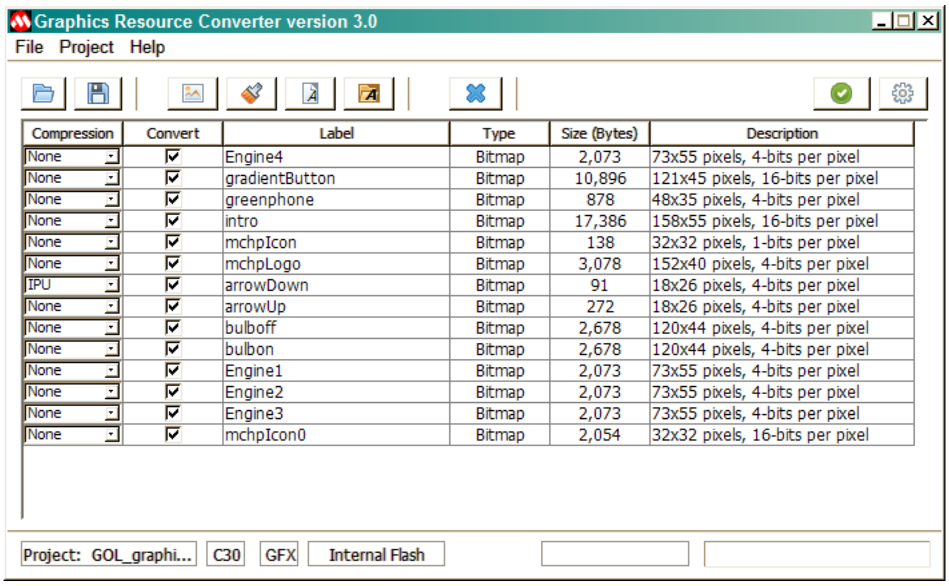
Graphics Resource Converter

Table of Contents

1 Graphics Resource Converter



Compression	Convert	Label	Type	Size (Bytes)	Description
None	✓	Engine4	Bitmap	2,073	73x55 pixels, 4-bits per pixel
None	✓	gradientButton	Bitmap	10,896	121x45 pixels, 16-bits per pixel
None	✓	greenphone	Bitmap	878	48x35 pixels, 4-bits per pixel
None	✓	intro	Bitmap	17,386	158x55 pixels, 16-bits per pixel
None	✓	mchpIcon	Bitmap	138	32x32 pixels, 1-bits per pixel
None	✓	mchpLogo	Bitmap	3,078	152x40 pixels, 4-bits per pixel
IPU	✓	arrowDown	Bitmap	91	18x26 pixels, 4-bits per pixel
None	✓	arrowUp	Bitmap	272	18x26 pixels, 4-bits per pixel
None	✓	bulboff	Bitmap	2,678	120x44 pixels, 4-bits per pixel
None	✓	bulbon	Bitmap	2,678	120x44 pixels, 4-bits per pixel
None	✓	Engine1	Bitmap	2,073	73x55 pixels, 4-bits per pixel
None	✓	Engine2	Bitmap	2,073	73x55 pixels, 4-bits per pixel
None	✓	Engine3	Bitmap	2,073	73x55 pixels, 4-bits per pixel
None	✓	mchpIcon0	Bitmap	2,054	32x32 pixels, 16-bits per pixel

Project: GOL_graphi... | C30 | GFX | Internal Flash

The utility allows converting images in bitmap (BMP extension) format and JPEG (JPG or JPEG extension) as well as operating system's installed fonts or True Type fonts directly from files (TTF extension) into formats to be used with Microchip Graphics Library. Bitmap and fonts are converted to a new optimized encoding for PIC microcontroller usage while the JPEG encoding of JPEG images are maintained.

Fonts maybe a copyrighted material so please ensure that you have the rights to use it. You may find free fonts distributed under Open Font License (OFL) agreement. Some of the fonts distributed under OFL may be found here http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&item_id=OFL_fonts.

Importing fonts into application can be performed in two ways. The first method is to identify a range of characters that you want to use and create a font table starting from the first character up to the last character in the range. The second method is to create a character filter file and based on the filter create a reduced character font table. The second method is effective in implementing multi-language applications using only a fraction of the memory required for a full font table implementation. This is especially true for Asian fonts such as Chinese, Japanese and Korean fonts.

Importing images also requires a step to convert images to BMP or JPEG format if the original format is of a different type. Multiple image editors that convert other formats to BMP or JPEG format are readily available from software vendors. Microsoft’s Paint application is one such image editor. For advanced image editing using an application called GIMP (www.gimp.org) is recommended because it supports capability to do generate optimized color palette for image which can reduce image size.

2 Release Notes

Microchip Graphics Resource Converter

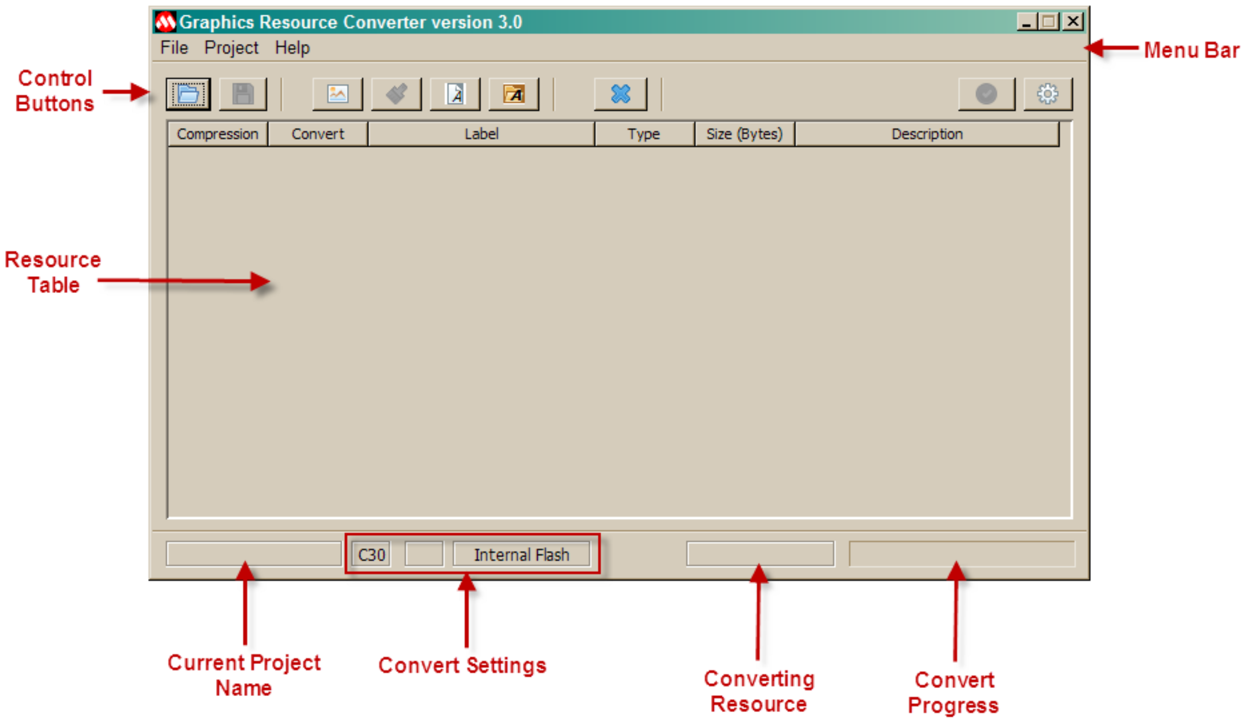
Version 3.00

This application is based on the JAVA programming language. To effectively run this program, the computer must have JRE 6 installed.

1. When importing images (BMP and JPEG), the user can select multiple files in the file dialog box to import.
2. Drag-and-drop is supported to import images.
3. The application supports selective resources to convert.
4. New installed font chooser.
5. Projects are saved in xml format.
6. Projects will save installed font information.
7. Source file output (C Array) for fonts, has the font character glyph in the comments.
8. Source file output has a table of contents for easier object searching.
9. Font filtering will check to make sure that the selected character range is supported by the selected font.
10. Multiple operating system platform support
11. New GUI interface.
12. Support for IPU unit.
    1. This is only supported by PIC microcontroller with graphics modules.
    2. Only supported on Microsoft Windows operating systems.
13. The font underline style has not been added to this version.
    1. The work around for this is to create the font without any underline style and after drawing the characters at runtime, use the primitive line function to add the underline.

3 Utility Interface

The main window of the Graphics Resource Converter utility has a list box to display items chosen for conversion and several control buttons.



Menu Bar

The Graphics Resource Converter menu bar can be used to add resources like images and font, as well as load and save projects, and there is access to help and support.

Control Buttons

The control buttons are used to add or remove resources, load and save projects, change convert settings and convert resources.

Resource Table

The resource table displays the current resources of a project.

Current Project

This label displays the current project that has been loaded. If there is no project currently loaded, the area will be blank.

Convert Settings

This set of labels displays the convert settings for the project.

Converting Resource

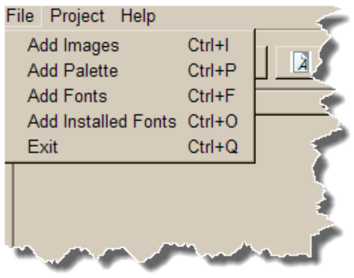
When the Graphic Resource Converter is converting resources, the current resource being converted is displayed.

Convert Progress

The overall progress of the when converting resources. This is based on the number of files and the relative size of the file to the overall project being converted.

3.1 Menu Bar

File



Add Images - Loads an image (BMP, JPG or JPEG) as a resource.

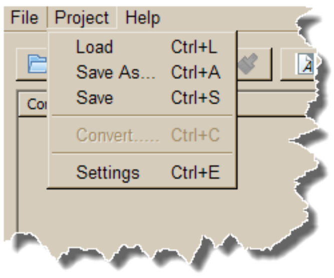
Add Palette - Loads a palette from a bitmap or GIMP file. This feature is only available for PIC microcontroller that have an internal graphics module.

Add Fonts - Loads a font from a true type font (.ttf) or Windows font file format (.fnt).

Add Installed Fonts - Loads a font from the operating system's list of installed fonts.

Exit - Quits the application

Project



Load - Loads a graphics resource project.

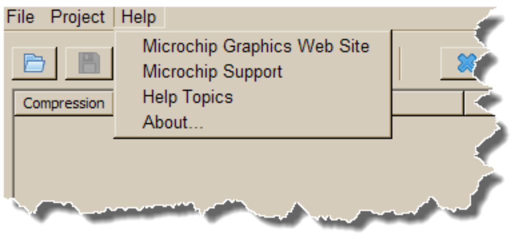
Save - Saves the current project.

Save As - Save the current project under a new project name.

Convert.... - Converts the project

Settings (see page 24) - Opens the settings dialog box.

Help



Microchip Graphics Web Site - Opens a web browser to the Microchip Graphics design center page.

Microchip Support - Opens a web browser to the Microchip Support log in page

Help Topics - Opens the Help documentation for the Graphics Resource Converter.

About.. - Opens the About window.

3.2 Control Buttons

Project Load - Loads a graphics resource project.

Project Save - Saves the current project.

Add Images - Loads an image (BMP, JPG or JPEG) as a resource.

Add Palette - Loads a palette from a bitmap or GIMP file. This feature is only available for PIC microcontroller that have an internal graphics module.

Add Fonts - Loads a font from a tty or fnt font file format.

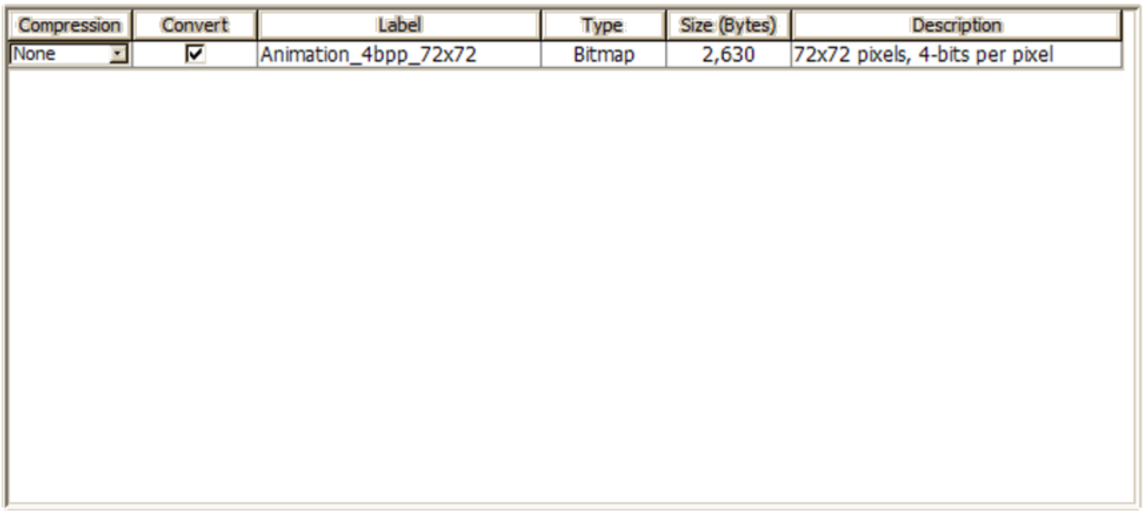
Add Installed Fonts - Loads a font from the operating system's list of installed fonts.

Remove Row - Removes selected row(s) from the resource table. This operation can not be reversed.

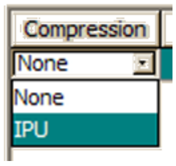
Convert - Converts the selected resources.

Settings (see page 24) - Sets the compiler, C30 or C32, graphics module and converted resource output.

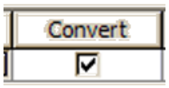
3.3 Resource Table



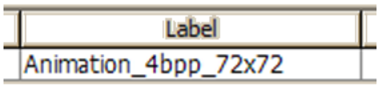
Compression	Convert	Label	Type	Size (Bytes)	Description
None	✓	Animation_4bpp_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel



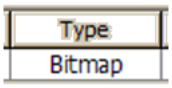
Compression - Indicates if any compression is used to reduce the size of a resource. This operation can only be used if the PIC microcontroller supports IPU compression and the resource is a bitmap image.



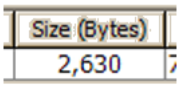
Convert - Converting resources, if this box is not checked, the resource will not be converted.



Label - The label that will be given to the resource that the Microchip Graphics Library will reference. For example, the label, Animation_4bpp_72x72, will be used by the Microchip Graphics Library.

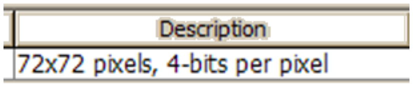


Type - The type of resource to be converted. Valid resource types are Bitmap, JPEG, Palette, Font, and Font File.



Size (Bytes)
2,630

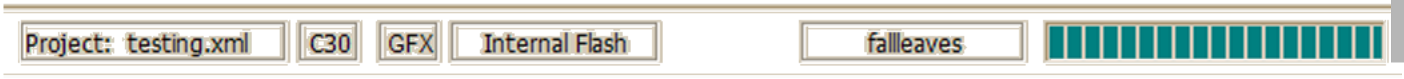
Size - The size, in bytes, of the converted resource. This size is representative of compressed or uncompressed resource.



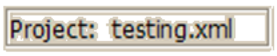
Description
72x72 pixels, 4-bits per pixel

Description - A description of the resource. For images, the description is the size, in pixels. For palettes, the description is the number of colors. For fonts, the description is the font character range and the height of the font in pixels. The user is responsible for not choosing C syntax names, i.e. char, short, int, or reserved keywords, i.e. for if, else.

