POB-BASIC Documentation

Table of contents

Document management

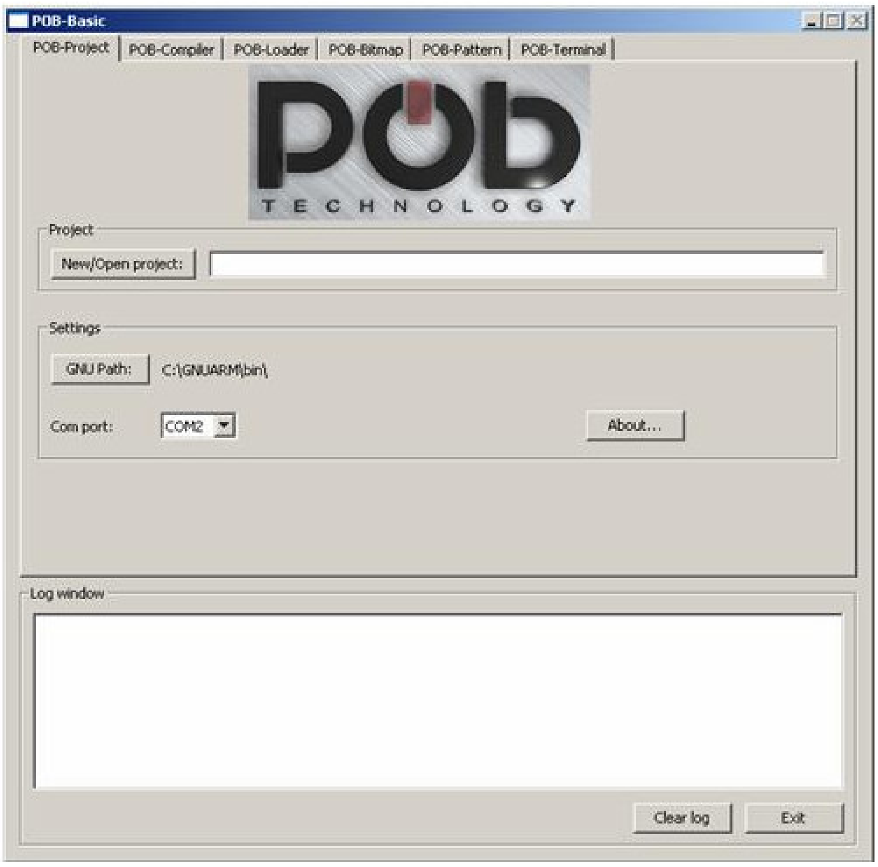
Filename	Manuel POB-Basic US.doc
Creation date	17/03/2006
Author	Baptiste BURLES

POB-Technology contacts

Address	POB-TECHNOLOGY 4, rue nicéphore niépce 69 007 LYON, FRANCE
Mail	contact@pob-technology.com
Phone	+33 (0)4 72 43 02 36
Fax	+33 (0)4 78 58 04 92

1 Introduction

POB-Basic is a set of tools for programming, loading and controlling the POBEYE module in Basic language. This tool is made of 6 modules:



GNU Path: C:\GNUARM\bin\

Com port: COM2

POB-TOOLS: Allows managing an application project.

POB-Compiler: Enables one-click source file compilation.

POB-Loader: Uploads an application to the POBEYE module.

POB-Bitmap: Produces picture library for the POB-LCD128.

POB-Pattern: Creates a pattern dictionary.

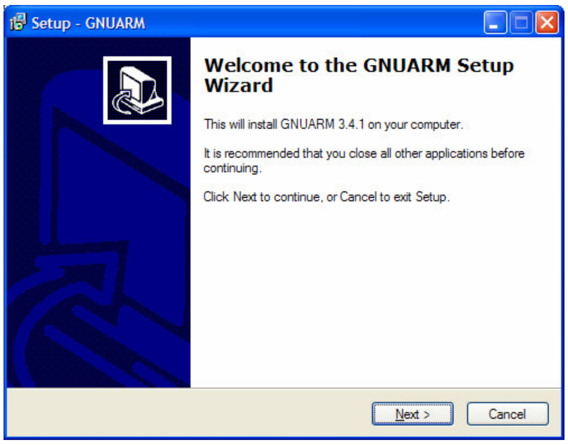
POB-Terminal: Helps debugging your application with the serial port.

2 POB-Basic installation

On the CD supplied with POBEYE module, you will find these two files.

▪ *Installation of the GNUARM compiler*

The file « **bu-2.15_gcc-3.4.1-c-c++-java_nl-1.12.0_gi-6.0.exe** » is the GNUARM compiler. POB-Basic uses this compiler to create your application.
To install the GNUARM compiler, simply double-click on the file.

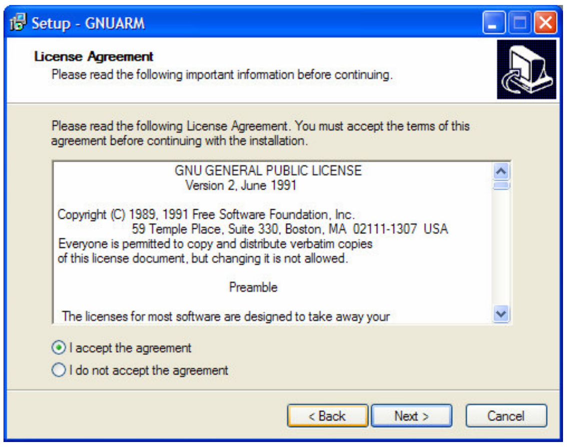

Setup - GNUARM

Welcome to the GNUARM Setup Wizard

This will install GNUARM 3.4.1 on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.


Setup - GNUARM

License Agreement
Please read the following important information before continuing.

Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

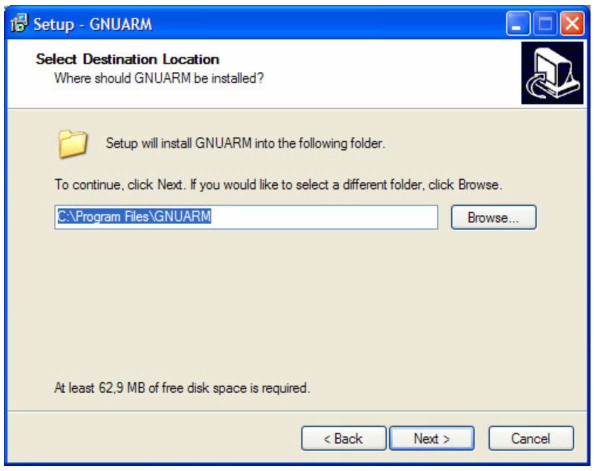
Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your

[✓ I accept the agreement]
I do not accept the agreement

GNUARM introduction and software license: GNU GPL


Setup - GNUARM

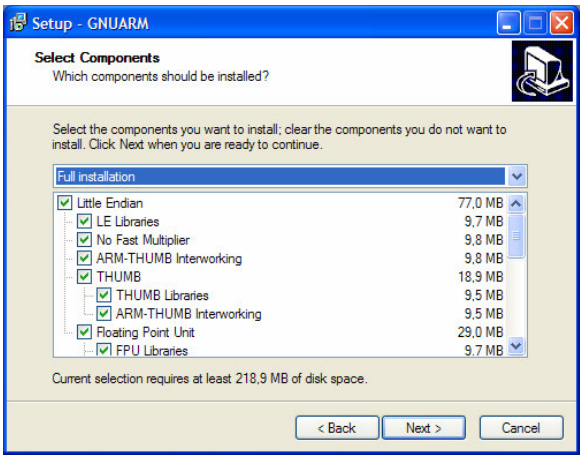
Select Destination Location
Where should GNUARM be installed?

Setup will install GNUARM into the following folder.

To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program Files\GNUARM

At least 62,9 MB of free disk space is required.


Setup - GNUARM

Select Components
Which components should be installed?

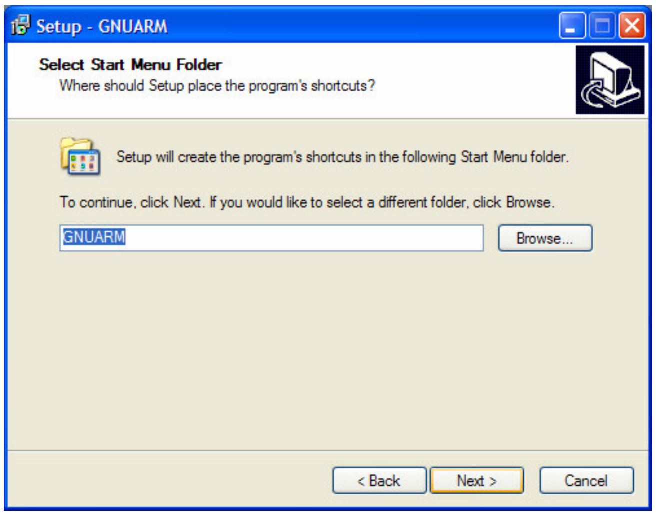
Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

Full installation

✓ Little Endian	77,0 MB
✓ LE Libraries	9,7 MB
✓ No Fast Multiplier	9,8 MB
✓ ARM-THUMB Interworking	9,8 MB
✓ THUMB	18,9 MB
✓ THUMB Libraries	9,5 MB
✓ ARM-THUMB Interworking	9,5 MB
✓ Floating Point Unit	29,0 MB
✓ FPU Libraries	9,7 MB

Current selection requires at least 218,9 MB of disk space.

