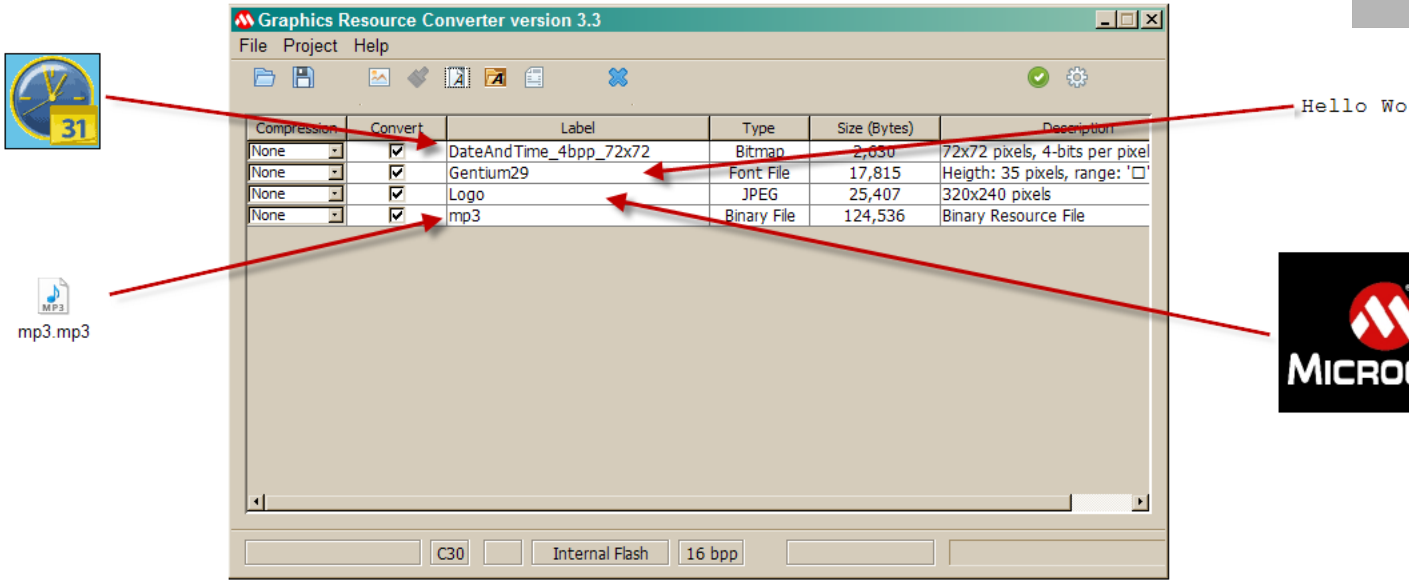
Graphics Resource Converter

Table of Contents

1 Graphics Resource Converter



Compression	Convert	Label	Type	Size (Bytes)	Description
None	✓	DateAndTime_4bpp_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel
None	✓	Gentium29	Font File	17,815	Heigth: 35 pixels, range: '□'
None	✓	Logo	JPEG	25,407	320x240 pixels
None	✓	mp3	Binary File	124,536	Binary Resource File

The Graphics Resource Converter converts images, bitmaps (BMP extension) and JPEG (JPG or JPEG extension), fonts, operating system's installed fonts or True Type fonts directly from files (TTF extension), binary files 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. The converter can also be used to create color palettes used by Microchip's graphics module based upon GIMP palettes or bitmaps.

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.

Importing binary files requires the file to have a .bin extension. Any file to convert in a raw binary format can be renamed to have the .bin extension and loaded into the converter.

Creating a palette can be done in a number of ways, GIMP palette or bitmaps. You can have the converter generate a palette based on the bitmaps that are currently loaded into the resource table. The converter will re-format the bitmaps to use the generated palette table.

2 SW License Agreement

MICROCHIP IS WILLING TO LICENSE THE ACCOMPANYING SOFTWARE AND DOCUMENTATION TO YOU ONLY ON THE CONDITION THAT YOU ACCEPT ALL OF THE FOLLOWING TERMS. TO ACCEPT THE TERMS OF THIS LICENSE, CLICK "I ACCEPT" AND PROCEED WITH THE DOWNLOAD OR INSTALL. IF YOU DO NOT ACCEPT THESE LICENSE TERMS, CLICK "I DO NOT ACCEPT," AND DO NOT DOWNLOAD OR INSTALL THIS SOFTWARE.