3.4 Status

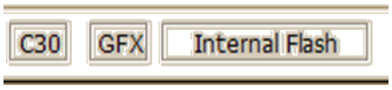


Project: testing.xml | C30 | GFX | Internal Flash | fallleaves



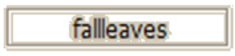
Project: testing.xml

Current Project - Shows the current project of the Graphics Resource Converter.



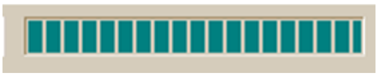
C30 | GFX | Internal Flash

Current Settings (see page 24) - Shows the current settings of the Graphics Resource Converter, the compiler, C30 or C32, graphics module, and what format the converted resources will be saved as.



fallleaves

Current Resource Being Converted - An indicator showing the current resource that is being converted.



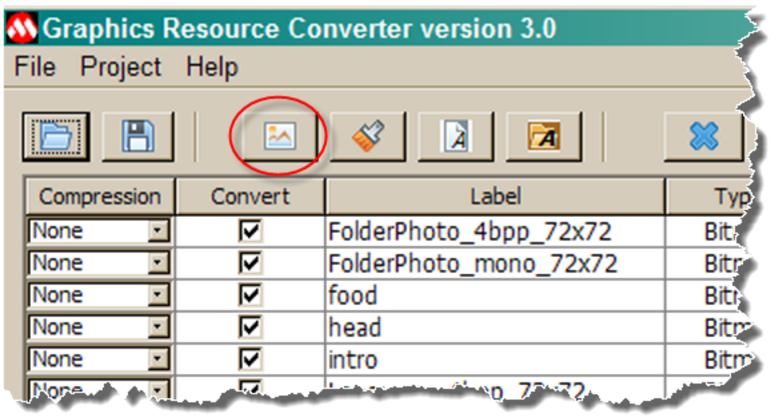
Convert Progress - The converting resource progress measured in bytes converted to total bytes to convert.

4 Using the Utility

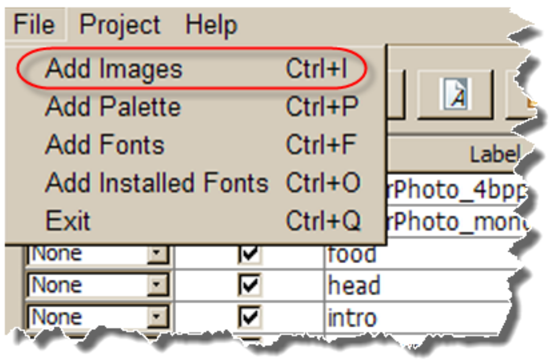
4.1 Adding Images to the Conversion List

To add an image (bitmap or JPEG format) for conversion the following steps should be done:

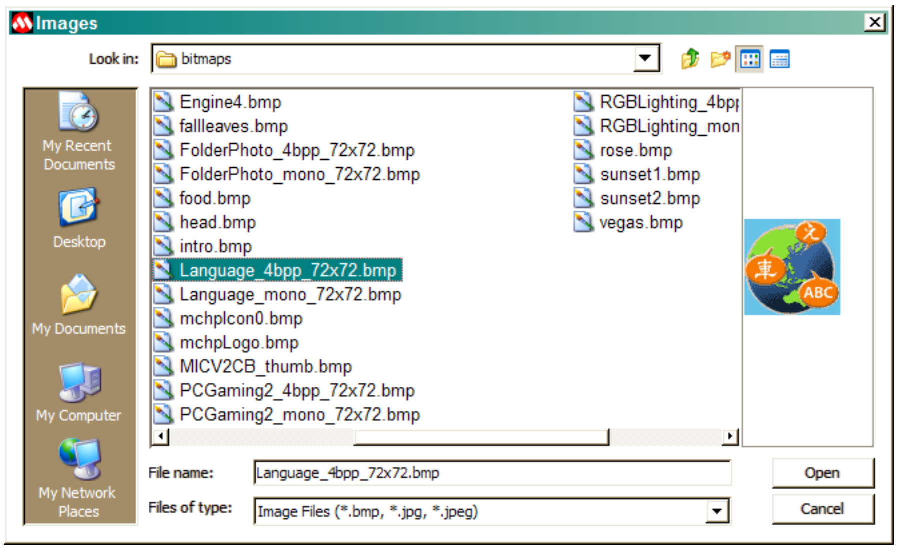
1. Press **Add Images** button or File menu item.



Compression	Convert	Label	Typ
None	✓	FolderPhoto_4bpp_72x72	Bit
None	✓	FolderPhoto_mono_72x72	Bitr
None	✓	food	Bitr
None	✓	head	Bitm
None	✓	intro	Bitm



2. A file dialog box will appear. The user can select a single or multiple files to import to the Graphics Resource Convert. When done, select the **Open** button.

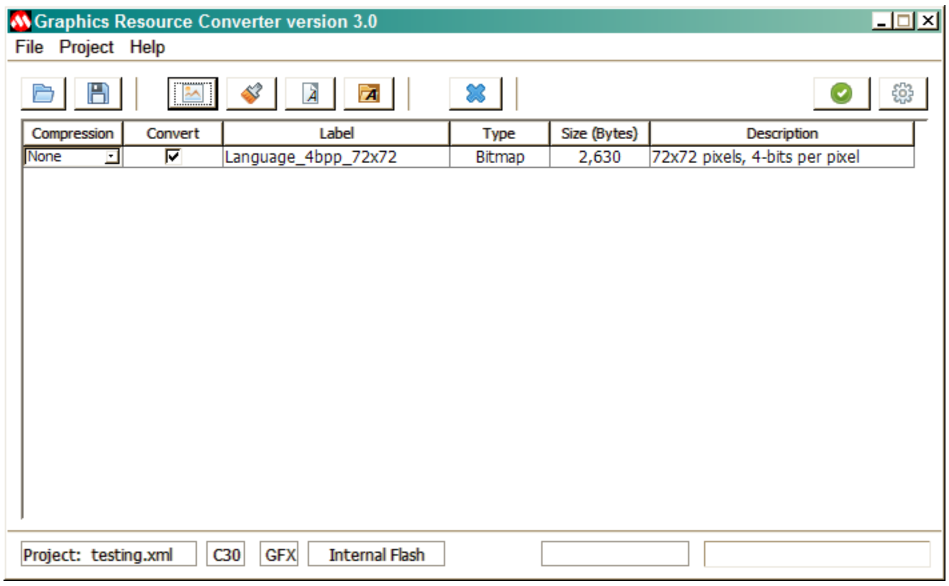


Look in: bitmaps

Engine4.bmp
fallleaves.bmp
FolderPhoto_4bpp_72x72.bmp
FolderPhoto_mono_72x72.bmp
food.bmp
head.bmp
intro.bmp
Language_4bpp_72x72.bmp
Language_mono_72x72.bmp
mchpIcon0.bmp
mchpLogo.bmp
MICV2CB_thumb.bmp
PCGaming2_4bpp_72x72.bmp
PCGaming2_mono_72x72.bmp
RGBLighting_4bp
RGBLighting_mon
rose.bmp
sunset1.bmp
sunset2.bmp
vegas.bmp

File name: Language_4bpp_72x72.bmp
Files of type: Image Files (*.bmp, *.jpg, *.jpeg)

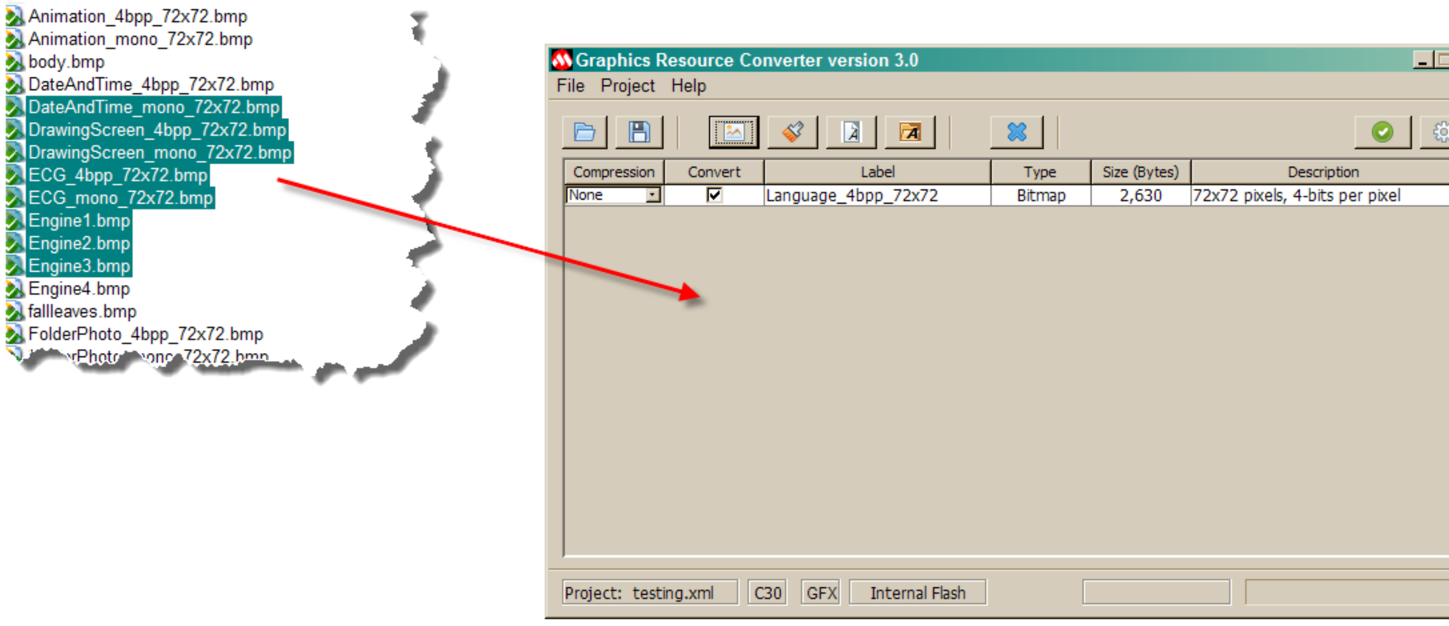
3. If selected file will be imported successfully the information about it (label, type, size and description) will be added in the list box of the main window. Label is generated from the file name, type defines that imported object is using a bitmap or a JPEG format, size of data is shown in bytes, and description contains basic information. Generated label must be used for the reference to this image in the application.



Compression	Convert	Label	Type	Size (Bytes)	Description
None	✓	Language_4bpp_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel

Project: testing.xml C30 GFX Internal Flash

4. Graphics Resource Converter also supports Drag-and-Drop for adding images. Simply select the images files and drag them to the resource table.

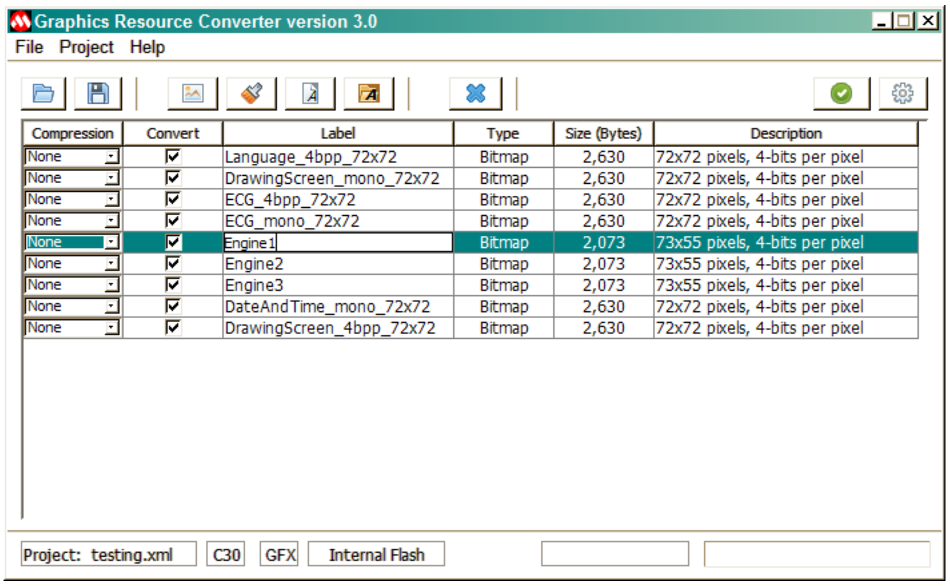


Animation_4bpp_72x72.bmp
Animation_mono_72x72.bmp
body.bmp
DateAndTime_4bpp_72x72.bmp
DateAndTime_mono_72x72.bmp
DrawingScreen_4bpp_72x72.bmp
DrawingScreen_mono_72x72.bmp
ECG_4bpp_72x72.bmp
ECG_mono_72x72.bmp
Engine1.bmp
Engine2.bmp
Engine3.bmp
Engine4.bmp
fallleaves.bmp
FolderPhoto_4bpp_72x72.bmp

Compression	Convert	Label	Type	Size (Bytes)	Description
None	✓	Language_4bpp_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel

Project: testing.xml | C30 | GFX | Internal Flash

4. To change the label double click on it and modify the label. The first label letter cannot be a number.



Compression	Convert	Label	Type	Size (Bytes)	Description
None	✓	Language_4bpp_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel
None	✓	DrawingScreen_mono_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel
None	✓	ECG_4bpp_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel
None	✓	ECG_mono_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel
None	✓	Engine1	Bitmap	2,073	73x55 pixels, 4-bits per pixel
None	✓	Engine2	Bitmap	2,073	73x55 pixels, 4-bits per pixel
None	✓	Engine3	Bitmap	2,073	73x55 pixels, 4-bits per pixel
None	✓	DateAndTime_mono_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel
None	✓	DrawingScreen_4bpp_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel

Project: testing.xml | C30 | GFX | Internal Flash

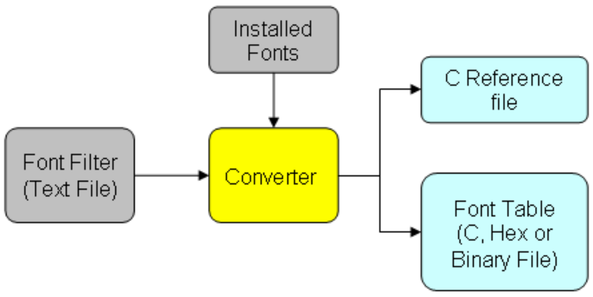
4.2 Adding Installed Fonts to the Conversion List

Importing Fonts using a Range of Characters

This option is normally done in languages with character sets of 255 characters or less. A range of characters is defined from one character index to the another character index. An example would be the ASCII character set where the English characters are defined from character index 32 or 0x20 hex (space character) to character index 126 or 0x7E hex(~ character). The font table generated for the range will contain all the characters from 0x20 to 0x7E inclusive. User's who want to use some other fonts with UNICODE character ranges can still be handled by this utility. The font table generated will still contain the characters with the defined range. Character index that has no defined characters glyph will be either a blank space or assigned to a default character. However, this approach is wasteful in terms of memory and it is recommended to use a font filter.

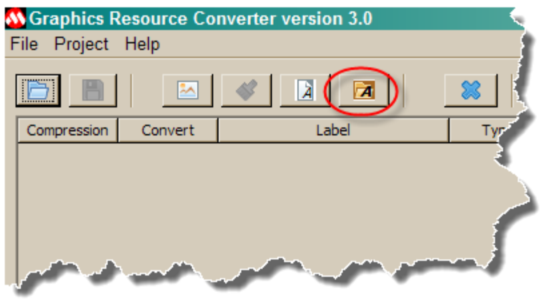
Importing Fonts using Character ID Filter

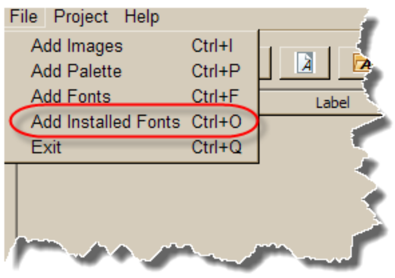
Another way to overcome the problem of memory requirements for fonts with thousands of character indexes is filtering the font table and recreating a reduced character set table which contains only the pre-defined characters. This approach will need a filter text file (described in Font Filter File Format (see page 33) section). The generated font table will contain all the character glyphs of the characters defined in the filter file. Except for the first character (character with the lowest character index) the character indexes of all the glyphs are changed. The utility will also generate a C source reference file to be used in the application to easily refer to the new font table. The figure below shows the general components in generating a reduced font table with its outputs.