Install repertory and software options (leave everything ticked)

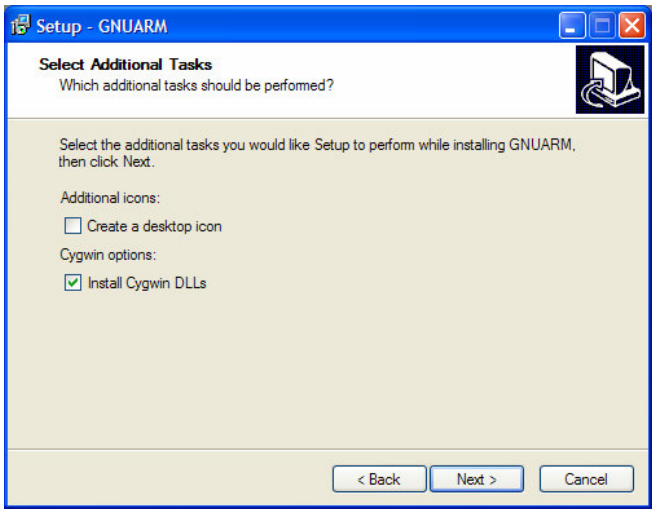

Setup - GNUARM

Select Start Menu Folder
Where should Setup place the program's shortcuts?

Setup will create the program's shortcuts in the following Start Menu folder.

To continue, click Next. If you would like to select a different folder, click Browse.

GNUARM

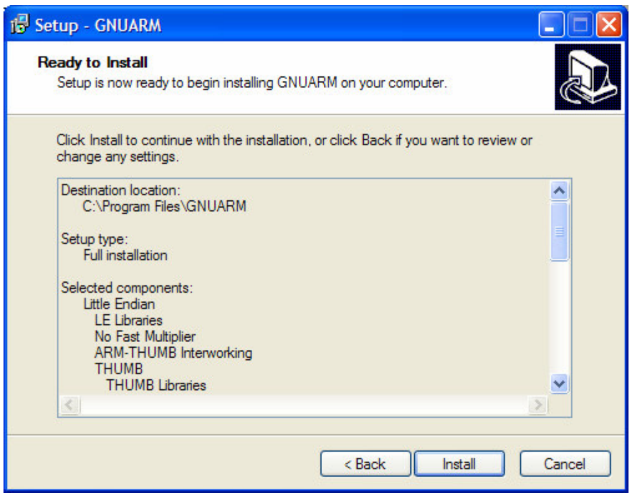

Setup - GNUARM

Select Additional Tasks
Which additional tasks should be performed?

Select the additional tasks you would like Setup to perform while installing GNUARM, then click Next.

Additional icons:
☐ Create a desktop icon
Cygwin options:
☑ Install Cygwin DLLs

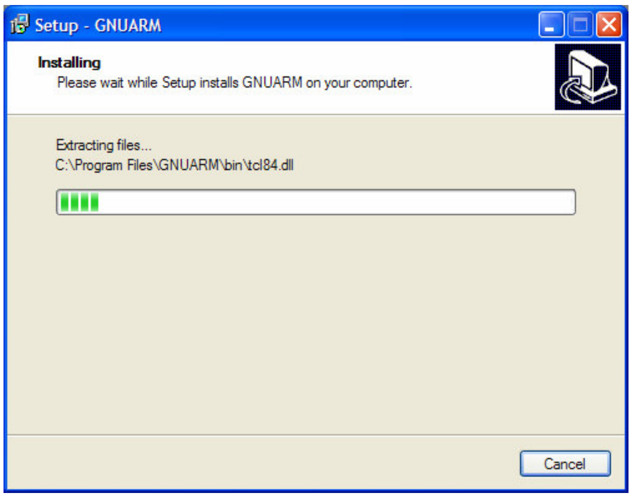
Start Menu and options: Deselect 'Create a desktop icon' and leave 'Install Cygwin DLLs'


Setup - GNUARM

Ready to Install
Setup is now ready to begin installing GNUARM on your computer.

Click Install to continue with the installation, or click Back if you want to review or change any settings.

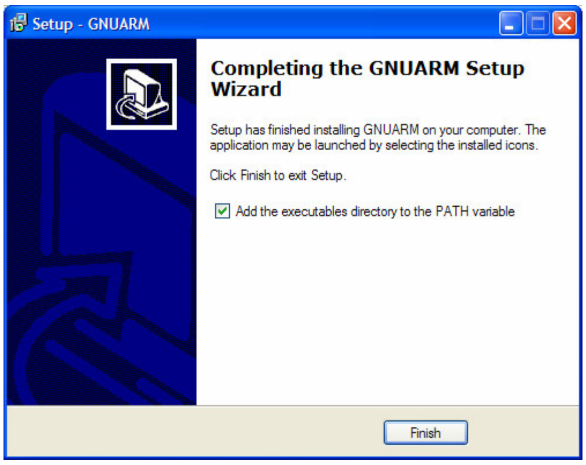
```
Destination location:
    C:\Program Files\GNUARM

Setup type:
    Full installation

Selected components:
    Little Endian
      LE Libraries
      No Fast Multiplier
      ARM-THUMB Interworking
      THUMB
        THUMB Libraries
```


Setup - GNUARM

Installing
Please wait while Setup installs GNUARM on your computer.

Extracting files...
C:\Program Files\GNUARM\bin\tcl84.dll

Summary of the components that will be installed and install begin


Setup - GNUARM

Completing the GNUARM Setup Wizard

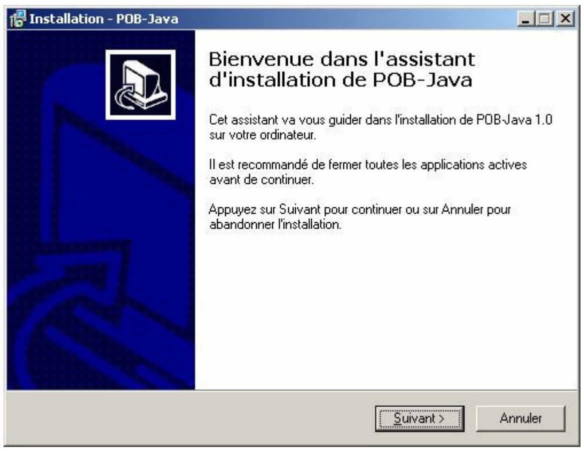
Setup has finished installing GNUARM on your computer. The application may be launched by selecting the installed icons.

Click Finish to exit Setup.

☑ Add the executables directory to the PATH variable

▪ *Installation POB-Basic*

Click on « **pobbasic_setup.exe** » to install POB-Basic. Follow the instructions for setup.

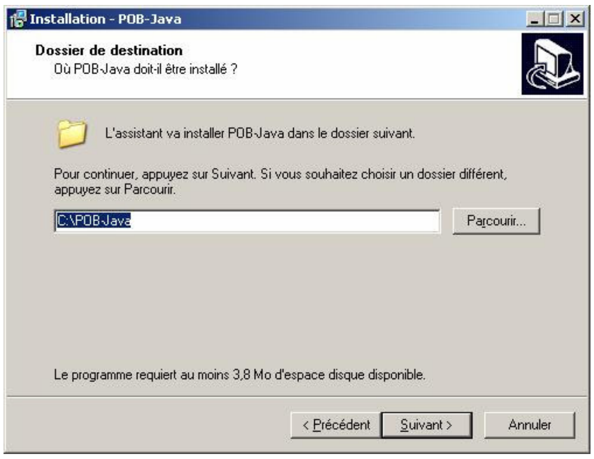

Installation - POB-Java

Bienvenue dans l'assistant d'installation de POB-Java

Cet assistant va vous guider dans l'installation de POB-Java 1.0 sur votre ordinateur.

Il est recommandé de fermer toutes les applications actives avant de continuer.

Appuyez sur Suivant pour continuer ou sur Annuler pour abandonner l'installation.