NON-EXCLUSIVE SOFTWARE LICENSE AGREEMENT

This Nonexclusive Software License Agreement ("Agreement") is a contract between you, your heirs, successors and assigns ("Licensee") and Microchip Technology Incorporated, a Delaware corporation, with a principal place of business at 2355 W. Chandler Blvd., Chandler, AZ 85224-6199, and its subsidiary, Microchip Technology (Barbados) II Incorporated (collectively, "Microchip") for the accompanying Microchip software including, but not limited to, Graphics Library Software, IrDA Stack Software, MCHPFSUSB Stack Software, Memory Disk Drive File System Software, mTouch(TM) Capacitive Library Software, Smart Card Library Software, TCP/IP Stack Software, MiWi(TM) DE Software, and/or any PC programs and any updates thereto (collectively, the "Software"), and accompanying documentation, including images and any other graphic resources provided by Microchip ("Documentation").

1. Definitions. As used in this Agreement, the following capitalized terms will have the meanings defined below:

a. "Microchip Products" means Microchip microcontrollers and Microchip digital signal controllers.

b. "Licensee Products" means Licensee products that use or incorporate Microchip Products.

c. "Object Code" means the Software computer programming code that is in binary form (including related documentation, if any), and error corrections, improvements, modifications, and updates.

d. "Source Code" means the Software computer programming code that may be printed out or displayed in human readable form (including related programmer comments and documentation, if any), and error corrections, improvements, modifications, and updates.

e. "Third Party" means Licensee’s agents, representatives, consultants, clients, customers, or contract manufacturers.

f. "Third Party Products" means Third Party products that use or incorporate Microchip Products.

2. Software License Grant. Microchip grants strictly to Licensee a non-exclusive, non-transferable, worldwide license to:

a. use the Software in connection with Licensee Products and/or Third Party Products;

b. if Source Code is provided, modify the Software, provided that no Open Source Components (defined in Section 5 below) are incorporated into such Software in such a way that would affect Microchip’s right to distribute the Software with the limitations set forth herein and provided that Licensee clearly notifies Third Parties regarding such modifications;

c. distribute the Software to Third Parties for use in Third Party Products, so long as such Third Party agrees to be bound by this Agreement (in writing or by "click to accept") and this Agreement accompanies such distribution;

d. sublicense to a Third Party to use the Software, so long as such Third Party agrees to be bound by this Agreement (in writing or by "click to accept");

e. with respect to the TCP/IP Stack Software, Licensee may port the ENC28J60.c, ENC28J60.h, ENCX24J600.c, and ENCX24J600.h driver source files to a non-Microchip Product used in conjunction with a Microchip ethernet controller;

f. with respect to the MiWi (TM) DE Software, Licensee may only exercise its rights when the Software is embedded on a Microchip Product and used with a Microchip radio frequency transceiver or UBEC UZ2400 radio frequency transceiver which are integrated into Licensee Products or Third Party Products.

For purposes of clarity, Licensee may NOT embed the Software on a non-Microchip Product, except as described in this Section.

3. Documentation License Grant. Microchip grants strictly to Licensee a non-exclusive, non-transferable, worldwide license to use the Documentation in support of Licensee's authorized use of the Software

4. Third Party Requirements. Licensee acknowledges that it is Licensee’s responsibility to comply with any third party license terms or requirements applicable to the use of such third party software, specifications, systems, or tools. Microchip is not responsible and will not be held responsible in any manner for Licensee’s failure to comply with such applicable terms or requirements.