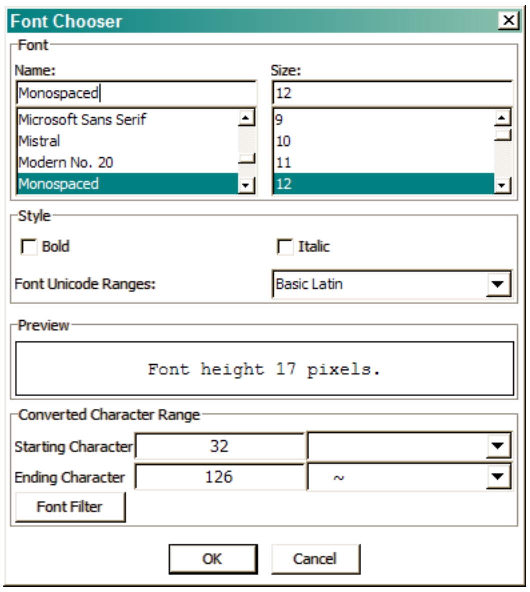
To add an installed font for conversion the following steps should be done:

1. Press **Add Installed Fonts** button or the File menu item. "Font Chooser" dialog will appear.

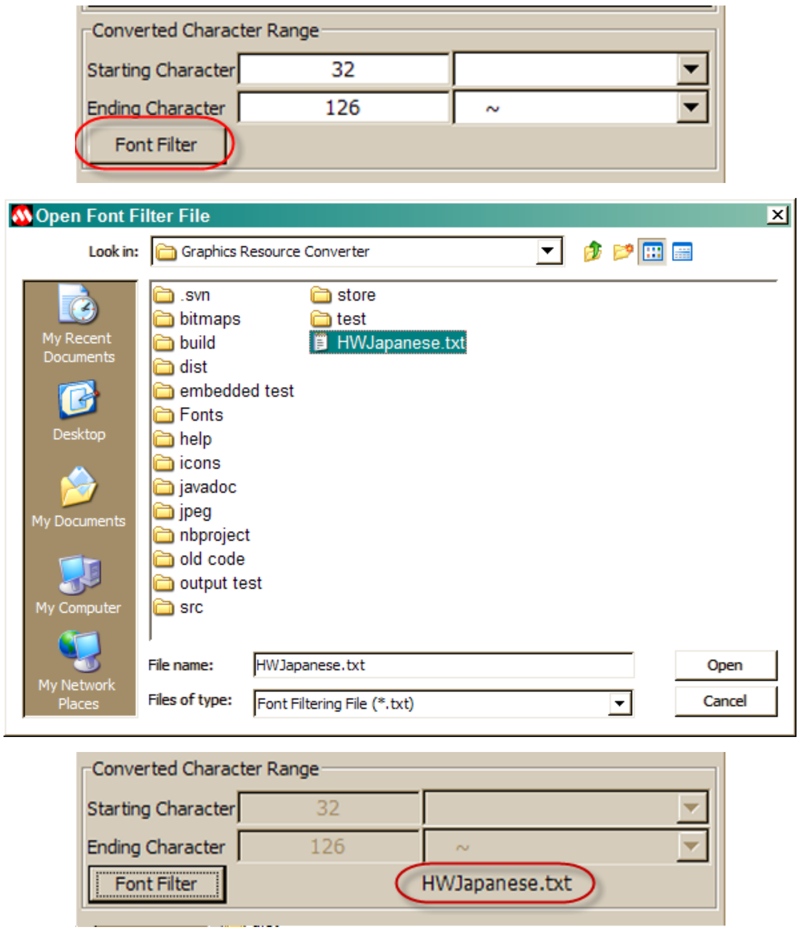




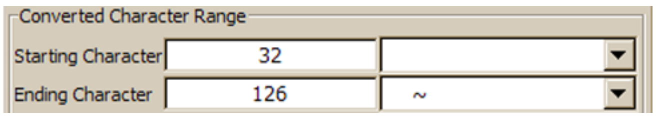
2. In the dialog select the installed fonts that will be used along with the "Font Style", "Font Size", "Unicode Range", "Starting Character", "Ending Character" and "Font Filter" button. An example of the selected font can be previewed with the height of the selected font in pixels.



3. A font filter can be selected by pressing the "Font Filter" button. When selecting the "Font Filter" button, a file dialog will appear. This gives the user the opportunity to assign a font character filter to the selected font. This filter will be used to extract character ID's that will comprise the font table.



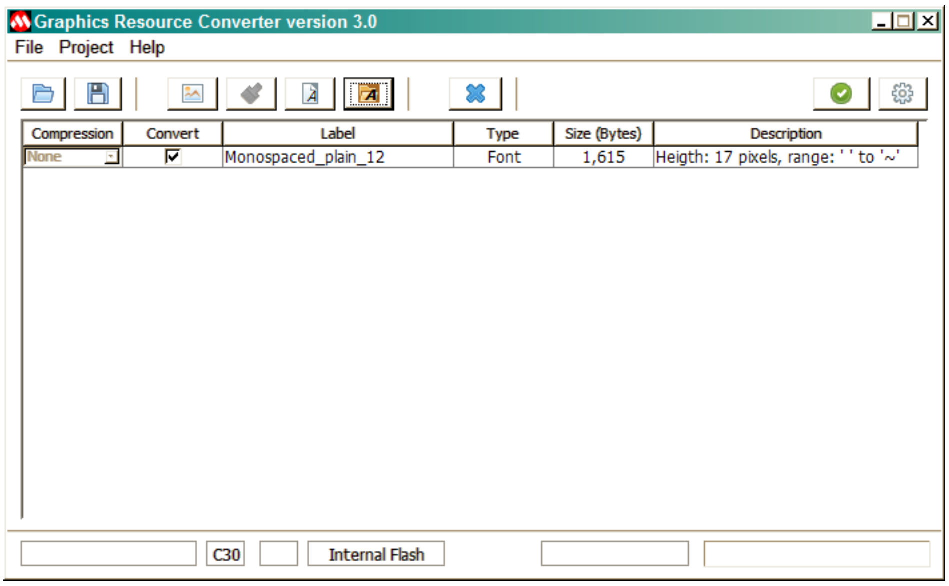
4. If filtering is not used, it is assumed by the utility that the user wants to use a range of characters instead. The user can choose a range of the characters that will be used in the font table. Starting Character sets the character index of the first character in the table. Ending Character sets the character index of the last character in the table.



Converted Character Range

Starting Character	32	
Ending Character	126	~

5. After selecting the font and all of it's attributes, press OK and it will appear in the resource table.



Compression	Convert	Label	Type	Size (Bytes)	Description
None	☑	Monospaced_plain_12	Font	1,615	Heigth: 17 pixels, range: ' ' to '~'

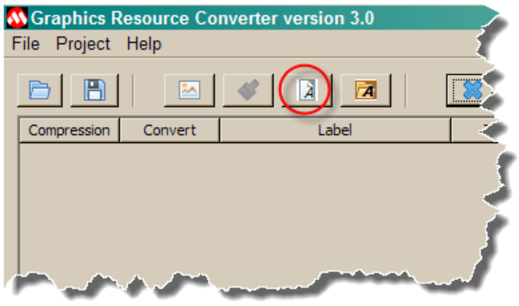
4.3 Adding Fonts from Files to the Conversion List

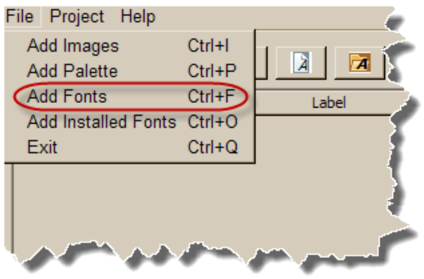
Importing Fonts from Files

Aside from the installed fonts, the user can also import a font table from a raster font file (*.fnt) or from a true type font file (*.ttf). Raster fonts can only import fonts with character sets ranging from 0-255. True type fonts on the other hand can be imported with the same options as installed fonts. They can be imported using a range of character ID or using a font filter file.

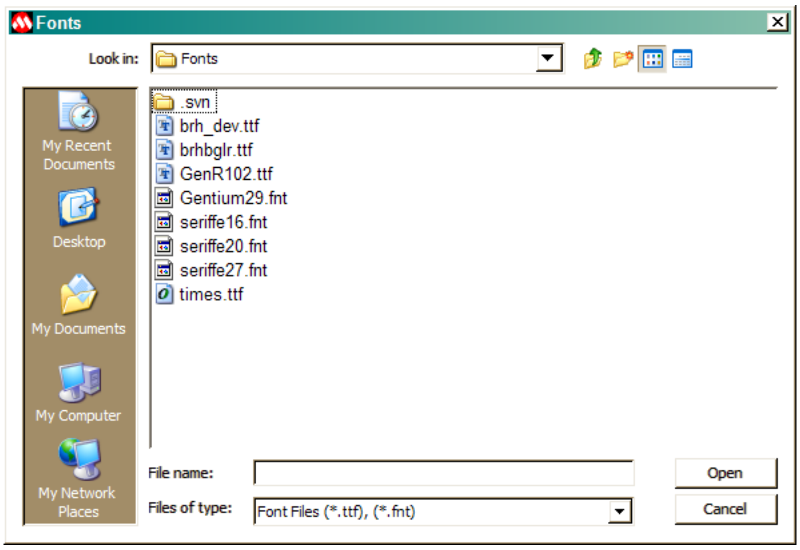
To add fonts using font files as inputs the following steps should be done:

1. Press **Add Fonts** button or File menu item. "Add font" dialog will appear.

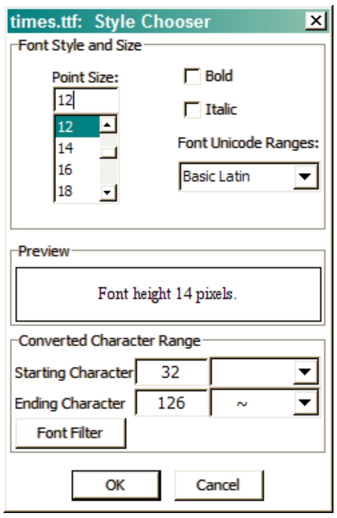




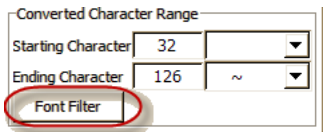
2. In the dialog select the file type extension to select importing raster font (*.fnt) or true type font (*.ttf). Browse and select the file and press **Open**.
   - Go to step 6 to generate raster fonts.
   - Go to step 3 to generate true type fonts.

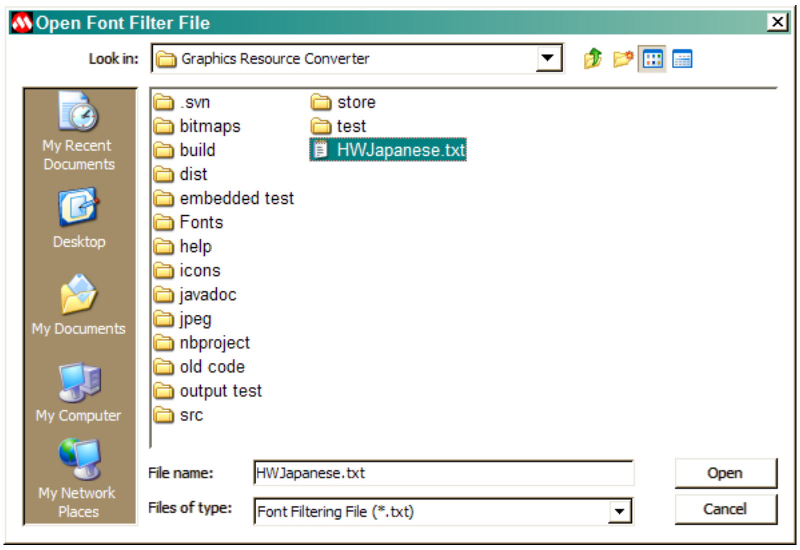


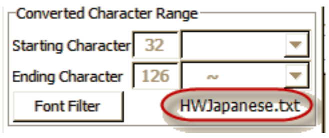
3. "Font Style Chooser" dialog will open. This dialog will show parameters for the selected true type font that can be set. Most importantly it shows the character sets or UNICODE ranges that the font supports.



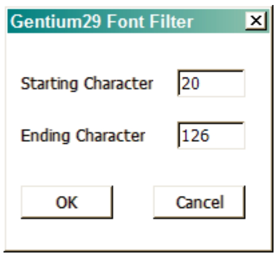
3. A font filter can be selected by pressing the "Font Filter" button. When selecting the "Font Filter" button, a file dialog will appear. This gives the user the opportunity to assign a font character filter to the selected font. This filter will be used to extract character ID's that will comprise the font table.


Converted Character Range
Starting Character: 32
Ending Character: 126 ~


Open Font Filter File
Look in: Graphics Resource Converter
.svn, bitmaps, build, dist, embedded test, Fonts, help, icons, javadoc, jpeg, nbproject, old code, output test, src, store, test, HWJapanese.txt
File name: HWJapanese.txt
Files of type: Font Filtering File (*.txt)


Converted Character Range
Starting Character: 32
Ending Character: 126 ~
Font Filter: HWJapanese.txt

5. Press **OK** to generate the font table.
6. When raster fonts are selected instead of true type fonts, instead of the "Font Style Chooser" dialog box "Font Filter" dialog box will appear.


Gentium29 Font Filter
Starting Character: 20
Ending Character: 126

7. Set the desired character range and press **OK** to generate the font table.

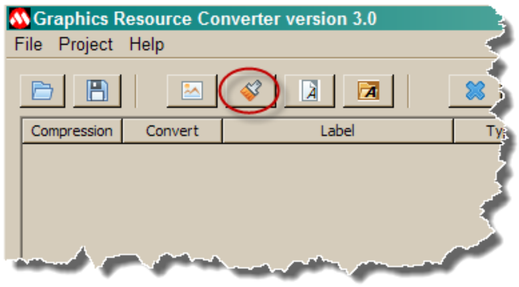
4.4 Adding Palettes to the Conversion List

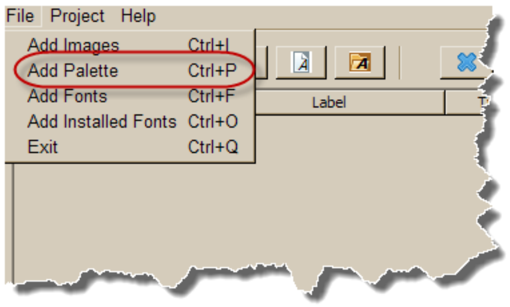
Importing Palettes

The tool can extract the color table of a bitmap file (bitmaps with 24-bit colors do not have color tables, the tool will not process these files) or load a GIMP palette file (*.gpl). To add a palette to the resource converter, the target must have an internal graphics module.