Installation - POB-Java

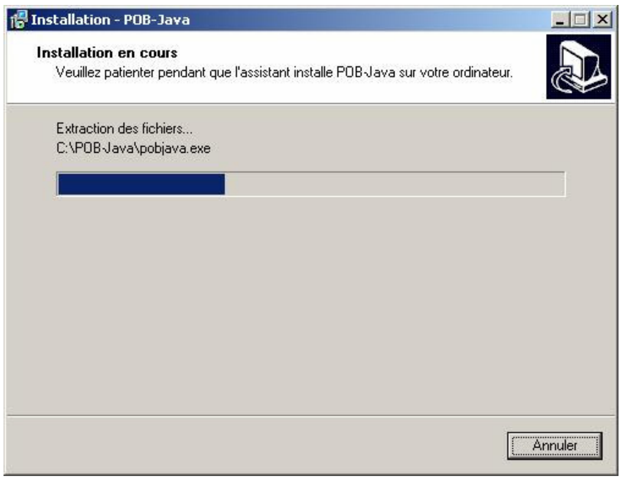
Dossier de destination
Où POB-Java doit-il être installé ?

L'assistant va installer POB-Java dans le dossier suivant.

Pour continuer, appuyez sur Suivant. Si vous souhaitez choisir un dossier différent, appuyez sur Parcourir.

C:\POB-Java

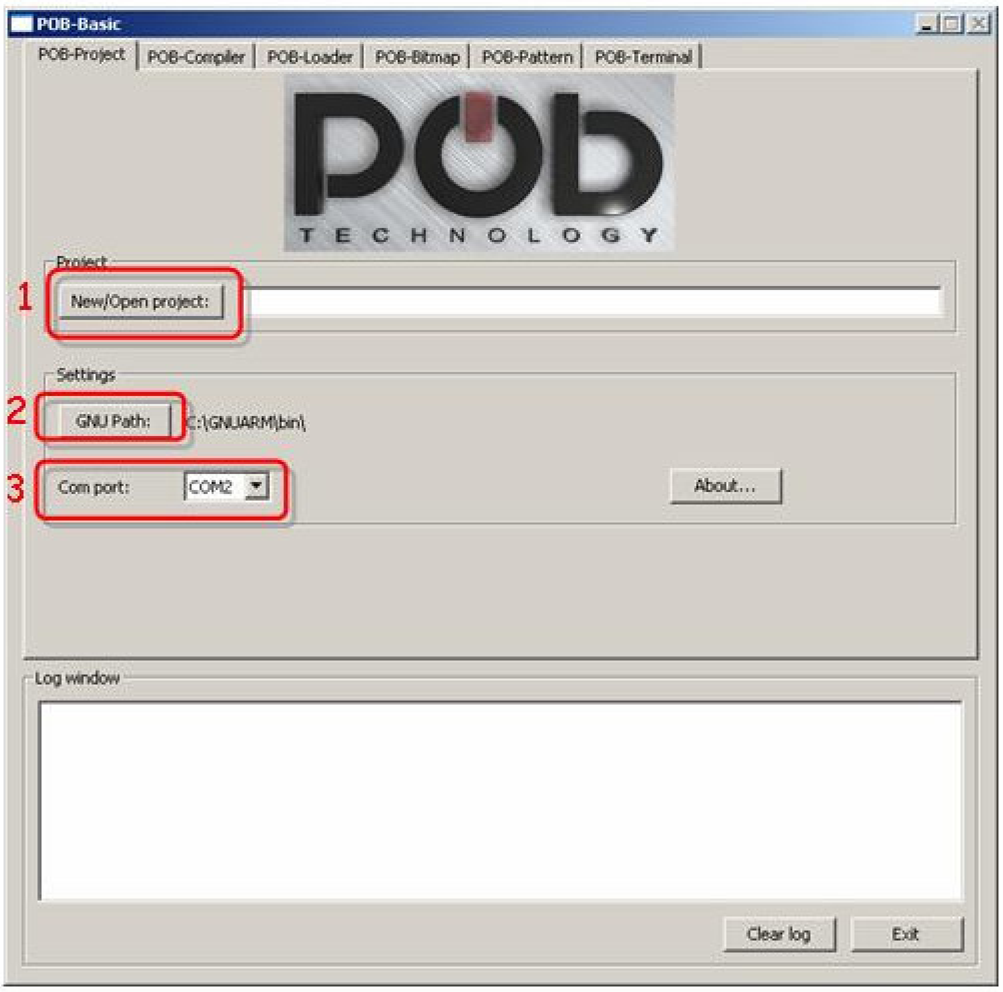
Le programme requiert au moins 3,8 Mo d'espace disque disponible.


Installation - POB-Java

Installation en cours
Veuillez patienter pendant que l'assistant installe POB-Java sur votre ordinateur.

Extraction des fichiers...
C:\POB-Java\pobjava.exe

3 POB-Basic Configuration



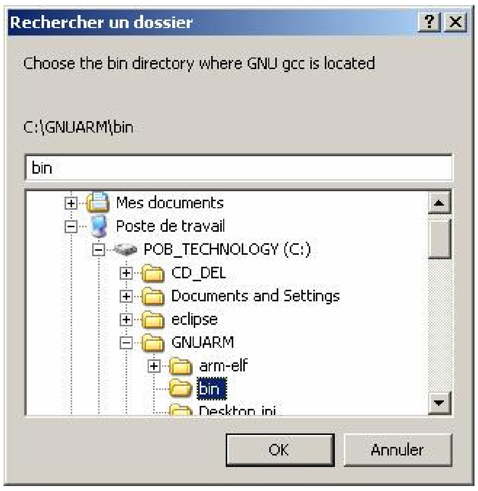
▪ *1 Manage a project application*

POB-Basic is used to manage a project application for the POB-EYE module.

The "*new/open project*" button allows the creation of a new project or opens an existing project. You must type the name of the new project or select an existing project (extension .pobbasic).

2 Path to use the GNUARM compiler

You must click on the 'bin' directory of GNUARM.
If you have installed it using the default settings, the default path is:
C:\Program Files\GNUARM\bin


Rechercher un dossier

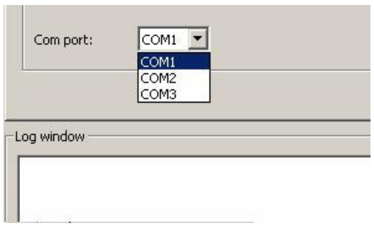
Choose the bin directory where GNU gcc is located

C:\GNUARM\bin

bin

3 Select the serial port

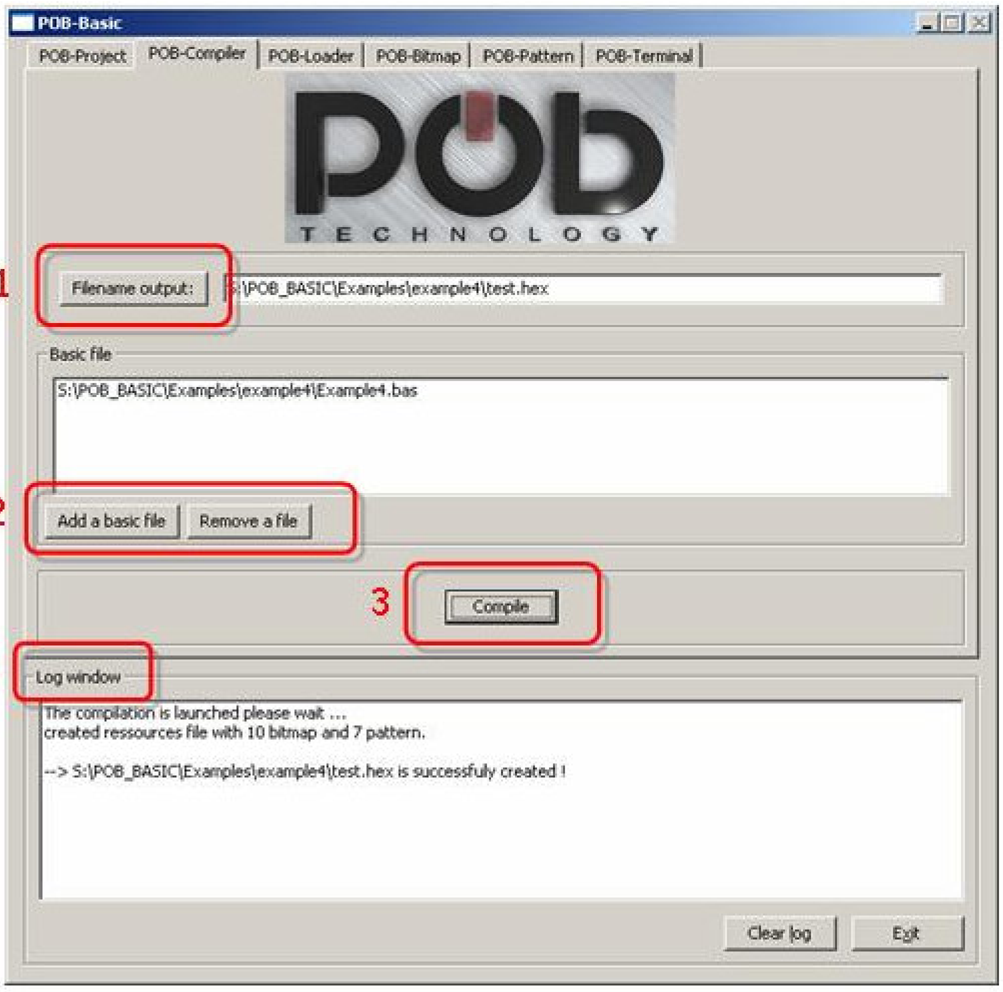
Choose the port number on which POBEYE is connected to your computer.