5. Open Source Components. Notwithstanding the license grant in Section 1 above, Licensee further acknowledges that certain components of the Software may be covered by so-called "open source" software licenses ("Open Source Components"). Open Source Components means any software licenses approved as open source licenses by the Open Source Initiative or any substantially similar licenses, including without limitation any license that, as a condition of distribution of the software licensed under such license, requires that the distributor make the software available in source code format. To the extent required by the licenses covering Open Source Components, the terms of such license will apply in lieu of the terms of this Agreement. To the extent the terms of the licenses applicable to Open Source Components prohibit any of the restrictions in this Agreement with respect to such Open Source Components, such restrictions will not apply to such Open Source Component.

6. Licensee Obligations. Licensee will not: (i) engage in unauthorized use, modification, disclosure or distribution of Software or Documentation, or its derivatives; (ii) use all or any portion of the Software, Documentation, or its derivatives except in conjunction with Microchip Products or Third Party Products; or (iii) reverse engineer (by disassembly, decompilation or otherwise) Software or any portion thereof. Licensee may not remove or alter any Microchip copyright or other proprietary rights notice posted in any portion of the Software or Documentation. Licensee will defend, indemnify and hold Microchip and its subsidiaries harmless from and against any and all claims, costs, damages, expenses (including reasonable attorney's fees), liabilities, and losses, including without limitation product liability claims, directly or indirectly arising from or related to the use, modification, disclosure or distribution of the Software, Documentation, or any intellectual property rights related thereto; (ii) the use, sale and distribution of Licensee Products or Third Party Products; and (iii) breach of this Agreement.

7. Confidentiality. Licensee agrees that the Software (including but not limited to the Source Code, Object Code and library files) and its derivatives, Documentation and underlying inventions, algorithms, know-how and ideas relating to the Software and the Documentation are proprietary information belonging to Microchip and its licensors ("Proprietary Information"). Except as expressly and unambiguously allowed herein, Licensee will hold in confidence and not use or disclose any Proprietary Information and will similarly bind its employees and Third Party(ies) in writing. Proprietary Information will not include information that: (i) is in or enters the public domain without breach of this Agreement and through no fault of the receiving party; (ii) the receiving party was legally in possession of prior to receiving it; (iii) the receiving party can demonstrate was developed by the receiving party independently and without use of or reference to the disclosing party's Proprietary Information; or (iv) the receiving party receives from a third party without restriction on disclosure. If Licensee is required to disclose Proprietary Information by law, court order, or government agency, License will give Microchip prompt notice of such requirement in order to allow Microchip to object or limit such disclosure. Licensee agrees that the provisions of this Agreement regarding unauthorized use and nondisclosure of the Software, Documentation and related Proprietary Rights are necessary to protect the legitimate business interests of Microchip and its licensors and that monetary damage alone cannot adequately compensate Microchip or its licensors if such provisions are violated. Licensee, therefore, agrees that if Microchip alleges that Licensee or Third Party has breached or violated such provision then Microchip will have the right to injunctive relief, without the requirement for the posting of a bond, in addition to all other remedies at law or in equity.

8. Ownership of Proprietary Rights. Microchip and its licensors retain all right, title and interest in and to the Software and Documentation including, but not limited to all patent, copyright, trade secret and other intellectual property rights in the Software, Documentation, and underlying technology and all copies and derivative works thereof (by whomever produced). Licensee and Third Party use of such modifications and derivatives is limited to the license rights described in Sections this Agreement.

9. Termination of Agreement. Without prejudice to any other rights, this Agreement terminates immediately, without notice by Microchip, upon a failure by Licensee or Third Party to comply with any provision of this Agreement. Upon termination, Licensee and Third Party will immediately stop using the Software, Documentation, and derivatives thereof, and immediately destroy all such copies.

3 Release Notes

Microchip Graphics Resource Converter

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

Version 3.08.21

Bug Fixes

1. Added support for font filter files that have multiple line lists.
2. Took out the backslash ("\") file separator to make it compatible across OS platforms.
3. Correct numerous spelling errors.
4. When setting RLE for a bitmap image and that image can not be used for RLE, the image would not be a part of the resources to convert after exiting the settings dialog box.
5. When changing the settings, the resources would change in position.
6. If a font was after a bitmap image in the resource table, the font compression would match the bitmap compression, even though fonts are not able to be compressed.
7. The uncompressed size accounted from padding bytes.
8. Skip over the BOM marker of the font filter, was producing a "?" character for the first string phrase label.
9. In the Setting dialog box, when selecting the graphics module, the graphics bits per pixel could still be set.
    1. When the graphics module is selected, the graphics bits per pixel is grayed out and set to 16bpp.
10. In the HEX address dialog box, the user is able to use upper and lower case letter for the starting address.
    1. For example: ff or FF
11. Corrected the offset calculations for the external memory conversion when using compression.
12. When entering the font point size in the font options dialog box, the font preview will be updated by either pressing <enter> or <tab>.

