very fast compiler available on the Palm platform. iziBasic itself is made with this compiler which can be found here: http://ppcompiler.free.fr/.
PP has this great feature of being able to handle multiple DragonBall code segments. iziBasic, by construction, cannot handle direct Palm API calls.
So, if for any reason, you need to build very quick routines or to access directly the Palm APIs, you might want to include some PP segments in your iziBasic projects. These code segments are easily added in a resource file (see RESOURCEFILE compiling directive) or on top of an iziBasic compiled program during a PP compilation.

Advice: give a look to the iBHelloPP sample program which shows a very simple implementation of a PP code segment call from an iziBasic program

TextFunctions:

CALLPP$(v|n , [c|t])

Notes:
- v|n is the code segment handle and has to be in the [100..999] range (the [0..99] range is reserved for iziBasic)
- [c|t]: pass a facultative text parameter to the PP segment
- returns a file operation error (1=true / 0=false) which can be read by the FILEERROR function. This can be used to know if the call succeeded.

The PP source code skeleton for a segment called by the CALLPP$ function has to be the following:

```
{$code appl,XXXX,code,nnn}
program YourProgramName;
type iBasFunType=function(S:string):string;
var iBasCallPP:iBasFunType;

type
// Insert here any type that your source code requires

const
// Insert here any const that your source code requires

// WARNING: DO NOT INSERT ANY VAR HERE. Only use local
// variables (in functions and procedures)

// Insert here any function or procedure that your
// source code requires

// You may rename the CallPP function if you like
function CallPP(S:string):string;
begin
 // Insert here some Pascal source code
end;

begin
// Also rename the CallPP function in next line if you
// did it above
 iBasCallPP:=CallPP;
end.
```

Notes:

- everything that is written in **black** is mandatory, in **blue** are comments and facultative definitions, in **green** are parameters to adapt
- **XXXX** is the CreatorID set with the CREATORID compiling directive in your iziBasic source code
- **nnn** is the code segment handle, as called by the CALLPP$ function, please remember that it has to be in the [100..999] range
- replace **YourProgramName** by the program name you defined in your iziBasic source code

Compiler errors:

The iziBasic compiler is a one pass compiler which processes several checks but some controls are not made. For instance, any:

- IF THEN [ELSE] ENDIF without ENDIF is not checked
- WHILE WEND without WEND is not checked
- FOR I=1 TO 10 : FOR I=1 TO 20 : NEXT : NEXT is not checked (use of the same Number variable in 2 overlapped loops)
- …

Of course, at runtime you will then get unexpected results!

CHAIN Stack Overflow

You are trying to chain more than 10 Memo files.
Advice: group small source codes together rather than splitting them.

Code Stack Overflow

Your code size has over passed 4000 bytes.
Advice: split your source code in several Memo files and link them with the CHAIN instruction.
Note: the Code Stack is of 4000 records but the iziBasic compiler uses 6 bytes of these records as internal registers.

DIM defined too late

The compiler has already started to reserve some Numbers or Text values.
Advice: put your DIM statement at the top of your source code, right after the Compiling Directives.

Jump Stack Overflow

Your Jump Stack has over passed its size: 70 records.
Advice: decrease the number of labels and of GOSUB / GOTO instructions.

Number Stack Overflow

Your Number Stack size has over passed 255 records.
Advice: use the CONST instruction for values that are fixed at the beginning of the program to decrease the Stack.
Notes:
- the Number Stack is of 255 records but the iziBasic compiler uses 5 of these records as internal registers and also a variable number of records according to your source code requirements
- the 26 spaces required by the A-Z variables are also reserved by iziBasic in the Number Stack

Single Statement too long

One Statement was over 62 characters. You will have to split it into 2 lines or 2 statements (separated by a ":" character).

Syntax error

There is a syntax error in the last statement processed by the compiler.
Advice: make sure that none of your labels starts with a reserved keyword. For example: CloseMyForm: or GoToMySubRoutine: are not good when MyFormClose and JumpToMySubRoutine are valid.

Text Stack Overflow

Your Text Stack size has over passed 200 records.
Advice: use the CONST instruction for values that are fixed at the beginning of the program to decrease the Stack.
Notes:
- the Text Stack is of 200 records but the iziBasic compiler uses 2 of these records as internal registers.
- the 26 spaces required by the A$-Z$ variables are also reserved by iziBasic in the Text Stack

Sample iziBasic source codes

In the iziBasic ZIP distribution file, you will find a directory called Examples.

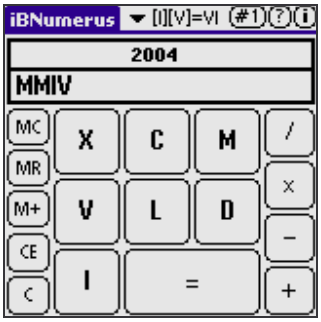
In this Examples directory are offered full source codes of 12 sample applications. Reading, running and working on these source codes will help you to quickly get a full understanding of the iziBasic capabilities as I believe that learning by the example is the best way to learn programming.

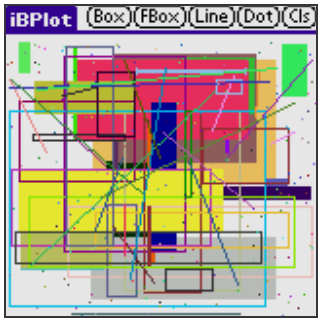
Therefore, I highly invite you to give a deep reading of these sample source codes and programs.

Note: some programs require the full iziBasic to be compiled; the trial version will not compile them as is. But, I provide a compiled version of all sample programs.

- Source codes which can be compiled with trial version:
  Bench1, Bench2, iBcHello, iBcMatches, iBcNumerus, iBHello
- Source codes which can only be compiled with full version:
  iBAddress, iBClock, iBHelloPP, iBMatches, iBNumerus, iBPlot

To give you an idea of the power of iziBasic, here are snapshots for 5 of the 12 sample applications:





Appendix: PIAF, QED, SiEd and SrcEdit DOC editors

The most sophisticated Palm hosted compilers (PP and OnBoardC for instance) rely on a DOC editor to write the source codes. iziBasic can read both the Memo Pad format or the DOC format.
Using a DOC editor is very convenient in most cases because DOC editors provide great editing features and they overpass the 4096 characters limit of the Memo Pad.

There are two types of DOC formats: a compressed one and an uncompressed one (which was the original one).
ⓘ iziBasic does not recognize the DOC compressed format, only the uncompressed format.

Below, you will find a non exhaustive reference list of five Palm hosted DOC editors which are flavoured by the developers' community and that can be directly launched by iziBasic (other DOC editors could be launched by iziBasic too, do not hesitate to ask).

PIAF

PIAF is the dedicated DOC editor for the Palm Pascal (PP) onboard compiler.
URL: http://ppcompiler.free.fr/ Cost: free, copyleft source code

QED

QED is a well established DOC editor, well known by the developers as it was one of the first (if not the very first one) Palm hosted DOC editors.
URL: http://qland.de/qed/ Cost: shareware ($12.95)

SiEd

SiEd is the newest of the DOC editors in this list. It has a very convenient functionality: it can save a file in the DOC format in the device's main memory and in a text format on an external memory card.
URL: http://www.benroe.com/sied/ Cost: free, GPL source code

SrcEdit

SrcEdit is the dedicated DOC editor for the OnboardC compiler.
URL: http:// onboardc.sourceforge.net/ Cost: free, GPL source code