Com port: COM1

4 POB-Basic page

4.1 POB-Compiler

POB-EYE module is programmed in the Basic language. The POB-Compiler will create an application written in Basic for the POB-EYE module. To create an application, follow the 4 following steps:



Filename output: S:\POB_BASIC\Examples\example4\test.hex

Basic file: S:\POB_BASIC\Examples\example4\Example4.bas

```
The compilation is launched please wait ...
created ressources file with 10 bitmap and 7 pattern.

--> S:\POB_BASIC\Examples\example4\test.hex is successfuly created !
```

1 – Naming your application:

Click on « *Filename output* »: a dialog box will be displayed to set the name of your application (.hex extension).

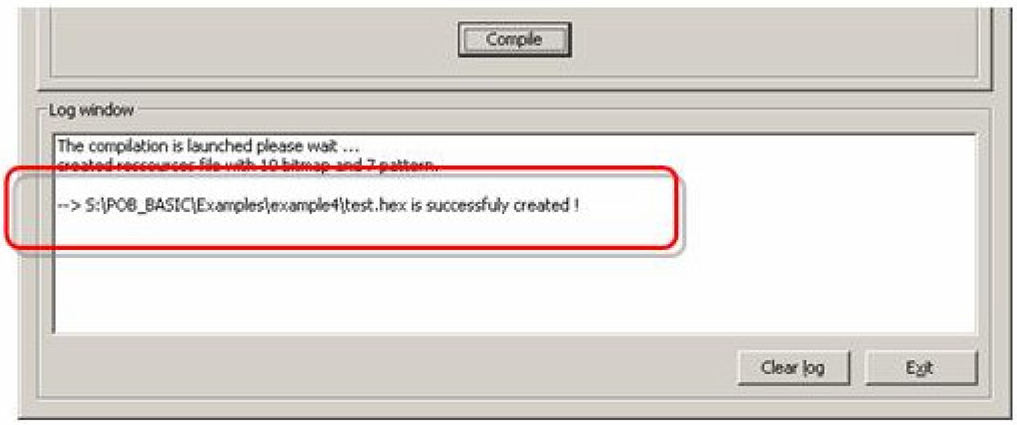
If the file already exists, you only have to select it.

2 - Adding files to be compiled:

You have to choose the list of source files that will be compiled. To add a file, click on the « *Add a basic file* » button. A dialog box appears and allows you to choose the file that you want to add.
If you want to delete a file from the list, use the « *Remove a file*» button.

3 - Compiling:

Click on the « ***Compile*** » button to launch the compilation of your application. During the compilation, you will see traces being displayed in the window called « **Log window**». Some "warning" or "error" messages may appear.



```
The compilation is launched please wait ...
created rescources file with 10 bitmap and 7 pattern.

--> S:\POB_BASIC\Examples\example4\test.hex is successfuly created !
```

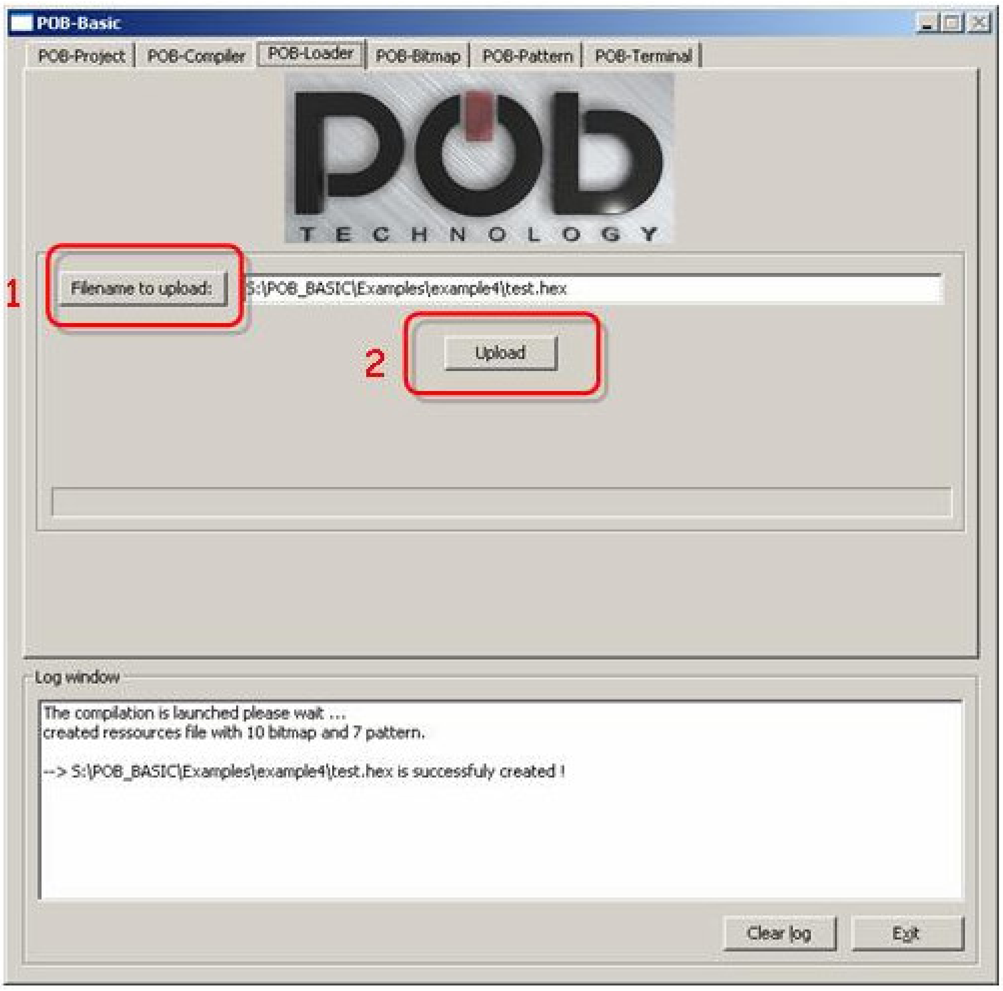
The last line that you will see appearing in the console will tell you if the application was successfully compiled. If the message « **... is successfully created!!!** » appears, your application is ready to be used in the POBEYE module.
If the compilation process failed, the message « **...is not created** » appears. You will then need to correct your application in relation to the message from the compiler.

Remarks:
The compilation includes the graphics and pattern resources. Before compiling, graphics or pattern resources must be created in POB-Bitmap/POB-Pattern if your application needs these resources.

4.2 POB-Loader

POB-Loader module is used to load a program (create in the previous module, Pob-compiler) in the POBEYE memory.

Two steps are necessary to load a program:



```
The compilation is launched please wait ...
created ressources file with 10 bitmap and 7 pattern.

--> S:\POB_BASIC\Examples\example4\test.hex is successfuly created !
```

1 - Selecting an application:

Click on « *Filename to upload* » to choose the application.

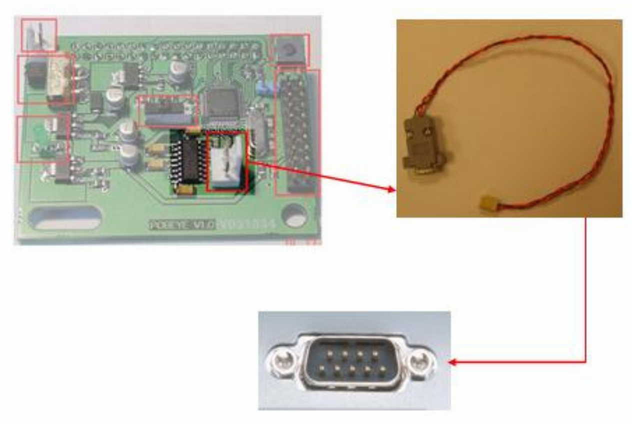
2 – Program uploads:

Remark: Before loading a program, POBEYE module must be on, in “programming” mode and link to your computer by the serial cable.

Preparing POBEYE module:

- POBEYE module must be link to your computer: see chapter 1.9 of pob-technology documentation.