New Features

1. Added command line support to parse a project file (xml)
2. Added a check sum (CRC) to the external HEX files. A CRC is generated and placed in the first 8 bytes of the external HEX file. The first four bytes are used to verify that the CRC data is present. These four bytes will always read 'M', 'C', 'H', 'P'. The next four bytes are a 32-bit generated CRC. The header file produced will have the 32-bit CRC value that can be matched to the one in external memory to verify that it is valid.
3. When loading a project, check to make sure that all of the files exist. If file(s) do not exist, ask the user if they would like to search to resolve the missing file.
    1. If the user does not want to resolve the file, then do not load the project.
    2. If the user resolves the file(s), ask them if they would like to save the project using the newly resolved files.
        1. Gives the option to re-name the project as well.
4. Check font filter files for a delimiter, "//". If there is not a delimiter present notify the user.

Documentation

1. Added the command line interface documentation to the help file.

3.1 Release Notes 3.03.01

Version 3.03.01

Bug Fixes

1. Drop-down combo box to choose the starting and ending character range in the font option dialog
    1. The character display has been fixed to show the correct character representation.
2. When loading projects that reference files that do not exist.
    1. The user will be given the option to delete the resource of the non-existent file and keep all valid resources or remove all resources.
    2. The project file will not be modified until the user selects to save the project.

3.2 Release Notes 3.03.00

Version 3.03

New Features

1. Multiple bits per pixel (bpp) output support. Based on the graphics controller, the GRC can output the conversion in 16 or 24 bpp. This feature does not limit input sources. For example, if the output conversion is set to 16bpp and a 24bpp btimap image is being converted, the image will be converted to a 16bpp image.
2. Conversion of bitmap images that are in 16bpp format. The GRC will convert bitmap images saved in x555 or 565 color format. The GRC will also up convert those images to 24bpp if converted as 24bpp Graphics Module.
3. The reference header file generated after converting resources will now contain a ***#define*** of the images width and height in pixels.
4. Generation of a palette from multiple bitmap images.
    1. Added reading of comments from the GIMP file.
    2. Allow the user to load a GIMP file to set comments
    3. Created a header file with the symbol names of the palette colors
5. Added Run Length Encoding (RLE) support for bitmaps that are 4bpp and 8bpp.
6. Added support for Font Filter files that are saved in a UTF-8 format.
7. The converter is able to converter raw binary data into a C array, hex file or binary file.
    1. Drag and drop support is available.
8. Added drag and drop support for project files.
9. Added command line support.

    1. Allows a user to run the converter tool from the command line interface without invoking the GUI.
10. Changed the function buttons, like project load and save, to tool bar buttons.
11. Use internal Inflate JAVA library
    1. Support multiple operating systems.

Bug Fixes

1. Switching to PSV data space from program space at the end of a resource conversion when converting using the C30 compiler type.
2. Handling of projects that have resources on two different drives (Windows only).
3. UTF-16 little and big endian font filter fix.
4. Handle blank lines in the font filter
5. When generating a hex file using images in IPU, use the IPU generated size to determine the address location.
6. Proper handling of loading and saving projects that contain FNT font files.
7. Scrolling through the list box in the font dialog using the arrow keys, the wrong information would be displayed (size for the name of the font).
8. Made the adding the correct file extension more robust when naming a conversion file
9. Corrected the font filter error where the encoding header with being placed in the symbol name..