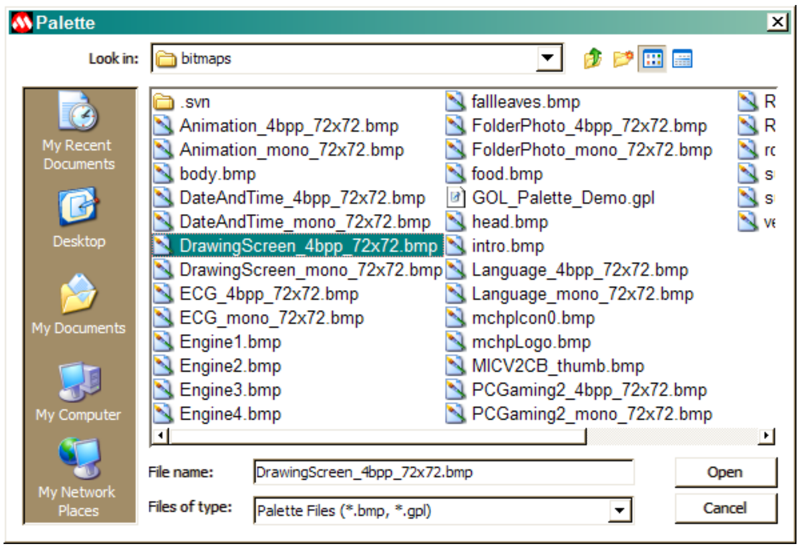
To load the color table of a bitmap file the following steps should be done:

1. Press **Add Palette** button or File menu item. "Open file" dialog will appear.

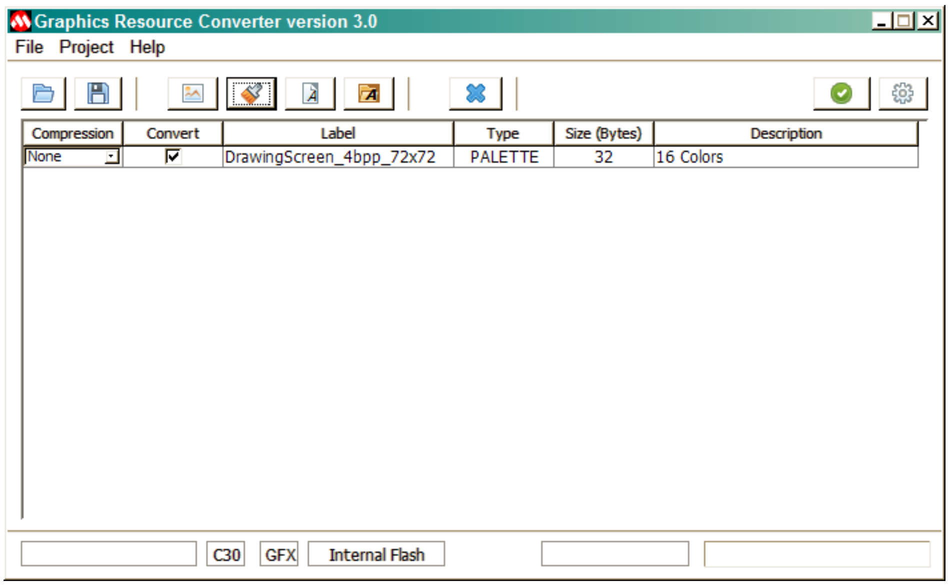




2. A file dialog will appear and by default, filters the Bitmap File (*.bmp filter) or the GIMP Palette file (*.gpl filter).Browse for the image file to be added and press **Open** button.

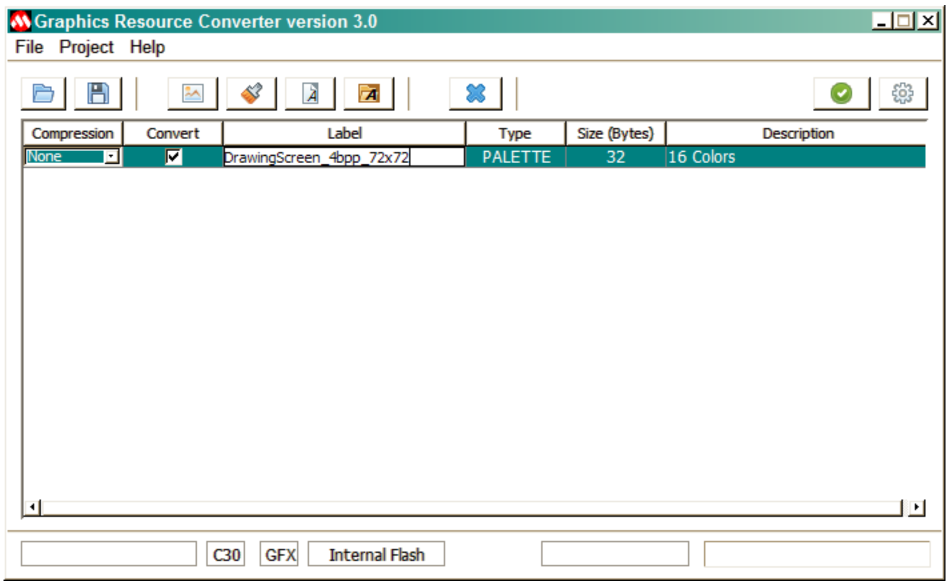


3. If selected file will be imported successfully the information about it (label, type, size and description) will be added in the list box of the main window. Label is generated from the file name, type defines that imported object is a palette, size of data is shown in bytes, and description contains the number of color entries in the palette. Generated label must be used for the reference to this image in the application.



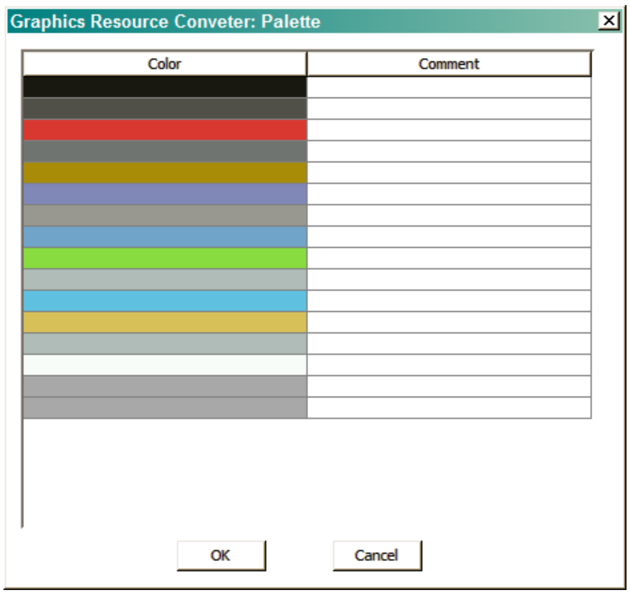
Compression	Convert	Label	Type	Size (Bytes)	Description
None	✓	DrawingScreen_4bpp_72x72	PALETTE	32	16 Colors

4. To change the label double click on it. Modify the label and press **OK** when done. The first label letter cannot be a number.

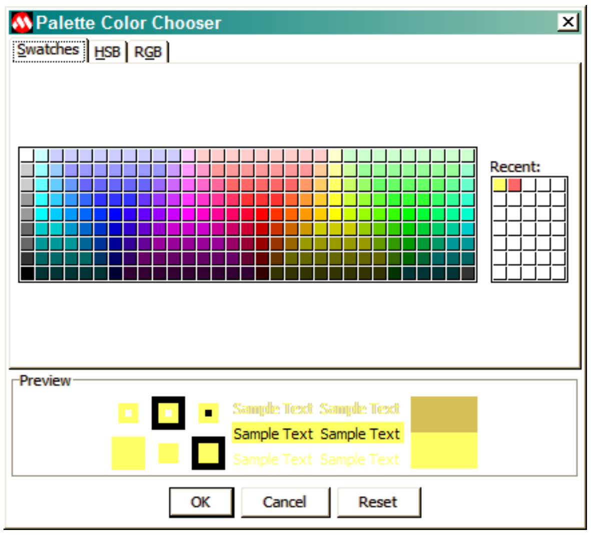


Compression	Convert	Label	Type	Size (Bytes)	Description
None	✓	DrawingScreen_4bpp_72x72	PALETTE	32	16 Colors

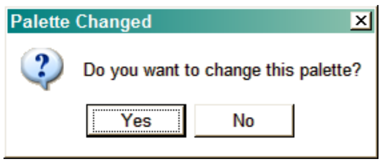
5. By double clicking on the palette entry a dialog box will appear to allow the user to edit the color table.



6. Double click on the color entry in the table to edit it. A color chooser will appear. Select the new color and select **OK**.



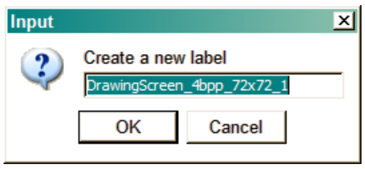
7. After done making edits, select **OK.** If you want to change the palette, select **YES**.



Palette Changed

Do you want to change this palette?

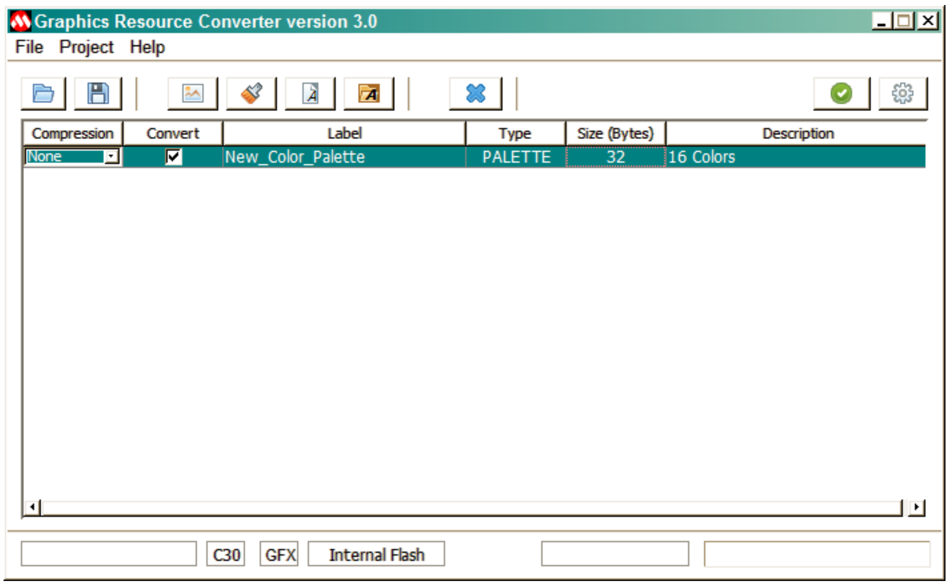
8. An input window will appear to create a new label.



Input

Create a new label

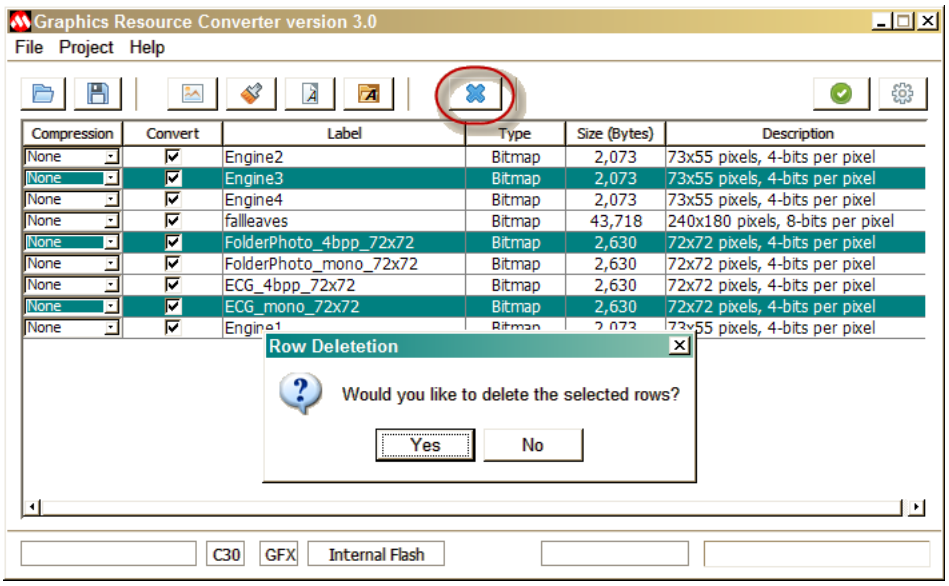
DrawingScreen_4bpp_72x72_1



Compression	Convert	Label	Type	Size (Bytes)	Description
None	✓	New_Color_Palette	PALETTE	32	16 Colors

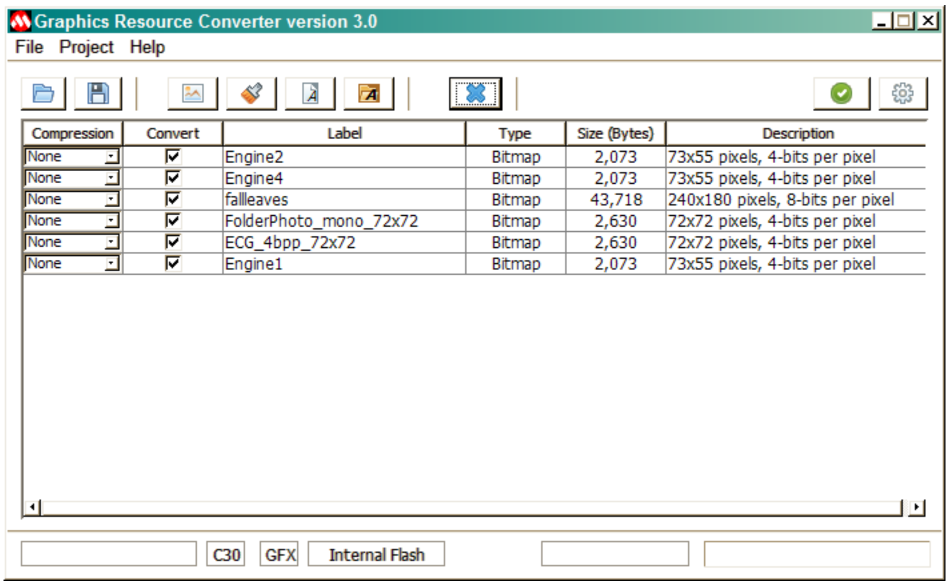
4.5 Removing Item from the Conversion List

To remove some item from conversion list select the item or items and press **Remove** button. This action can not be undone.



Compression	Convert	Label	Type	Size (Bytes)	Description
None	✓	Engine2	Bitmap	2,073	73x55 pixels, 4-bits per pixel
None	✓	Engine3	Bitmap	2,073	73x55 pixels, 4-bits per pixel
None	✓	Engine4	Bitmap	2,073	73x55 pixels, 4-bits per pixel
None	✓	fallleaves	Bitmap	43,718	240x180 pixels, 8-bits per pixel
None	✓	FolderPhoto_4bpp_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel
None	✓	FolderPhoto_mono_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel
None	✓	ECG_4bpp_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel
None	✓	ECG_mono_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel
None	✓	Engine1	Bitmap	2,073	73x55 pixels, 4-bits per pixel

Would you like to delete the selected rows?



Compression	Convert	Label	Type	Size (Bytes)	Description
None	✓	Engine2	Bitmap	2,073	73x55 pixels, 4-bits per pixel
None	✓	Engine4	Bitmap	2,073	73x55 pixels, 4-bits per pixel
None	✓	fallleaves	Bitmap	43,718	240x180 pixels, 8-bits per pixel
None	✓	FolderPhoto_mono_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel
None	✓	ECG_4bpp_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel
None	✓	Engine1	Bitmap	2,073	73x55 pixels, 4-bits per pixel

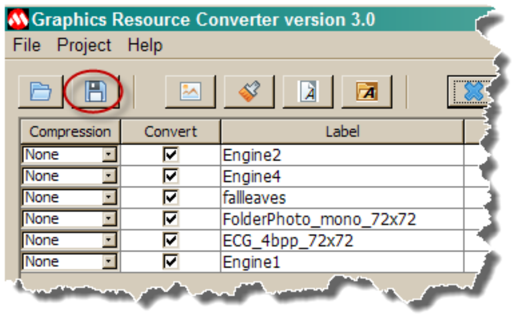
4.6 Saving and Loading Projects

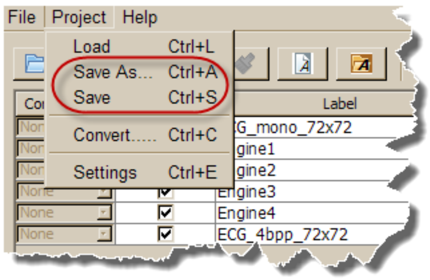
The utility allows users to save the loaded images, palettes and fonts into a project and to load a previously saved project.