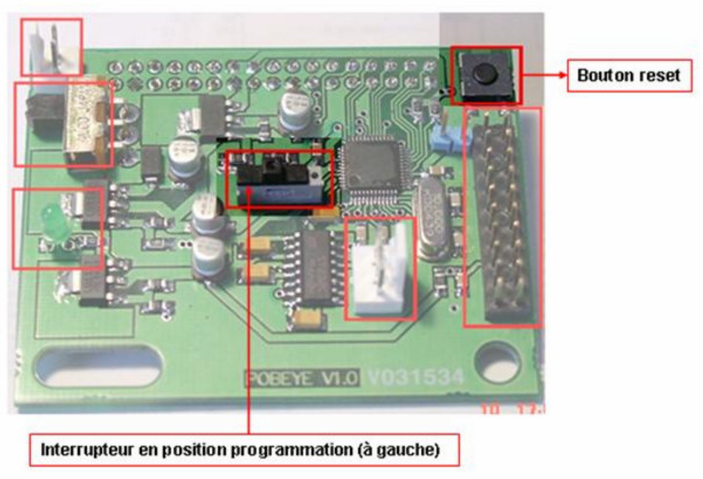
- POBEYE module must be in “programming” mode: see chapter 1.3 of pob-technology documentation.



To load a program, click on the “**Upload**” button. If the loading proceeds correctly, you should see the progress bar changing.

If this does not work, an error message will appear. You will have to follow the instructions to solve the problem. If the problem persists, do not hesitate to contact the POB-Technology support: support@pob-technology.com

Remark: If the modules are already on, simply switch to programming mode the switch “programming/execution” and press the reset button.

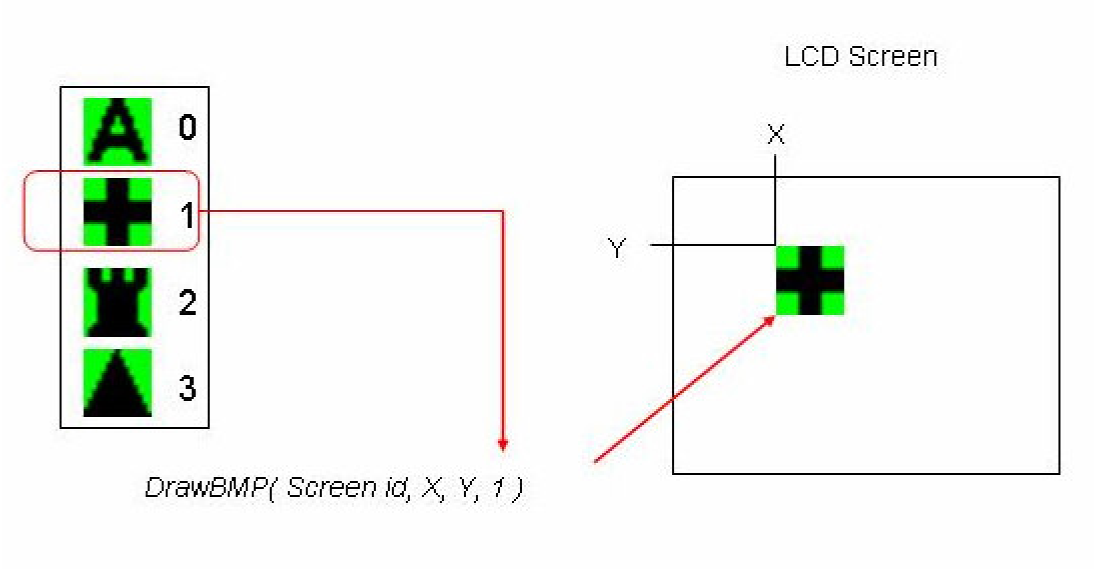


4.3 POB-Bitmap

POB-Bitmap generates graphic resources for the POB-LCD128. POB-LCD allows, for example, real time visualization of what the camera sees or to act as a user interface.

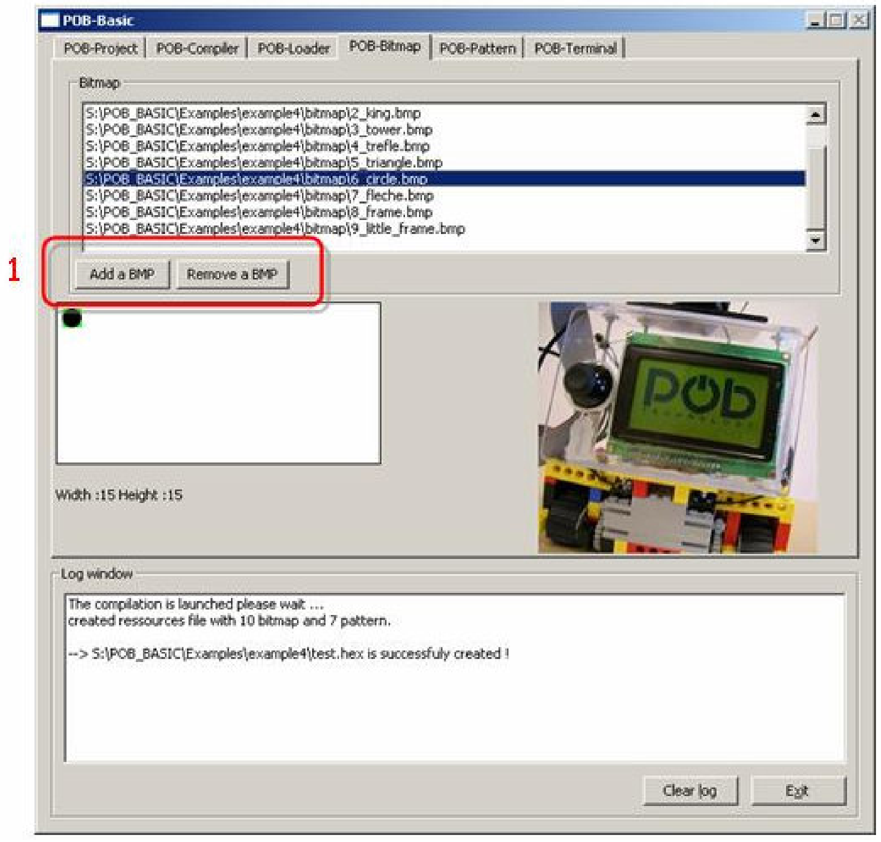
The graphics resources can be displayed using the library supplied with the basic library. The graphic functions can manage images transparency to carry out the superposition of images on the LCD screen.

Images are displayed using the « *DrawBMP* » function:



This function needs screen id, x y coordinates and bitmap number.

▪ *Graphic resources generation:*



```
S:\POB_BASIC\Examples\example4\bitmap\2_king.bmp
S:\POB_BASIC\Examples\example4\bitmap\3_tower.bmp
S:\POB_BASIC\Examples\example4\bitmap\4_trefle.bmp
S:\POB_BASIC\Examples\example4\bitmap\5_triangle.bmp
S:\POB_BASIC\Examples\example4\bitmap\6_circle.bmp
S:\POB_BASIC\Examples\example4\bitmap\7_fleche.bmp
S:\POB_BASIC\Examples\example4\bitmap\8_frame.bmp
S:\POB_BASIC\Examples\example4\bitmap\9_little_frame.bmp
```

Width :15 Height :15

```
The compilation is launched please wait ...
created ressources file with 10 bitmap and 7 pattern.

--> S:\POB_BASIC\Examples\example4\test.hex is successfuly created !
```

1 – Manage images:

To add or remove an image, press the “*Add Bmp*” or “*Remove Bmp*” button.

Remark: The images put in the library must respect the following format: Bitmap 256 colors, maximum size of 256 per 256 pixels.

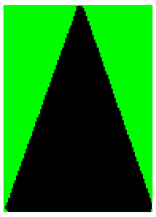
POB-Bitmap uses 3 colors to draw the graphic resources:

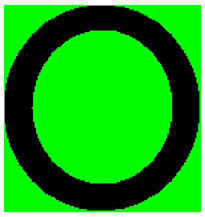
- Black: black pixel is drawing on the LCD.
- White: white pixel is drawing on the LCD.
- Green (Red 0, green 255, Blue 0): transparency color (allow stack images).

▪ *About transparency:*

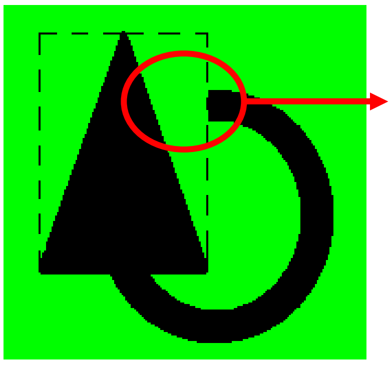
Definition of transparency color:
This color allows a pixel the possibility of not taking shape. Thus, by this means, one keeps what is already drawn.