3.3 Release Notes 3.00.00

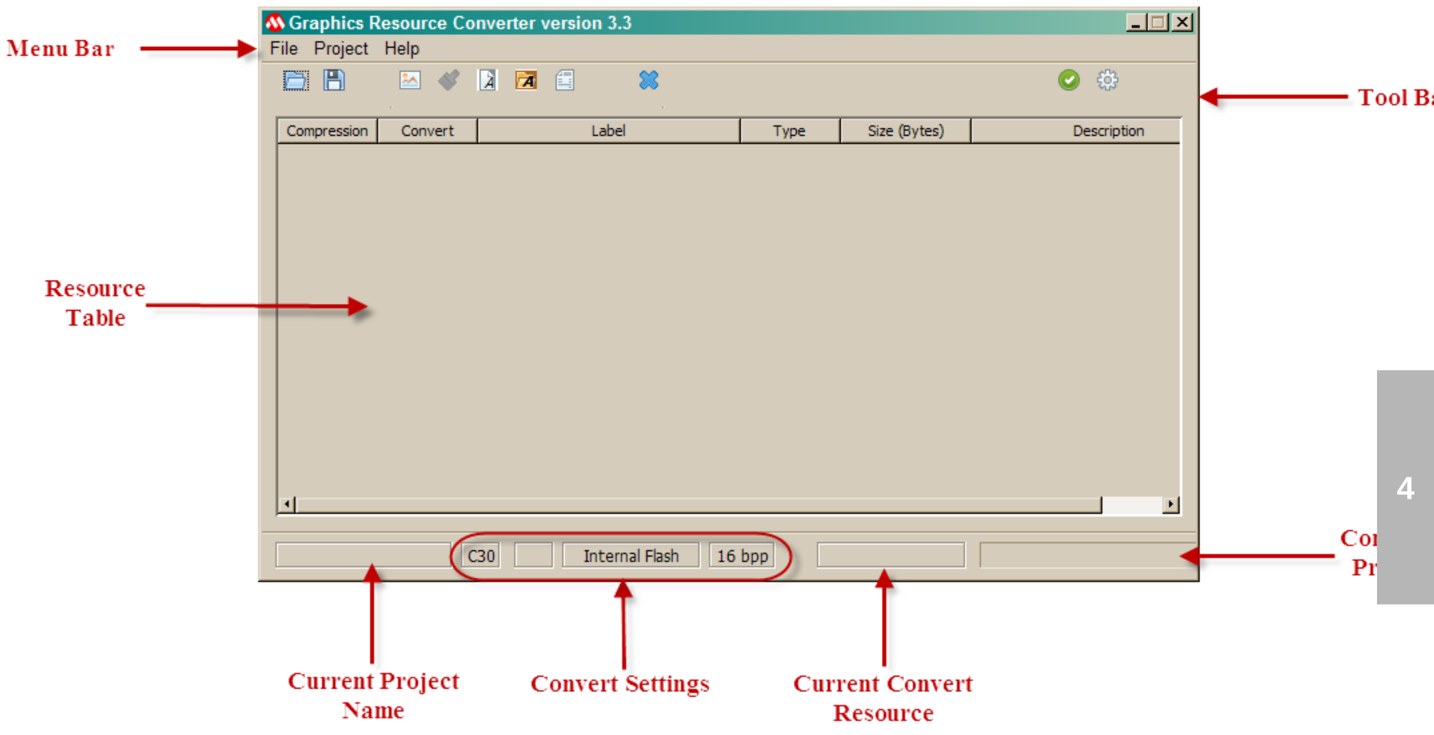
Version 3.00

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.

4 User 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.

Tool Bar Buttons

The tool bar 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 Name

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.

Current Converting Resource

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

Converting Progress

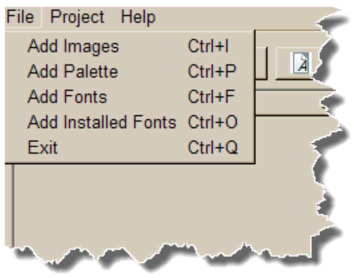
The overall progress of the when converting resources. This is based on the number of files and the relative size of the file to the overall project being converted.

4.1 Menu Bar

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.