Saving a Project

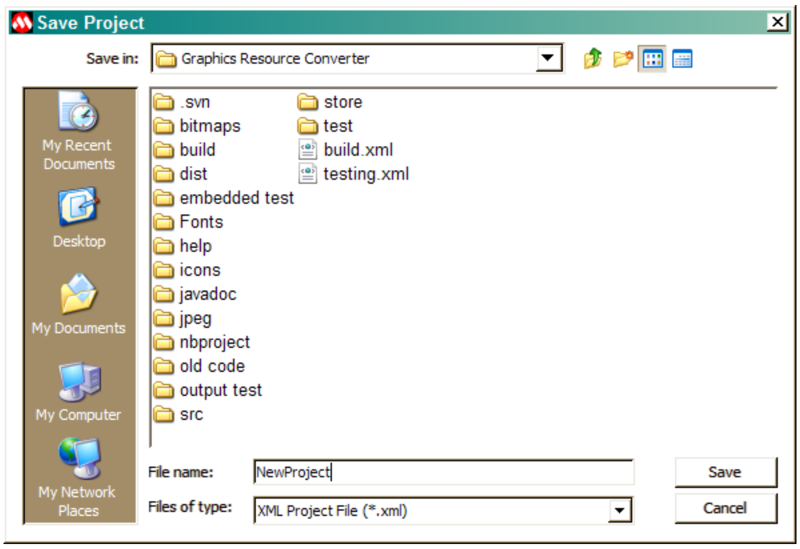
To save a project the following steps should be done:

1. Press **Save** button or Project Menu Item.





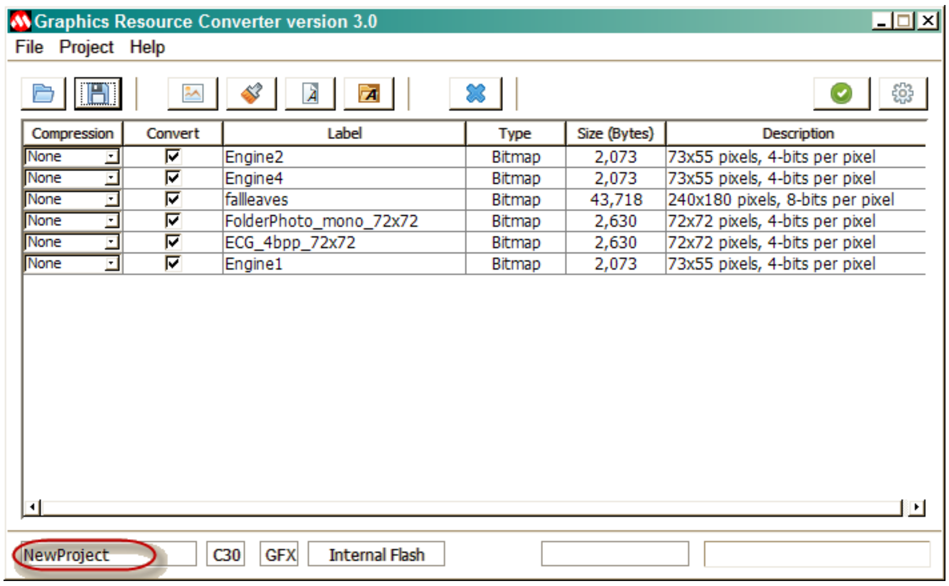
2. A Save As dialog will appear and, by default, filters the project file (*.xml filter).Type the new project file name or browse for the project file that will be overwritten and and press **Save** button.



File name: NewProject

Files of type: XML Project File (*.xml)

3. The saved project will create a *.xml file and the project name will be placed in the status bar.



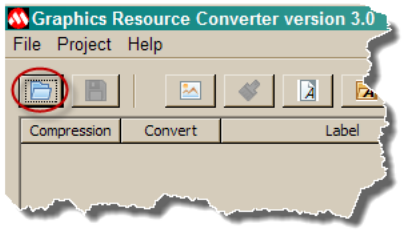
Compression	Convert	Label	Type	Size (Bytes)	Description
None	✓	Engine2	Bitmap	2,073	73x55 pixels, 4-bits per pixel
None	✓	Engine4	Bitmap	2,073	73x55 pixels, 4-bits per pixel
None	✓	fallleaves	Bitmap	43,718	240x180 pixels, 8-bits per pixel
None	✓	FolderPhoto_mono_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel
None	✓	ECG_4bpp_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel
None	✓	Engine1	Bitmap	2,073	73x55 pixels, 4-bits per pixel

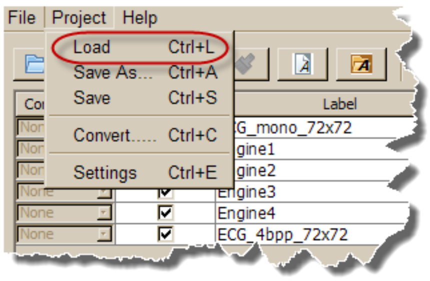
NewProject C30 GFX Internal Flash

Loading a Project

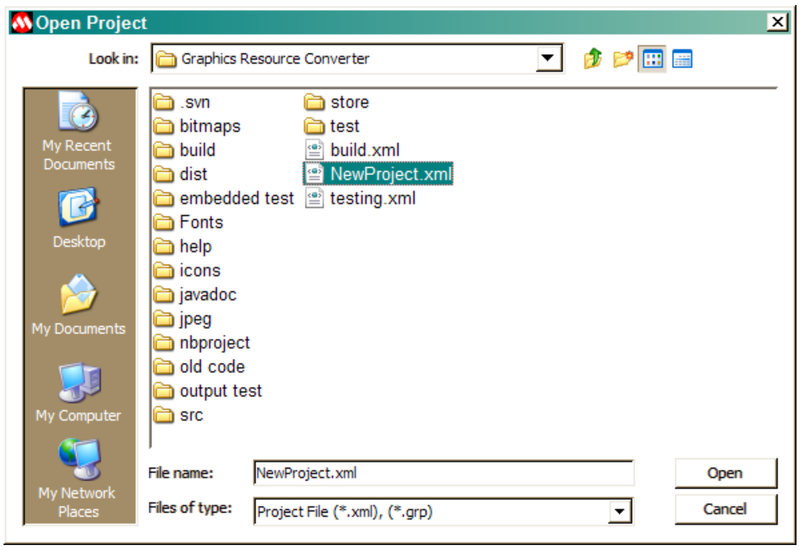
To load a project the following steps should be done:

1. Press **Load** button or the Project menu item.





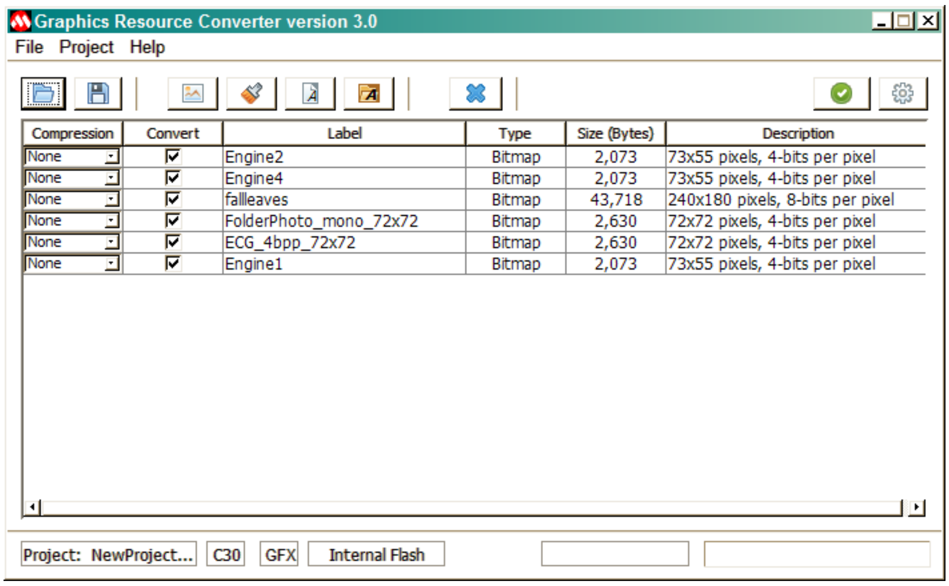
2. A Open dialog will appear and, by default, filters the project file (*.grp filter or *.xml filter).Type the project file name or browse for the project file that will be loaded and and press **Open** button.



File name: NewProject.xml

Files of type: Project File (*.xml), (*.grp)

3. After selecting the project, the resource table will be populated and the status will be updated.



Compression	Convert	Label	Type	Size (Bytes)	Description
None	✓	Engine2	Bitmap	2,073	73x55 pixels, 4-bits per pixel
None	✓	Engine4	Bitmap	2,073	73x55 pixels, 4-bits per pixel
None	✓	fallleaves	Bitmap	43,718	240x180 pixels, 8-bits per pixel
None	✓	FolderPhoto_mono_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel
None	✓	ECG_4bpp_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel
None	✓	Engine1	Bitmap	2,073	73x55 pixels, 4-bits per pixel

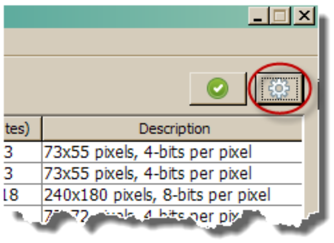
Project: NewProject... C30 GFX Internal Flash

4.7 Settings

The Graphics Resource Converter can be configured for difference types of converting mediums along with different device builds.

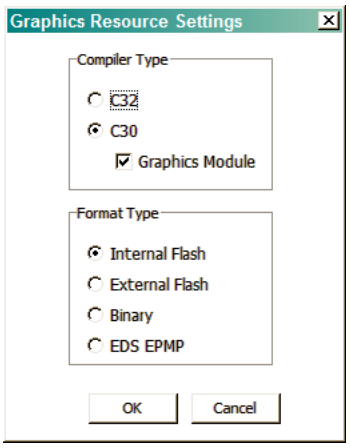
To change the Graphics Resource Converter's settings

1. Press the **Settings** button.



tes)	Description
3	73x55 pixels, 4-bits per pixel
3	73x55 pixels, 4-bits per pixel
18	240x180 pixels, 8-bits per pixel

2. A Graphics Resource Setting dialog box will appear.



Graphics Resource Settings

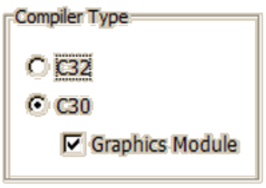
Compiler Type

( ) C32
(•) C30
[✓] Graphics Module

Format Type

(•) Internal Flash
( ) External Flash
( ) Binary
( ) EDS EPMP

3. The Compiler Type panel contains the device build type
   1. C32 - Select this to convert for PIC32MX devices
   2. C30 - Select this to convert for PIC24 and dsPIC devices.
   3. Checking the **Graphics Module** will indicate that the device also has an internal graphics module.

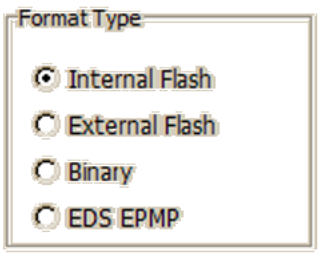


Compiler Type

( ) C32
(•) C30
[✓] Graphics Module

4. The Format Type panel contains the medium for the converted resources
   1. Internal Flash - The converted resources will be stored as C array structures. Proper device resources are needed to store these C array structures.
   2. External Flash - The converted resources will be stored as a Intel HEX file. This HEX file can be uploaded to an external memory source. The device will access this memory source to retrieve the resource data.

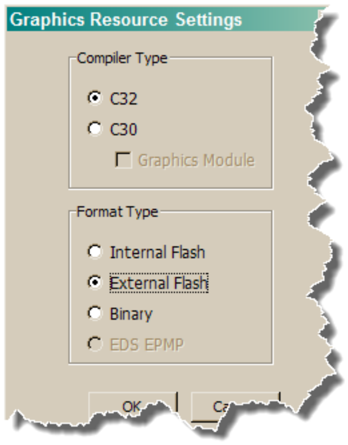
3. Binary - The converted resources will be converted into a binary file.
4. EDS EPMP - On devices with EDS memory space, the converted resources can be placed into this EDS space.



Format Type

[✓ Internal Flash]
External Flash
Binary
EDS EPMP

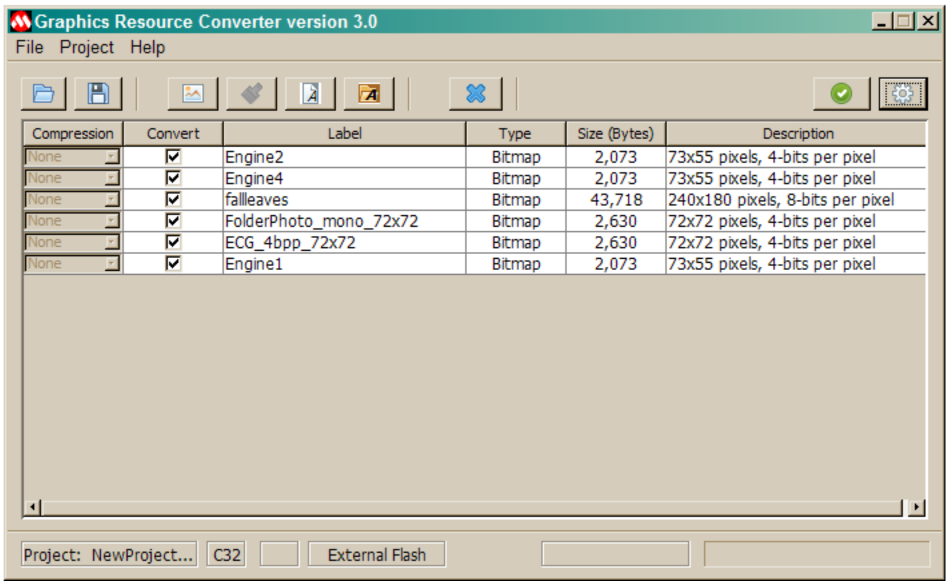
5. After selecting the desired settings, select **OK** and the status bar will be populated.



Graphics Resource Settings

Compiler Type
[✓ C32]
C30
Graphics Module

Format Type
Internal Flash
[✓ External Flash]
Binary
EDS EPMP



Graphics Resource Converter version 3.0

Compression	Convert	Label	Type	Size (Bytes)	Description
None	✓	Engine2	Bitmap	2,073	73x55 pixels, 4-bits per pixel
None	✓	Engine4	Bitmap	2,073	73x55 pixels, 4-bits per pixel
None	✓	fallleaves	Bitmap	43,718	240x180 pixels, 8-bits per pixel
None	✓	FolderPhoto_mono_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel
None	✓	ECG_4bpp_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel
None	✓	Engine1	Bitmap	2,073	73x55 pixels, 4-bits per pixel

Project: NewProject... C32 External Flash

4.8 Converting

4.8.1 Converting into C file

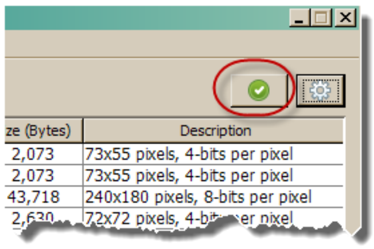
There is a limitation on the memory used when generating font images in C files (to be stored in internal flash for PIC24 devices). The font images are placed in the const section. The const section has a maximum size of 32 Kbytes. Therefore, the maximum size that the font image can have is 32 Kbyte assuming that no other data will reside in the const section. When generating more than one font images, the total size of all the font images must fit into the 32Kbyte space. If one of the font image requirement exceeds 32 Kbytes, that font image must be stored in external memory. When stored in external memory, the limitation will be the external memory size. The reason for storing fonts in the const section in internal flash is performance. It is faster to retrieve and display characters in the screen when placed in the const section.