For example, take's 2 images:



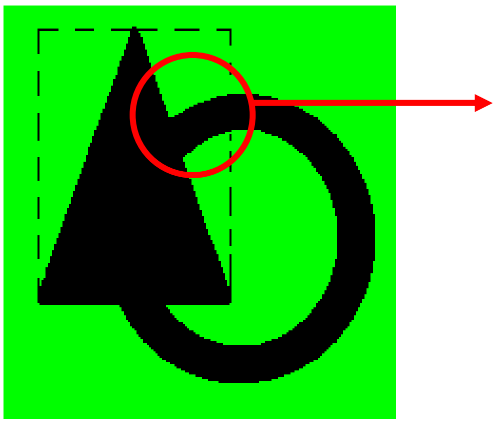


If you draw the triangle on the circle without the management of the color transparency, here is what happens:



The triangle frame erases the circle.

With the transparency management, the superposition of images is possible. Here is the result:

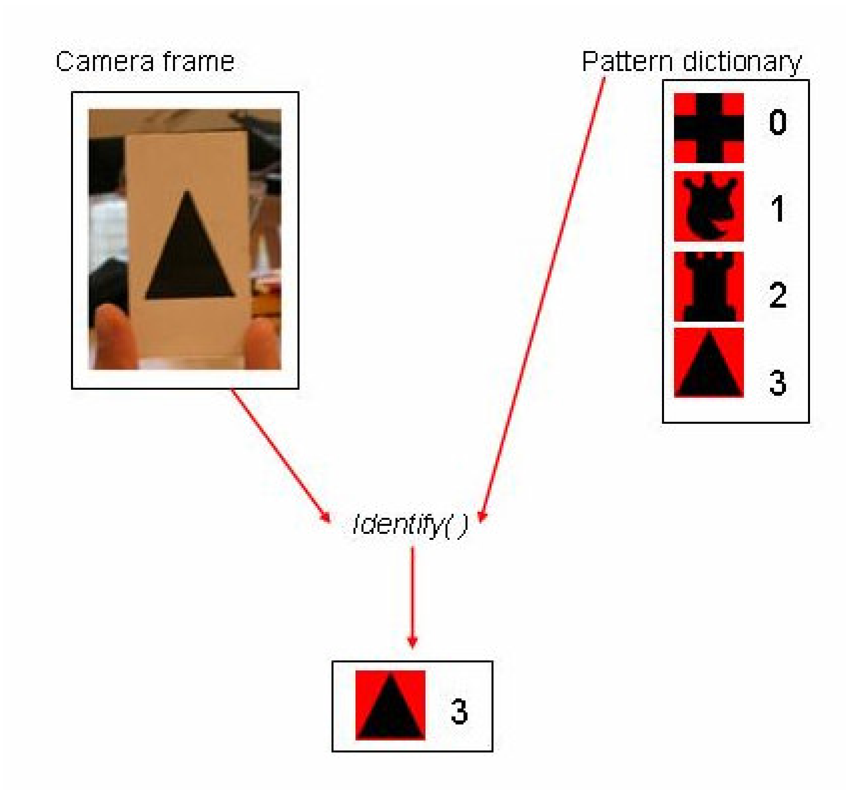


Transparency management allows the part of the circle under the triangle to be shown.

4.4 POB-Pattern

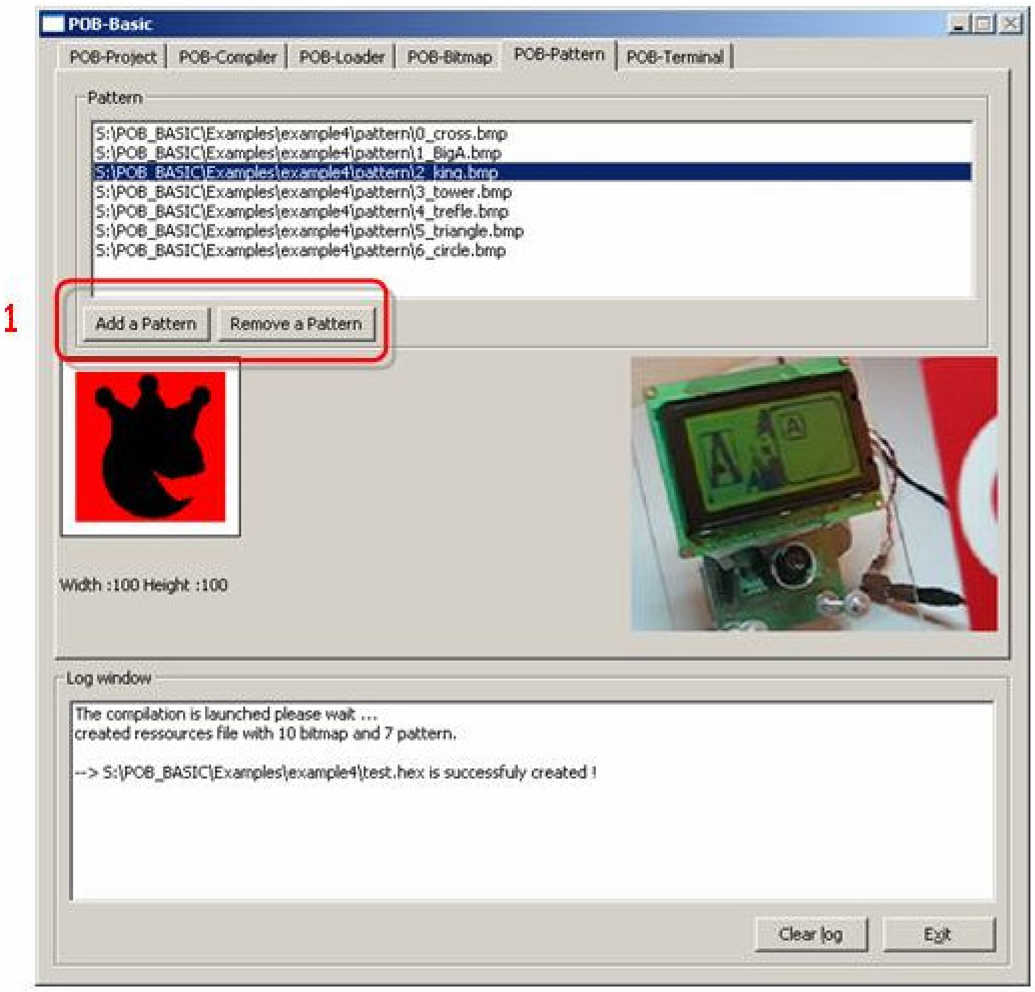
POB-Pattern allows you to create a pattern dictionary. This dictionary allows you to find patterns starting from the images of the POB-EYE camera.

POB-Pattern files is automatically includes in your application during the compilation (see POB-Compiler). The pattern recognition is the carried out using functions of the library supplied with POB-Basic tools.



The « *Identify* » function uses the camera frame and the pattern dictionary to recognize an image. As output, the function fills an array with the various forms recognized from the image.

Follow these steps to build your pattern file:



```
S:\POB_BASIC\Examples\example4\pattern\0_cross.bmp
S:\POB_BASIC\Examples\example4\pattern\1_BigA.bmp
S:\POB_BASIC\Examples\example4\pattern\2_king.bmp
S:\POB_BASIC\Examples\example4\pattern\3_tower.bmp
S:\POB_BASIC\Examples\example4\pattern\4_trefle.bmp
S:\POB_BASIC\Examples\example4\pattern\5_triangle.bmp
S:\POB_BASIC\Examples\example4\pattern\6_circle.bmp
```

Width :100 Height :100

```
The compilation is launched please wait ...
created ressources file with 10 bitmap and 7 pattern.

--> S:\POB_BASIC\Examples\example4\test.hex is successfuly created !
```

1 – Manage images:

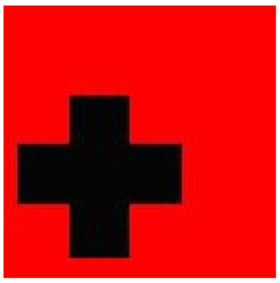
For add or remove an image, press the “*Add Bmp*” or “*Remove Bmp*” button.

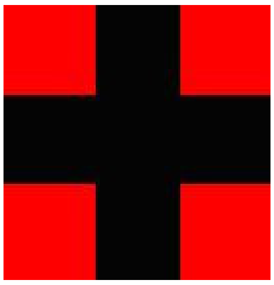
Remark: Image must respect the following format:
Bitmap 256 colors, **size of 100 per 100 pixels**.

The forms must be drawn in black with a red background

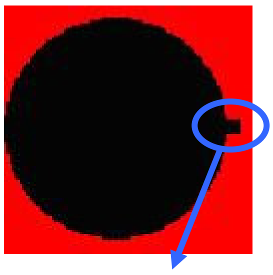
To obtain the best possible result during the creation of the dictionary, certain conditions should be observed:

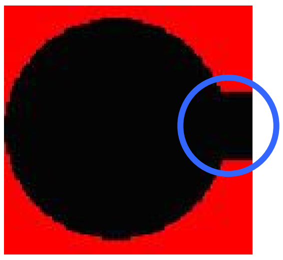
- The drawn image must take a maximum space within the framework of 100 per 100 pixels.


Drawing not wide enough

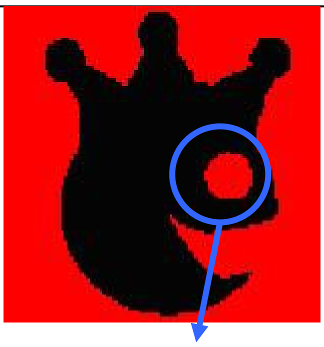

The space totally used

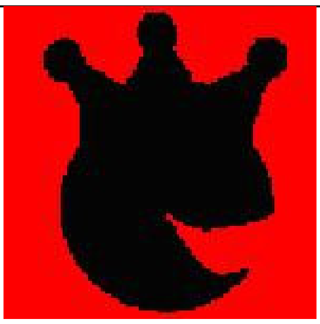
- It is necessary to avoid small details on the image.


Too small detail


Large enough detail

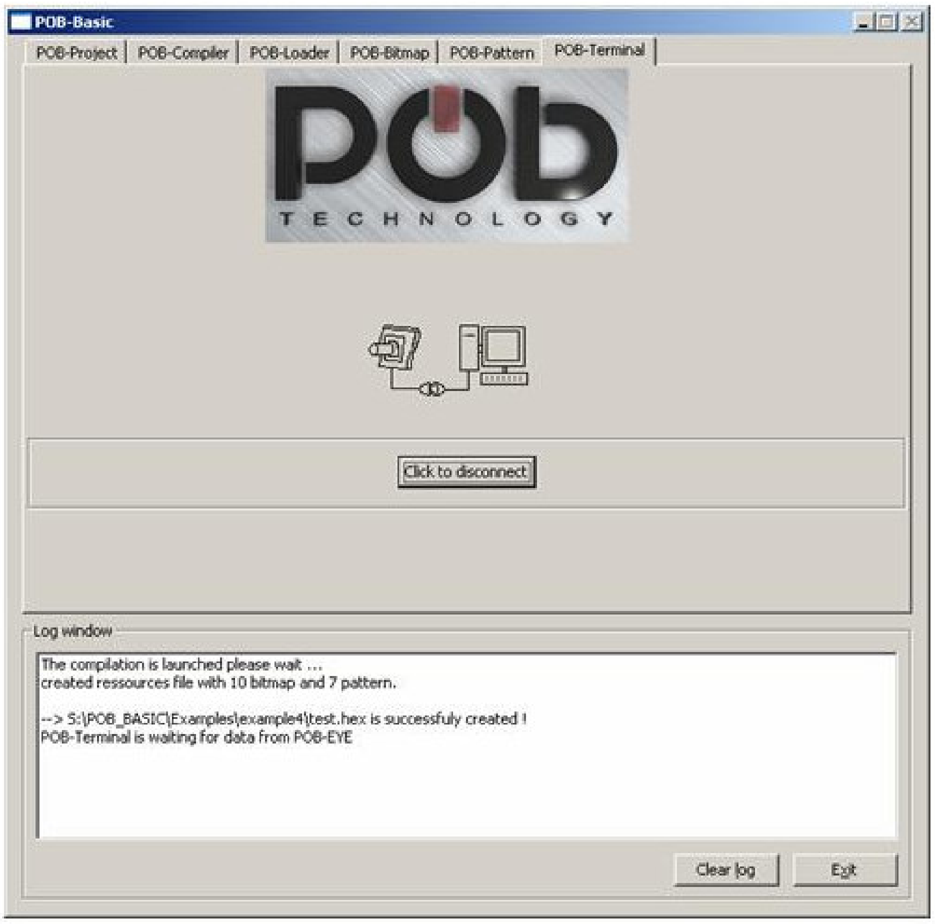
- The form should not have any empty space.


The form is empty!


Part without empty spaces

4.5 POB-Terminal

POB-Terminal interface allows you to display messages from POB-EYE module throughout the serial port. The aim of the POB-Terminal is to facilitate the development of your application.

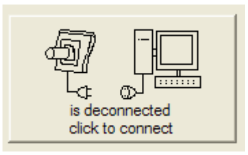


```
The compilation is launched please wait ...
created ressources file with 10 bitmap and 7 pattern.

--> S:\POB_BASIC\Examples\example4\test.hex is successfuly created !
POB-Terminal is waiting for data from POB-EYE
```

1 - Connecting POB-Terminal:

The “*Click to connect*” button is used to connect POB-Terminal to the POB-EYE module.

When POB-Terminal is connected, you can disconnect it with the “*Click to disconnect*” button.



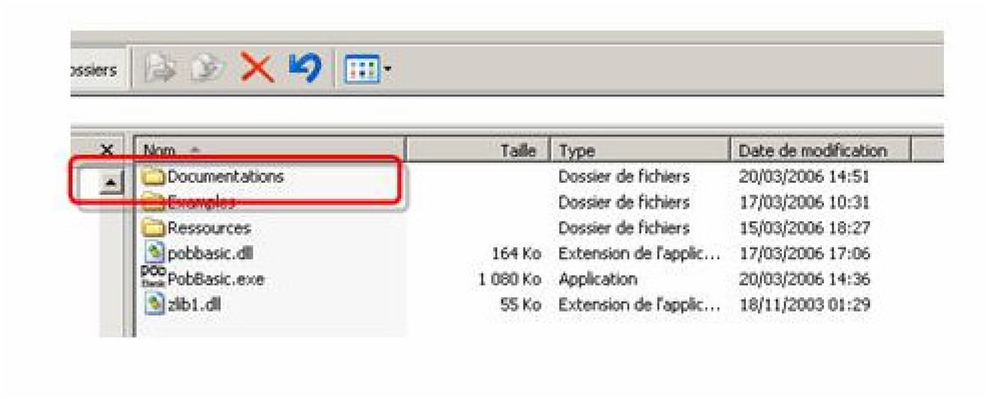
2 – Clearing messages:

You can clear the POB-Terminal messages with the “*Clear log*” button.

5 Basic Developement on POB-EYE

▪ *Basic library help*

Library documentation is in the “*Documentations*” repertory.



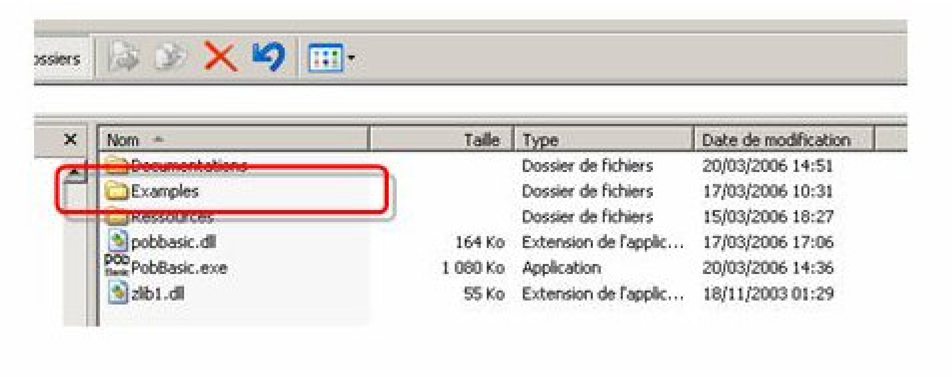
Nom	Taille	Type	Date de modification
Documentations		Dossier de fichiers	20/03/2006 14:51
Exemples		Dossier de fichiers	17/03/2006 10:31
Ressources		Dossier de fichiers	15/03/2006 18:27
pobbasic.dll	164 Ko	Extension de l'applic...	17/03/2006 17:06
PobBasic.exe	1 080 Ko	Application	20/03/2006 14:36
zlib1.dll	55 Ko	Extension de l'applic...	18/11/2003 01:29

▪ *Basic syntax help*

The basic syntax used in POBEYE is described in the “*Documentations*” repertory.

6 Sample application

All the examples are in the "*Examples*" repertory.