For PIC32, there is no limitation. As long as the internal flash has space you can pack in more font images.

Conversion of C file containing arrays to be located in internal flash memory is similar to the conversion of the Hex file.

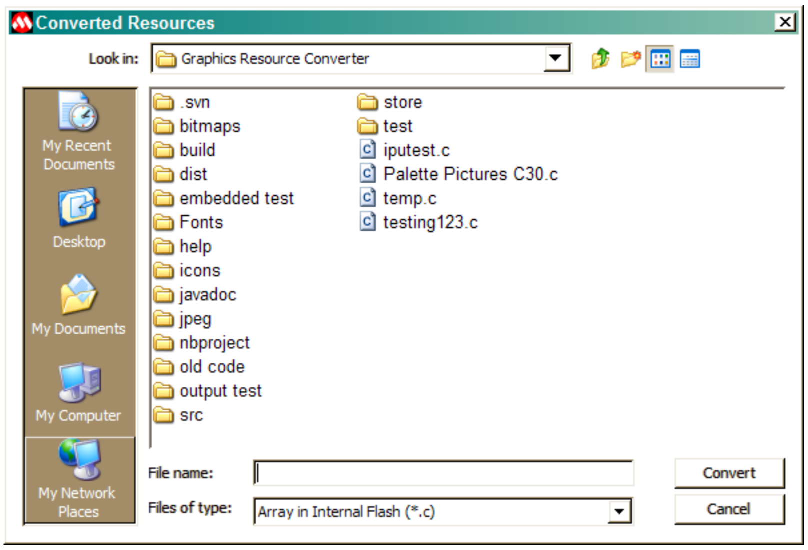
This conversion type should be used to store fonts and bitmaps into internal flash memory. The following steps must be performed:

1. Press **Convert** button.



ze (Bytes)	Description
2,073	73x55 pixels, 4-bits per pixel
2,073	73x55 pixels, 4-bits per pixel
43,718	240x180 pixels, 8-bits per pixel
2,630	72x72 pixels, 4-bi er pixel

2. “Converted Resources” dialog will appear.



Converted Resources

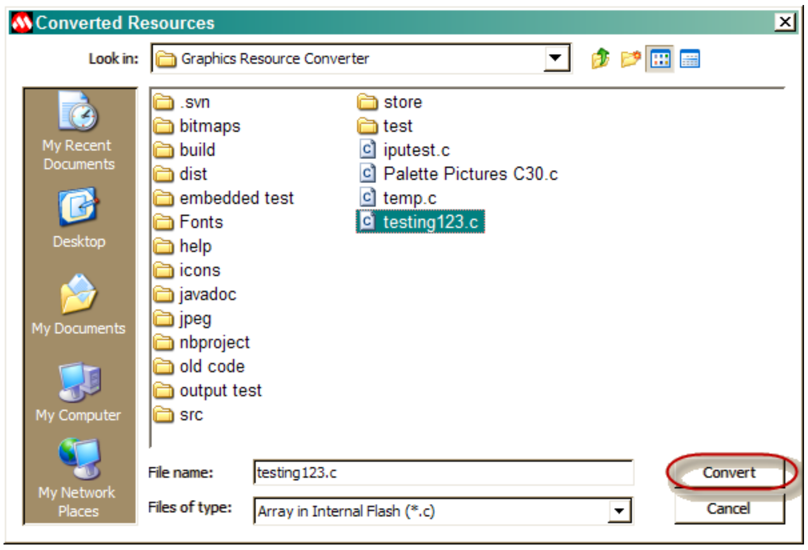
Look in: Graphics Resource Converter

.svn
bitmaps
build
dist
embedded test
Fonts
help
icons
javadoc
jpeg
nbproject
old code
output test
src
store
test
iputest.c
Palette Pictures C30.c
temp.c
testing123.c

File name:

Files of type: Array in Internal Flash (*.c)

3. Press **Convert** button.



Converted Resources

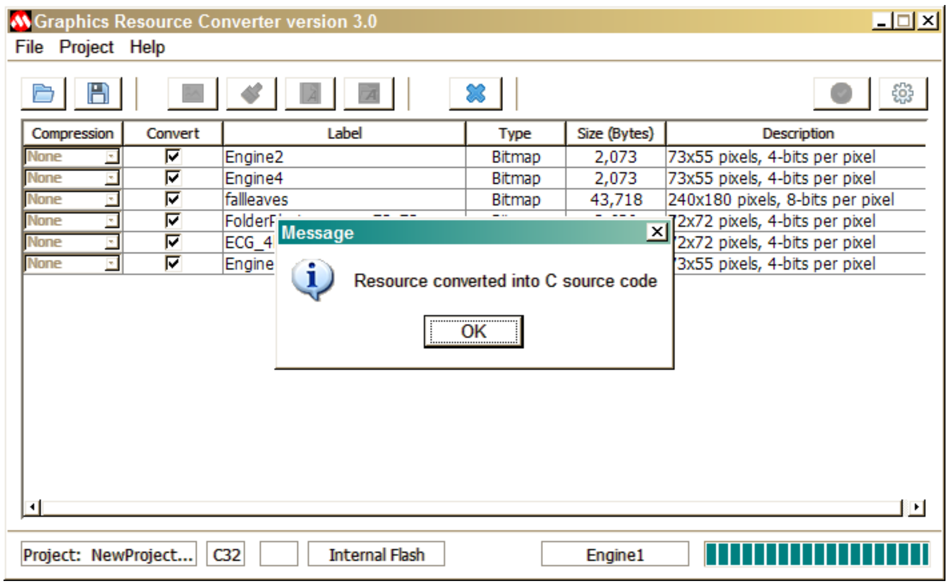
Look in: Graphics Resource Converter

.svn
bitmaps
build
dist
embedded test
Fonts
help
icons
javadoc
jpeg
nbproject
old code
output test
src
store
test
iputest.c
Palette Pictures C30.c
temp.c
testing123.c

File name: testing123.c

Files of type: Array in Internal Flash (*.c)

4. When the resources have been converted, a dialog will appear indicating such.


Graphics Resource Converter version 3.0

File Project Help

Compression	Convert	Label	Type	Size (Bytes)	Description
None	✓	Engine2	Bitmap	2,073	73x55 pixels, 4-bits per pixel
None	✓	Engine4	Bitmap	2,073	73x55 pixels, 4-bits per pixel
None	✓	fallleaves	Bitmap	43,718	240x180 pixels, 8-bits per pixel
None	✓	FolderF			72x72 pixels, 4-bits per pixel
None	✓	ECG_4			72x72 pixels, 4-bits per pixel
None	✓	Engine			73x55 pixels, 4-bits per pixel

Message

Resource converted into C source code

OK

Project: NewProject... C32 Internal Flash Engine1

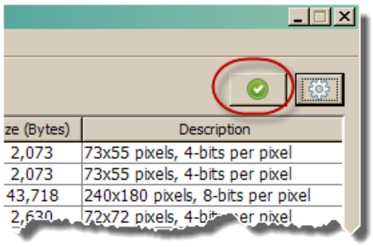
4.8.2 Conversion into Intel Hex File

This output format should be selected to store bitmaps and fonts in external memory. Address of each object will be aligned by 4 bytes border. In addition to Intel hex file the utility will generate C file containing structures FONT_EXTERNAL and IMAGE_EXTERNAL. These structures must be used in application to access converted pictures and fonts in external memory. Name of the reference C file will be the same as name of the hex file with ".c" appended. This file will be compiled with the application to enable an easy reference to the strings that will be used in the application. Please refer to Font Reference File Output (see page 34) description.

4

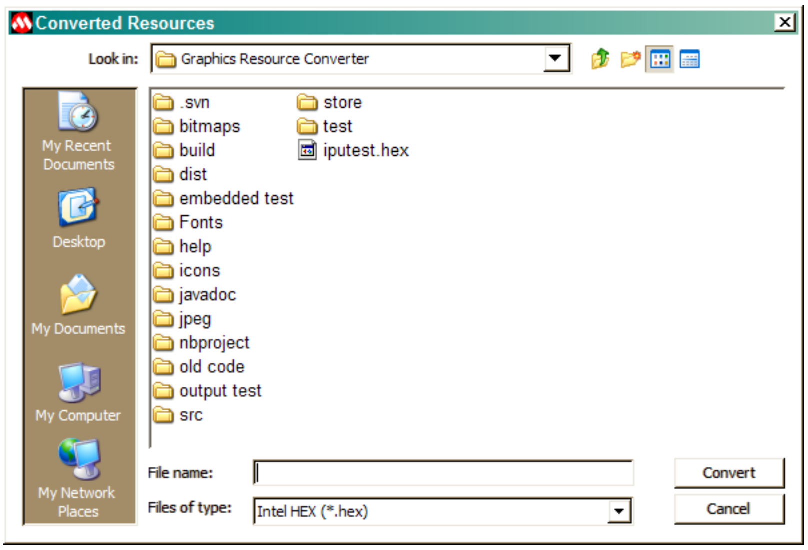
To create Intel Hex and reference C files following steps must be performed:

1. Press **Convert** button.

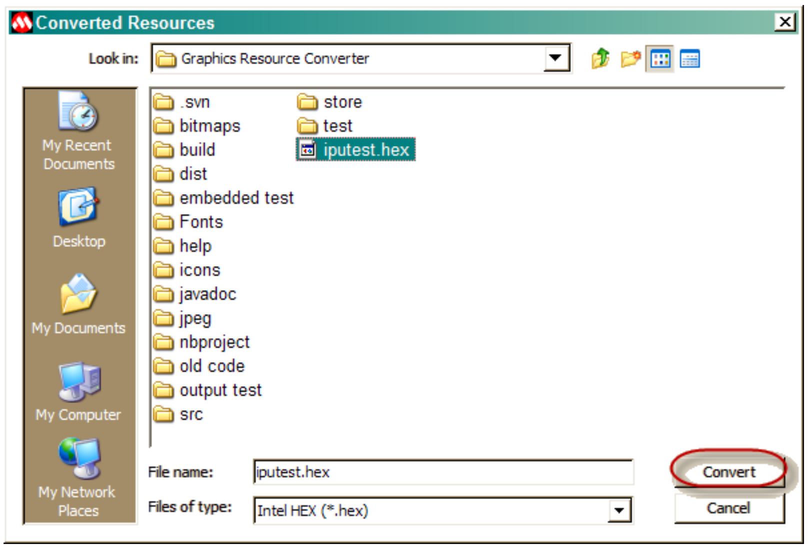


ze (Bytes)	Description
2,073	73x55 pixels, 4-bits per pixel
2,073	73x55 pixels, 4-bits per pixel
43,718	240x180 pixels, 8-bits per pixel
2,630	72x72 pixels, 4-bits per pixel

2. "Convert Resources" dialog will appear.



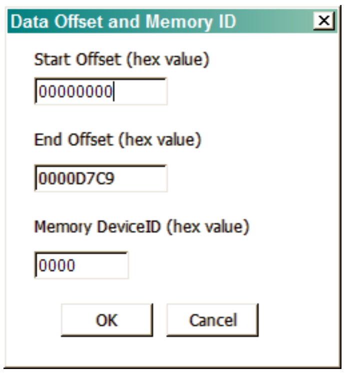
3. Press **Convert** button.



File name: iputest.hex

Files of type: Intel HEX (*.hex)

4. “Data offset and Memory ID” dialog will appear.

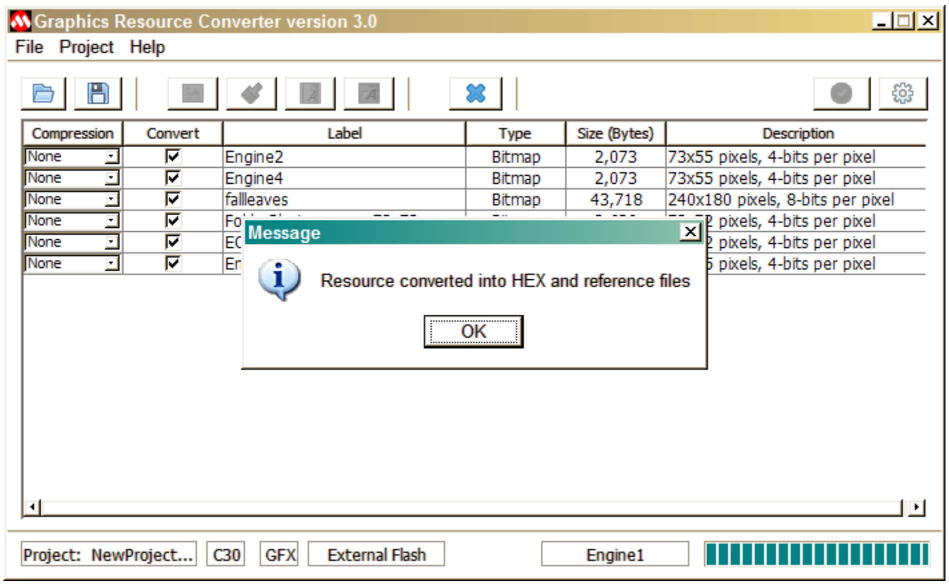


Start Offset (hex value): 00000000

End Offset (hex value): 0000D7C9

Memory DeviceID (hex value): 0000

7. Enter start address if the converting data must have special location in the external memory.
8. Enter memory ID. Memory ID is a unique number assigned by application for the memory chip where data will be stored. This number allows distinguishing memory chip if the application has several of them.
9. Press **OK**. Conversion Complete message will appear when files are created successfully.


Graphics Resource Converter version 3.0

Compression	Convert	Label	Type	Size (Bytes)	Description
None	✓	Engine2	Bitmap	2,073	73x55 pixels, 4-bits per pixel
None	✓	Engine4	Bitmap	2,073	73x55 pixels, 4-bits per pixel
None	✓	fallleaves	Bitmap	43,718	240x180 pixels, 8-bits per pixel
None	✓	Fo			2 pixels, 4-bits per pixel
None	✓	EC			2 pixels, 4-bits per pixel
None	✓	Er			5 pixels, 4-bits per pixel

Message

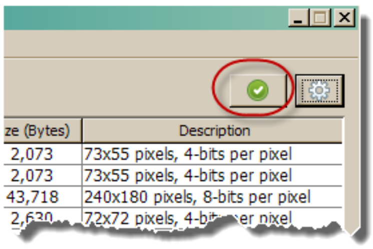
Resource converted into HEX and reference files

Project: NewProject... C30 GFX External Flash Engine1

4.8.3 Converting into Binary file

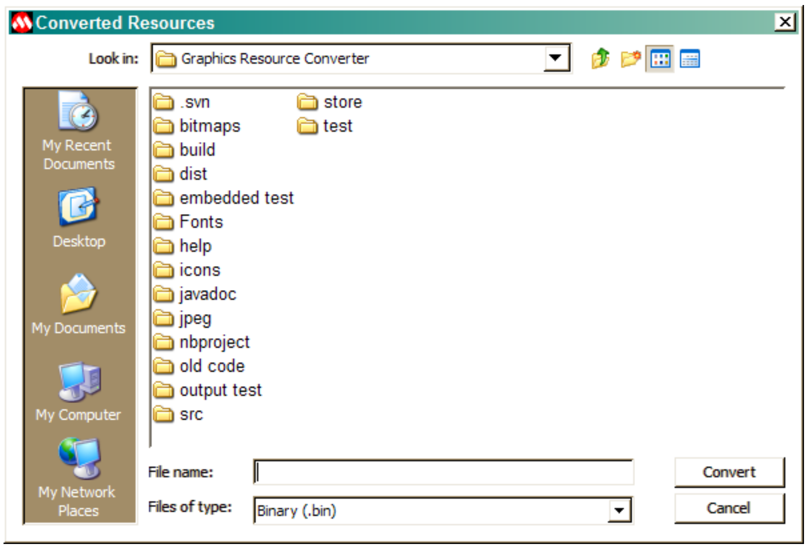
This output format allows creating binary files for objects in the conversion list. Each bitmap or font image will be stored in a separate file. The following steps should be done:

1. Press **Convert** button.



ze (Bytes)	Description
2,073	73x55 pixels, 4-bits per pixel
2,073	73x55 pixels, 4-bits per pixel
43,718	240x180 pixels, 8-bits per pixel
2,630	72x72 pixels, 4-bits per pixel

2. “Converted Resources” dialog will appear.


Converted Resources

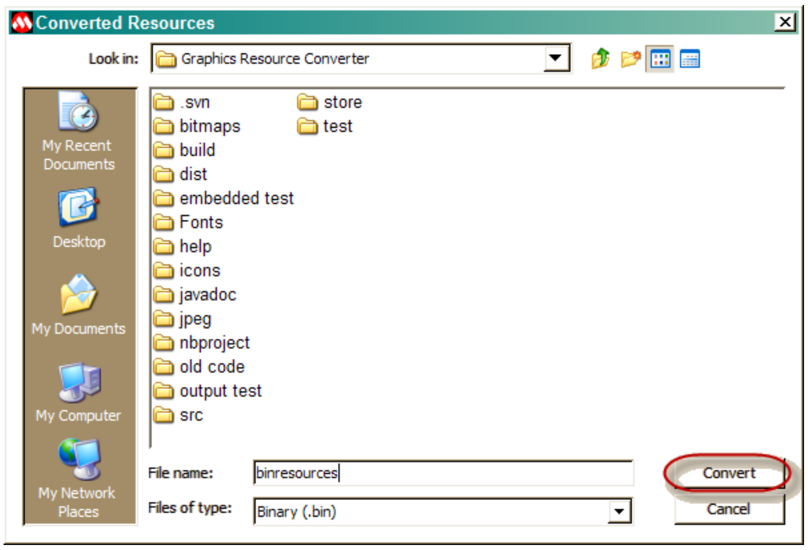
Look in: Graphics Resource Converter

.svn, bitmaps, build, dist, embedded test, Fonts, help, icons, javadoc, jpeg, nbproject, old code, output test, src, store, test

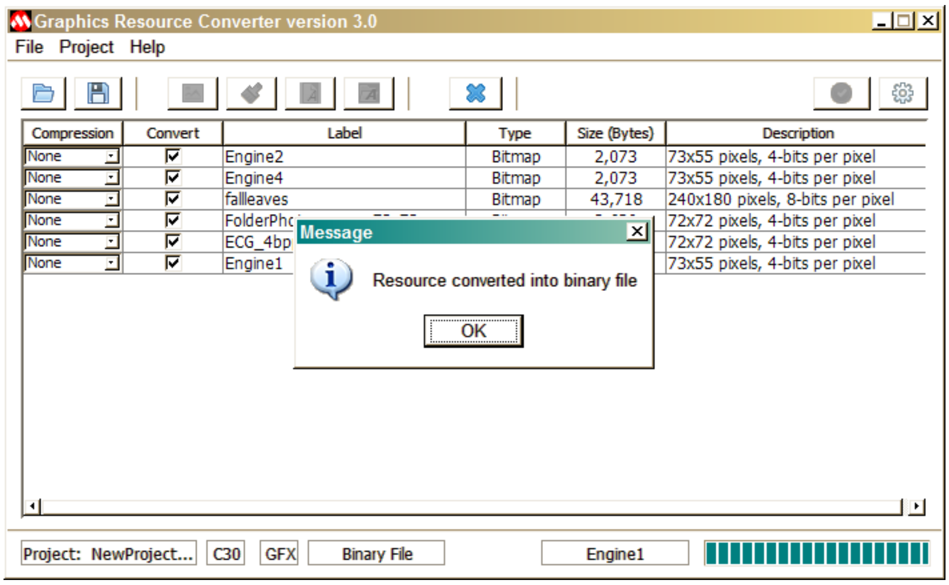
File name:

Files of type: Binary (.bin)

3. Press **Convert** button.



4. When the resources have been converted, a dialog will appear indicating such.



Compression	Convert	Label	Type	Size (Bytes)	Description
None	✓	Engine2	Bitmap	2,073	73x55 pixels, 4-bits per pixel
None	✓	Engine4	Bitmap	2,073	73x55 pixels, 4-bits per pixel
None	✓	fallleaves	Bitmap	43,718	240x180 pixels, 8-bits per pixel
None	✓	FolderPhc			72x72 pixels, 4-bits per pixel
None	✓	ECG_4bp			72x72 pixels, 4-bits per pixel
None	✓	Engine1			73x55 pixels, 4-bits per pixel

Resource converted into binary file

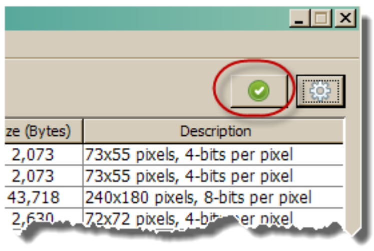
Project: NewProject... C30 GFX Binary File Engine1

4.8.4 Conversion into Intel HEX in EDS Space

This output format should be selected to store bitmaps and fonts in external memory. Address of each object will be aligned by 4 bytes border. In addition to Intel hex file the utility will generate C file containing structures FONT_EXTERNAL and GFX_IMAGE_HEADER. These structures must be used in application to access converted pictures and fonts in external memory. Name of the reference C file will be the same as name of the hex file with ".c" appended. This file will be compiled with the application to enable an easy reference to the strings that will be used in the application. Please refer to Font Reference File Output (see page 34) description. The device that the resources are being converted to must have EDS space access through EPMP. Please check the device datasheet for this information.

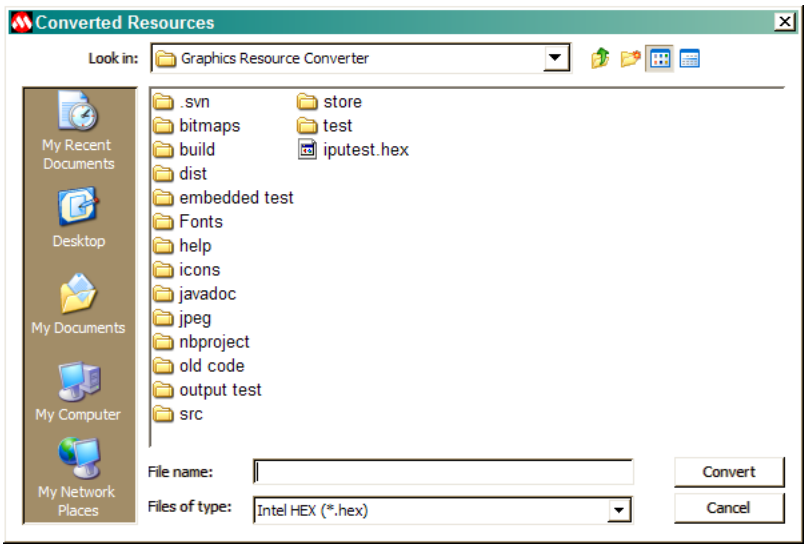
To create Intel Hex and reference C files following steps must be performed:

1. Press **Convert** button.

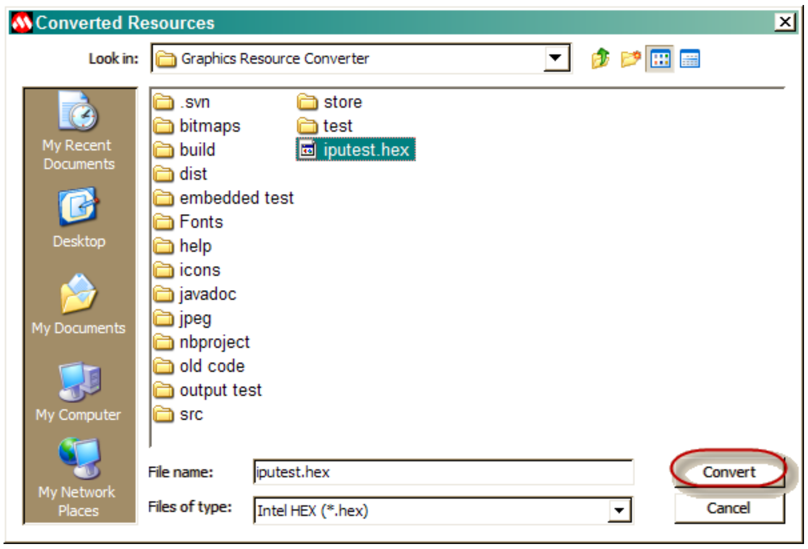


ze (Bytes)	Description
2,073	73x55 pixels, 4-bits per pixel
2,073	73x55 pixels, 4-bits per pixel
43,718	240x180 pixels, 8-bits per pixel
2,630	72x72 pixels, 4-bits per pixel

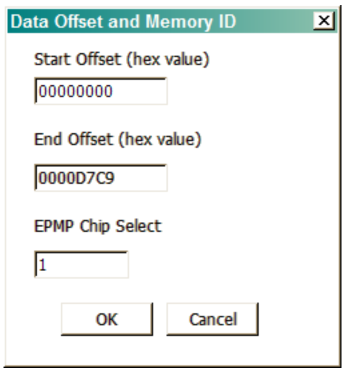
2. “Convert Resources” dialog will appear.



3. Press **Convert** button.



4. “Data offset and Memory ID” dialog will appear.



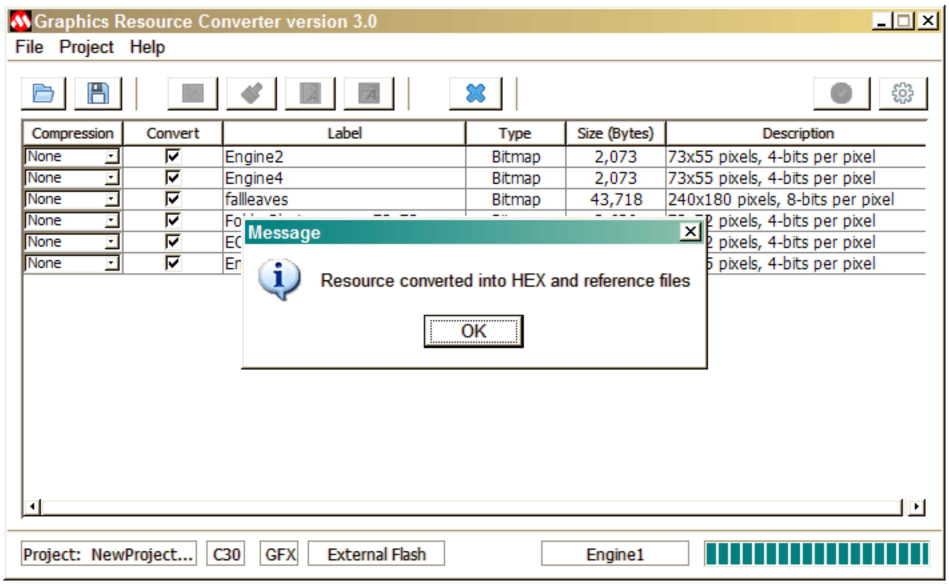
Data Offset and Memory ID

Start Offset (hex value): 00000000

End Offset (hex value): 0000D7C9

EPMP Chip Select: 1

7. Enter start address if the converting data must have special location in the external memory.
8. Enter EPMP Chip Select. The EPMP Chip Select is the value of the chip select being used. Zero is not a valid number for the chip select.
9. Press **OK**. Conversion Complete message will appear when files are created successfully.



Graphics Resource Converter version 3.0

Compression	Convert	Label	Type	Size (Bytes)	Description
None	✓	Engine2	Bitmap	2,073	73x55 pixels, 4-bits per pixel
None	✓	Engine4	Bitmap	2,073	73x55 pixels, 4-bits per pixel
None	✓	fallleaves	Bitmap	43,718	240x180 pixels, 8-bits per pixel
None	✓	Fo			pixels, 4-bits per pixel
None	✓	EC			pixels, 4-bits per pixel
None	✓	Er			pixels, 4-bits per pixel

Message

Resource converted into HEX and reference files

Project: NewProject... C30 GFX External Flash Engine1

5 Input and Output File Formats

5.1 Font Filter File Format

The font filter file is a text file created in a text editor capable of handling Unicode fonts and saving text files in 16-bit Unicode format. The format of the filter file is shown below:

```
ButtonStr: ??? // Buttons
CheckBoxStr: ???????? // Checkbox
RadioButtonStr: ?????? //Radio buttons
GroupBoxStr: ???????? //GroupBox
StaticTextStr: ?????? //StaticText
SliderStr: ????? //Slider
ProgressBarStr: ??????? //Progress bar
ListBoxStr: ??????? //List box
EditBoxStr: ?????? //Edit box
MeterStr: ???? //Meter
DialStr: ???? //Dial
PictureStr: ?? //Picture
StaticTextLstStr: ????????
                    ??????
                    ?????
                    ?????????
                    ????????
                    ?????      //Microchip Graphics Library Static Text and Group Box Test.
include:    1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ    //
include:    abcdefghijklmnopqrstuvwxyz              //
include:    "!#$%&'()*+`-.,/:;<=>?@[\]^_"      //dummy string to include the standard ASCII
character set numbers and alphabet.
```

Each line is divided into three sections:

String Label Section	String Section	Comment Section
ButtonStr:	???	// Buttons

1. The string label section - This is the same string label that will be used in the C reference file that will define the character array created for the string it describes. "include" is a special string label that signifies the characters in the string section will be included in the font table but the generated C reference file will not include the string. Note that the character IDs may change so to maintain the character ID of the ASCII characters the whole range of characters from 32 to 127 must be explictly included (as shown in the example above).
2. The string section - The source of the character ID filter to generate the reduced font table.
3. the comment section - This is an optional comment section that users may want to add to the string. The comments are optional but the "//" comment indicator is required.

The string label section should be characters using the ASCII codes with identifier names complying with standard C format. This is a requirement since the compiler will not be able to generate code when variable names are not using the standard C formats. The string section will be encoded into an array of 2 byte character ID that the utility will generate. Each line should be terminated by a newline character.

Spaces are counted as characters in the string. Except for the new line character ("/n" or 0x000A), tabs and other control characters are ignored. An example is shown in the "StaticTextLstStr" shown above.

An example of an editor that can be used is the **Word Pad**. Another good editor is the **BabelPad Version 1.9.3**. This is an editor tool available at http://www.babelstone.co.uk/Software/BabelPad.html.