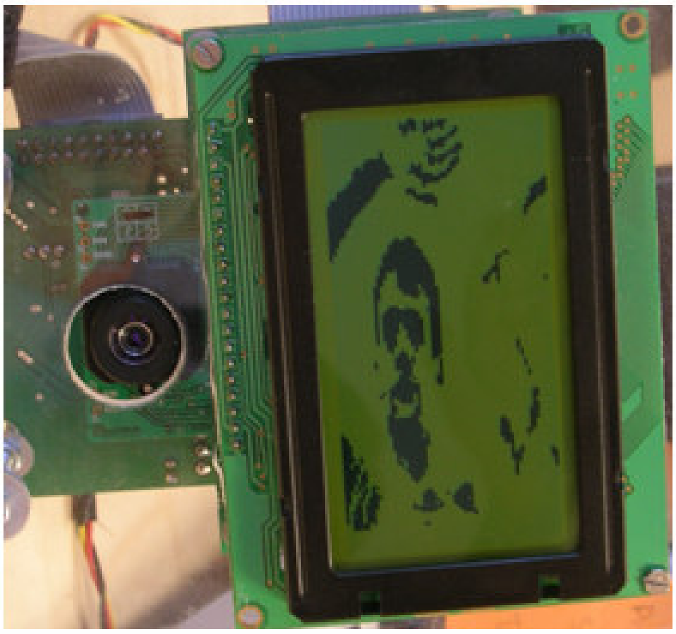


Nom	Taille	Type	Date de modification
Documentations		Dossier de fichiers	20/03/2006 14:51
Examples		Dossier de fichiers	17/03/2006 10:31
Ressources		Dossier de fichiers	15/03/2006 18:27
pobbasic.dll	164 Ko	Extension de l'applic...	17/03/2006 17:06
PobBasic.exe	1 080 Ko	Application	20/03/2006 14:36
zlib1.dll	55 Ko	Extension de l'applic...	18/11/2003 01:29

You will find a pob-basic project, source code and all resources for the project in the "*Examples*" folder.

Real time display on POB-LCD:

This example used the POB-EYE module and the POB-LCD128.



A program written in Basic must contain a unique Procedure « ***ProcedureMain*** ». This procedure will be the first to be called when executing the program.

```
Procedure ProcedureMain()
```

To use the POB-LCD128 screen you need to call « ***LCDInit*** » function.

```
LCDInit()
```

To draw on the screen you have to use a graphic buffer « ***NewGraphic*** ». All operation will be done on this buffer before being displayed on the screen.

The last parameter indicates if the buffer will use 1 bit per pixel (SLOWGRAPHIC, will save some memory however calculation speed will decrease) or 8 bits per pixel (FASTGRAPHIC, uses more memory but the graphic display is faster).

This function returns graphic buffer's number.

```
GraphicId.l = NewGraphic( 128,64, FASTGRAPHIC )
```

To retrieve images from camera you need to initialize the camera using the « ***NewRGBFrame*** » function.

```
NewRGBFrame()
```

Before using the graphic buffer you have to clear it.

```
ClearGraphic( GraphicId )
```

The program main loop will grab an image from the camera using « *GrabRGBFrame* », then will binarized the image before writing it in the graphic buffer in memory and display it on the LCD screen.

```
While 0=0
    GrabRGBFrame()
```

At the beginning the image is in color since the LCD is in black and white you have to binarize the image for display.

```
    BinaryFrame()
```

Then the graphic buffer is filled with one of the RGB component from the camera. In this example the red component was used. Nevertheless you might want to store the Green or Bleu component; anyway they are the same because they have been binarized previously.

```
    k.l = 0
    For i.l = 0 To 64

        For j.l = 0 To 120
```

The « ***GetRedPixel*** » function will grab a pixel from the red component.

```
            pixel.b = GetRedPixel( i+(j*88) )
```

The « ***SetGraphicPixel*** » function will write the pixel value in the graphic buffer.

```
            SetGraphicPixel( GraphicId , k , pixel )
            k = k +1
        Next
        k+=8;
    Next
```

Finally the graphic buffer is displayed on the screen.

```
    LCDDraw(GraphicId)

Wend
EndProcedure
```

▪ *Display images on POB-LCD128*

This program will display images on the LCD and uses all the graphical functions such a "*draw line, draw point...*"

A program written in Basic must contain a unique Procedure « ***ProcedureMain*** ». This procedure will be the first to be called when executing the program.

```
Procedure ProcedureMain()
```

You have to initialize the POB-LCD screen before use.

```
; Init the LCD screen
LCDInit()
```

All drawing operations are using a graphic buffer before being displayed.

```
; create a new graphic buffer
GraphicId.l = NewGraphic( 128, 64, FASTGRAPHIC)

ClearGraphic(GraphicId)
```

To display an image in graphic buffer you have to use « ***DrawBMP*** » function. Parameters are: Graphic buffer's number, X and Y coordinates on screen and the number of the image displayed.

```
; Draw at x=30, y=10 the bitmap 0 (bitmap 0 is the first bitmap in POB-Bitmap tools)
DrawBMP(GraphicId,30,10,0)

DrawBMP(GraphicId,45,10,1)
DrawBMP(GraphicId,65,10,2)
```

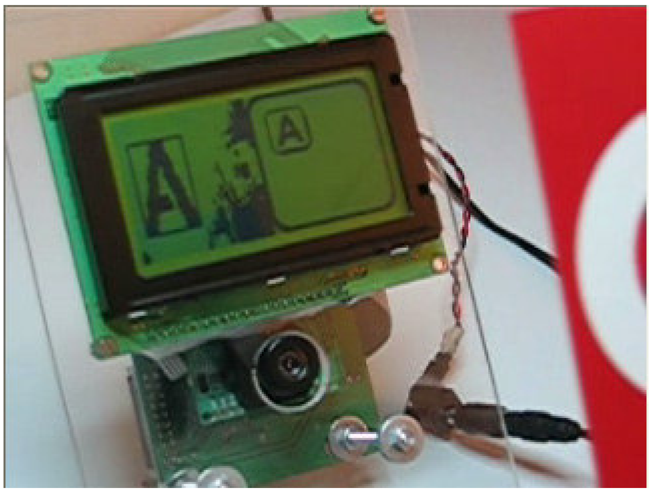
To draw a line, use the function « DrawALine » and for a dot use the function « DrawAPoint ».

```
    ; Draw at x=10, y=10 to x=25, y =30 a Line
    DrawALine(GraphicId,10,10,25,30)

    ; Draw at x=20, y=10 a Point
    DrawAPoint(GraphicId,20,10);

    ; Draw the graphic buffer On the LCD screen
    LCDDraw(GraphicId)

EndProcedure
```

▪ *Pattern recognition and advanced graphic buffer manipulation*

In this last example we are using 2 graphic buffers, a pattern recognition algorithm and a graphical interface.

The goal of this example is to do pattern recognition, to display the image on the left side of the screen and the pattern recognized on the right side of the screen.



```
Procedure ProcedureMain()
```

To store the recognized patterns from the camera we have created an array named *form* type « *Forms* » of a size 10.

Remark :

Access to any elements from the array is done using the following syntax :
« *UneForme = form(0)* » will grab From 0 from array.
To have access to the structure fields (in our case the structure is From) use the following syntax :
« *id = UneForme\id* » will grab form identifier of « UneForme »

```
; Array to store the form
Dim form.Forms(10)
```

We initialize the LCD screen.

```
LCDInit()
NewRGBFrame()
```

We are creating 2 graphic buffers:

```
; Create 2 graphic buffers : the Left For the camera frame, the Right For display recognized form
Left.l = NewGraphic( 64 ,64 ,FASTGRAPHIC )
Right.l = NewGraphic(64,64, SLOWGRAPHIC)

ClearGraphic(Left)
ClearGraphic(Right)

While 0 = 0
```

From camera, we grab the images and binarize :

```
    ; Get camera frame and binary it
    GrabRGBFrame()
    BinaryFrame()

    XPic.l=8
    YPic.l=8
```

In right buffer we draw a window.

```
    DrawBMP(Right,0,0,8 );
```

The function « *DrawCameraFrame* » can display directly an image from the camera in a 64 by 64 buffer.

```
    ; Draw current camera frame in the Left graphic buffer
    DrawCameraFrame( Left )
```

Finally we identify the patterns. Please note that the images have to be binarized before any recognition process. The « ***Identify*** » function takes as a parameter a pattern array and returns the number of the pattern recognized.

```
    ; Identify form in the current camera frame
    nbform.l = Identify( form )

    For i.l = 0 To nbform
```

We paste the right buffer in a frame with the image :

```
        DrawBMP(Right,XPic, YPic,9);

        DrawBMP(Right,XPic+3, YPic+3, form(i)\id -1 )

        XPic = XPic + 28
        If (XPic+28) > 64
            YPic += 26
            XPic = 8
        EndIf
    Next
```

Finally we draw the 2 screens POB-LCD128.

```
    ; Draw On the LCD the 2 graphic buffers
    LCDDrawRight( Right )
    LCDDrawLeft( Left )


Wend

EndProcedure
```

Contact POB-Technology

POB-TECHNOLOGY
4, rue nicéphore niépce
69 680 CHASSIEU
FRANCE

Web: www.pob-technology.com

Mail: contact@pob-technology.com

Phone: +33 (0)4 72 43 02 36
Fax: +33 (0)4 78 58 04 92