5

5.2 Font Reference File Output

The font reference file is created to help in the usage of the filtered font table and referencing strings in the application. An example of the output of the font reference file is shown below:

```
XCHAR ButtonStr[] = {0x00B2, 0x00A6, 0x00BD, 0x0000}; // Buttons
XCHAR CheckBoxStr[] = {0x00A8, 0x009A, 0x00A9, 0x009E, 0x00B2, 0x00A9, 0x009E, 0x00A5,
0x0000}; // Checkbox
XCHAR RadioButtonStr[] = {0x00B8, 0x00A4, 0x009B, 0x00B2, 0x00A6, 0x00BD, 0x0000}; //Radio
buttons
XCHAR GroupBoxStr[] = {0x009F, 0x00BA, 0x00BE, 0x00B0, 0x00B2, 0x00A9, 0x009E, 0x00A5,
0x0000}; //GroupBox
XCHAR StaticTextStr[] = {0x00EC, 0x00DB, 0x00AA, 0x009D, 0x00A5, 0x00AB, 0x0000};
//StaticText
XCHAR SliderStr[] = {0x00A5, 0x00B8, 0x0099, 0x00A7, 0x00BE, 0x0000}; //Slider
XCHAR ProgressBarStr[] = {0x00B0, 0x00BC, 0x009F, 0x00BB, 0x00A5, 0x00AD, 0x00BE, 0x0000};
//Progress bar
XCHAR ListBoxStr[] = {0x00B9, 0x00A5, 0x00AB, 0x00B2, 0x00A9, 0x009E, 0x00A5, 0x0000};
//List box
XCHAR EditBoxStr[] = {0x00E3, 0x00EB, 0x00B2, 0x00A9, 0x009E, 0x00A5, 0x0000}; //Edit box
XCHAR MeterStr[] = {0x00B5, 0x00BE, 0x00A6, 0x00BE, 0x0000}; //Meter
XCHAR DialStr[] = {0x00A7, 0x0099, 0x00B6, 0x00BA, 0x0000}; //Dial
XCHAR PictureStr[] = {0x00DE, 0x00C7, 0x0000}; //Picture
XCHAR StaticTextLstStr[] = {0x00B3, 0x0099, 0x009E, 0x00BC, 0x00A8, 0x00A9, 0x00B0, 0x0090,
0x000A,
                            0x009F, 0x00B8, 0x00AE, 0x0098, 0x00A9, 0x009E, 0x000A,
                            0x00B8, 0x0099, 0x00AF, 0x00B8, 0x00B9, 0x000A,
                            0x00EC, 0x00DB, 0x00AA, 0x009D, 0x00A5, 0x00AB, 0x0087, 0x0093,
0x0092, 0x000A,
                            0x009F, 0x00BA, 0x00BE, 0x00B0, 0x00B2, 0x00A9, 0x009E, 0x00A5,
0x000A,
                            0x0090, 0x00AA, 0x00A5, 0x00AB, 0x0083, 0x0000};    //Microchip
Graphics Library Static Text and Group Box Test.
```

The character IDs listed in the arrays are not the same as the original IDs. This is due to the fact that the utility will be generating a font table with contiguous and no unused character ID. Unused character IDs in the original font table are removed resulting in memory saving.

5.3 External Memory Reference Output

External memory reference file output example is shown below:

```
#include "Graphics.h"

FONT_EXTERNAL Mona_1 = {0x0001,0x0000,0x00000000};
FONT_EXTERNAL Mona = {0x0001,0x0000,0x00002280};
BITMAP_EXTERNAL myLogo = {0x0001,0x0000,0x000052FC};
```

In this example, two Mona font of different sizes was generated.

5

6 Output Image and Font Data Formats

6.1 Output Bitmap Image Format

Image Structure for Bitmap

Block	Name	Bits	Description
Header	Reserved	8	Reserved
	BPP	8	Bits per pixel in the bitmap
	Height	16	Image height in pixels
	Width	16	Image width in pixels
Color Table	First color value	16	
	...	...	...
	Last color value	16	
Raster Data	...	...	...

Color Table Entry Format

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Value	R	R	R	R	R	G	G	G	G	G	G	B	B	B	B	B	

Raster Data Encoding

Pixels are stored left-to-right, up-to-bottom. Color indices are zero based, meaning a pixel color of 0 represents the first color table entry, a pixel color of 255 (if there are that many) represents the 256th entry. For images with more than 256 colors there is **NO** color table.

Raster Data Encoding for 1bit/black & white images

Every byte holds 8 pixels, its highest order bit representing the leftmost pixel. There are 2 color table entries.

Raster Data Encoding for 4bit/16 color images

Every byte holds 2 pixels, the least significant 4 bits represents the leftmost pixel. There are 16 color table entries.

Raster Data Encoding for 8bit/256 color images

Every byte holds 1 pixel. There are 256 color table entries.

Raster Data encoding for hicolor images

Hicolor images will be down sized to 16bpp. Every 2bytes / 16bit holds 1 pixel. The pixels are no color table pointers. There are no color table entries. Color coded in format in the Color Table Entry Format shown above.

IPU Compression

After the bitmap image has been converted to the Microchip Graphics Library bitmap image, the IPU compression is preformed. The IPU compression is offered on a Microchip devices that have IPU engine.

6.2 Font Image Format

Image Structure for Fonts

Block	Char	Name	Bits	Description
Font Header		Font ID	8	User-assigned ID number
		Reserve	4	Reserved for future use (must be set to 0)
		orientation	2	Orientation of the character glyphs (0,90,180,270 degrees)
		Reserve	2	Reserved for future use (must be set to 0)
		First Char	16	Character code of first character in font (e.g. 32)
		Last Char	16	Character code of last character in font (e.g. 3006)
		Height	8	Character height in pixels
		Reserve	8	Reserved for future use (must be set to 0)
Character Table	1st Char	Width	8	Width in pixels
		Offset LSB	8	Offset from font table start (8 LSBs)
		Offset MSB	16	Offset from font table start (16 MSBs)
	2nd Char	Width	8	Width in pixels
		Offset LSB	8	Offset from font table start (8 LSBs)
		Offset MSB	16	Offset from font table start (16 MSBs)
	...	...	...	...
	...	...	...	...
	...	...	...	...
	Last Char	Width	8	Width in pixels
		Offset LSB	8	Offset from font table start (8 LSBs)
		Offset MSB	16	Offset from font table start (16 MSBs)
Font Bitmap	1st Char		Char width dependent	Bitmap data of character
	2nd Char		Char width dependent	Bitmap data of character
	...	...	...	...
	Last Char		Char width dependent	Bitmap data of character

Font Header

All characters have the same height. FirstChar and LastChar define the first and last characters in the font image.

Character Table

This table is an array of entries each consisting of two 2-byte WORDs. The first byte of each entry is the character width.

The second BYTE and second WORD of each entry is the byte 24 bits offset from the beginning of the image to the character bitmap. The number of entries in the table is calculated as ((LastChar - FirstChar) + 1.

Font Bitmap

This field contains the character bitmap definitions. Each character is stored as a contiguous set of bytes. Each bit represents one pixel. Pixels are stored left-to-right, up-to-bottom.

7 Examples

7.1 Generating Reduced Font Tables

When using a font filter, the generated font table will only include characters in the strings section of the filter file (see Font Filter File Format (see page 33) for details). To maintain the character ID's of the standard ASCII character table use the special string label "include" and include all the characters in the string from character ID 32 to 127.

```
include:    " !#$%&'()*+`-.,/:;<=>?@[\]^_{|}~"    //
include:    1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ  //
include:    abcdefghijklmnopqrstuvwxyz // the standard ASCII character set from ID's 32 -
127
```

This will generate the font table with the ASCII characters with ID from 32 to 127. Doing this enables the user to refer to each character in the application code in a normal fashion as shown in the example code below:

```
static char StringName[] = "Hello World";
```

However, if the characters included are not the complete set of the 32 to 127 character IDs, the generated font table may change the character ID's of some or all the characters. For example:

```
include:    1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ    //
include:    abcdefghijklmnopqrstuvwxyz // not the complete standard ASCII character set
from ID's 32 - 127
```

The generated font table will assign a character ID of 33 to '0' (character zero) and not 48 as seen from the ASCII table. The reason is that the range of characters from '!' to '/' was not included. The utility will remove the unused characters and move the next character to the location of the first removed character. The scheme will be performed on all characters thus the generated font table will have a completely new character IDs. When this happens, the code above will not work since the C30 or C32 compiler assumes that the string "Hello World" will use the standard character IDs of all the characters. The code must be modified to this form:

```
XCHAR HelloStr[] = {0x0032,0x0049,0x0050,0x0050,0x0053,0x0020,
                    0x0041,0x0053,0x0056,0x0050,0x0048, 0x0000}; //"Hello World";
```

If no other characters in the ASCII set will be used other than the characters that comprises the "Hello World" string it will be best to use a string label to define the string "Hello World" in the font filter file.

```
HelloWorldStr:    Hello World      // generate only these characters in the font table
```

The generated reference file (see page 34) will contain this declaration:

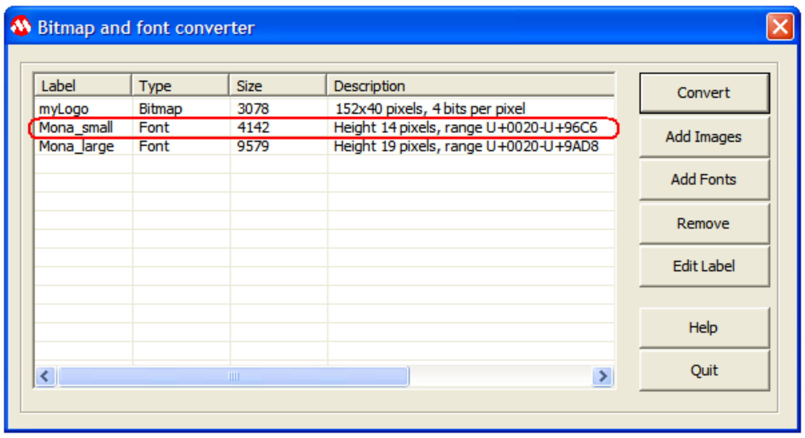
```
XCHAR HelloStr[] = {0x002B,0x0034,0x0037,0x0037,0x0038,0x0020,
                    0x0030,0x0038,0x003A,0x0037,0x0033,0x0000}; // using reduced font table
```

This method frees the user from manually calculating the converted character ID's.

Notice the 0x0020 is the space character. This is the only character that will maintain the character ID since by default the utility always start from the space character. Control characters are omitted from the generated table.

7.2 Generating Bitmaps and Multiple Font Tables

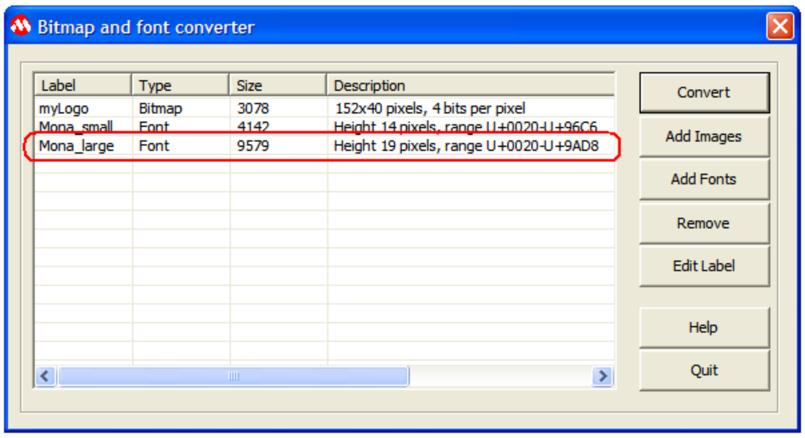
In applications that uses multiple font types and sizes, multiple font tables will be generated. To illustrate, the example below shows two generated font tables and a bitmap.



Label	Type	Size	Description
myLogo	Bitmap	3078	152x40 pixels, 4 bits per pixel
Mona_small	Font	4142	Height 14 pixels, range U+0020-U+96C6
Mona_large	Font	9579	Height 19 pixels, range U+0020-U+9AD8

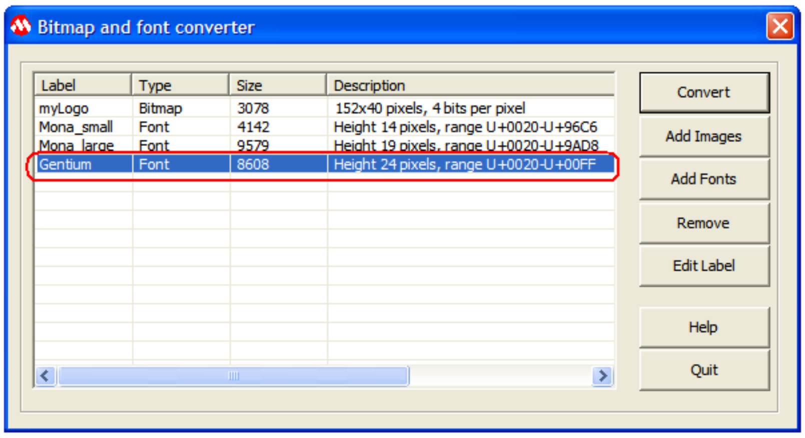
The item encircled in red used Mona font and a font filter file defining the strings and characters that will be included in the table. The font size is shown as 14 pixels. The lowest character ID is 0x0020 and the highest ID is 0x96C6. The font table size that will be generated will depend on the number of characters defined in the font filter file (please see Font Filter File Format (see page 33) for details).

The next generated font table is generated from the same installed Mona font but the size and the font filter file used is different. The first character ID is still 0x0020 but the last character ID is 0x9AD8. Simply because of the different filter file used.



Label	Type	Size	Description
myLogo	Bitmap	3078	152x40 pixels, 4 bits per pixel
Mona_small	Font	4142	Height 14 pixels, range U+0020-U+96C6
Mona_large	Font	9579	Height 19 pixels, range U+0020-U+9AD8

A third font table is added to the list and this time NO font filter was used, font table was generated from GenI102.ttf file and the default character range of 0x0020 to 0x007F was used. This font will be used for purely ASCII strings.

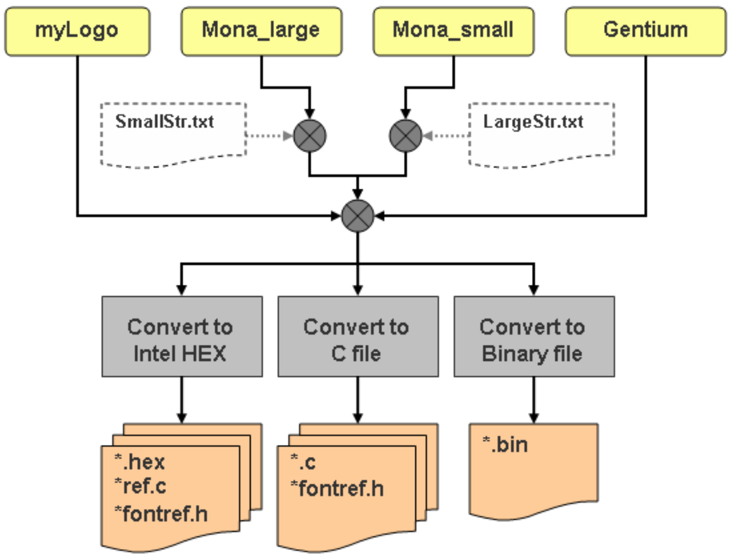


Label	Type	Size	Description
myLogo	Bitmap	3078	152x40 pixels, 4 bits per pixel
Mona_small	Font	4142	Height 14 pixels, range U+0020-U+96C6
Mona_large	Font	9579	Height 19 pixels, range U+0020-U+9AD8
Gentium	Font	8608	Height 24 pixels, range U+0020-U+00FF

The table below shows the summary of the example:

Item Type	Source	Font Size	Font Filter File	Output Structures
Bitmap	MyLogo.bmp	NA	NA	MyLogo bitmap structure
Font Table	Mona installed font	14 pixels	SmallStr.txt	Mona_small font structure
Font Table	Mona installed font	19 pixels	LargeStr.txt	Mona_large font structure
Font Table	Gentium TTF File (GenI102.ttf)	24 pixels	none	Gentium font structure

The generated output structures are dependent on the label that were assigned. In the example, the used labels are shown in the figures above. Also depending on the selected final output formats the following files will be generated by the utility:



The files *.hex, *.c and *.bin will contain the generated structures of the items in the list. The *fontref.h file will only be generated if a font filter file was added to the font items. The *ref.c file is only generated when creating HEX output. when creating the *.bin file, reference files can be created using one of the two other options. Binary file output can either reside in the external memory or internal memory so the user have the option to generate the required c or h file depending on how the binary output will be utilized in the application.

Index