File



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

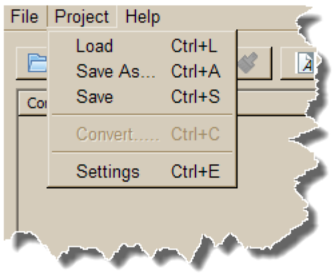
Add Palette - Loads a palette from a bitmap or GIMP file. This feature is only available for PIC microcontroller that have an internal graphics module.

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.

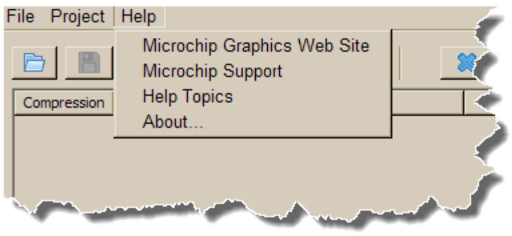
Save - Saves the current project.

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

Convert.... - Converts the project

Settings (see page 31) - Opens the settings dialog box.

Help



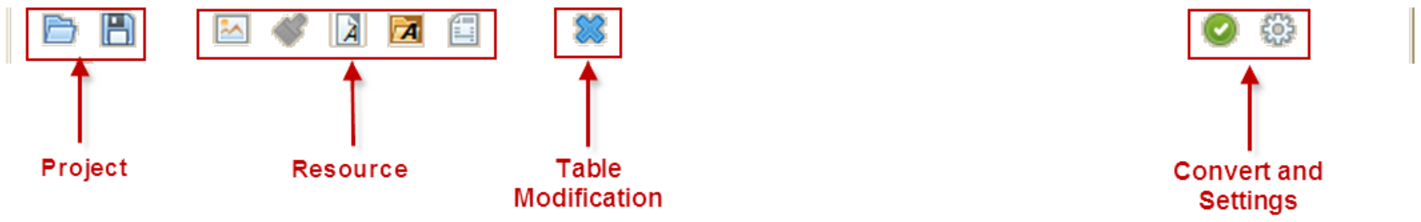
Microchip Graphics Web Site - Opens a web browser to the Microchip Graphics design center page.

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

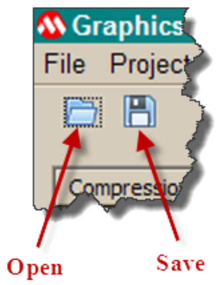
Help Topics - Opens the Help documentation for the Graphics Resource Converter.

About.. - Opens the About window.

4.2 Tool Bar Buttons



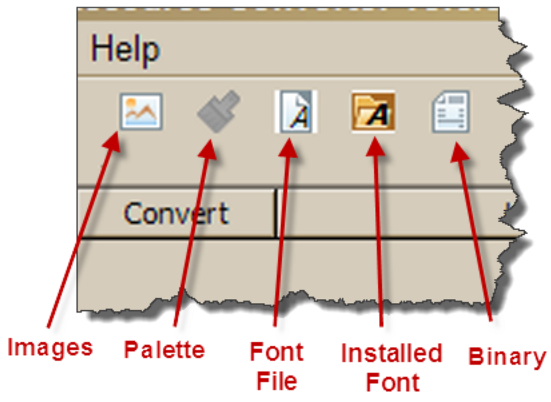
Project



Load - Loads a graphics resource project.

Save - Saves the current project.

Resource



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

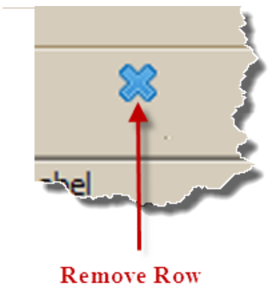
Palette - Loads a palette from a bitmap or GIMP file. This feature is only available for PIC microcontroller that have an internal graphics module.

Font File - Loads a font from a ttf or fnt font file format.

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

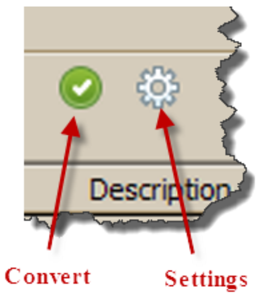
Binary - Loads a raw binary file (.bin) as a resource.

Table Modification



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

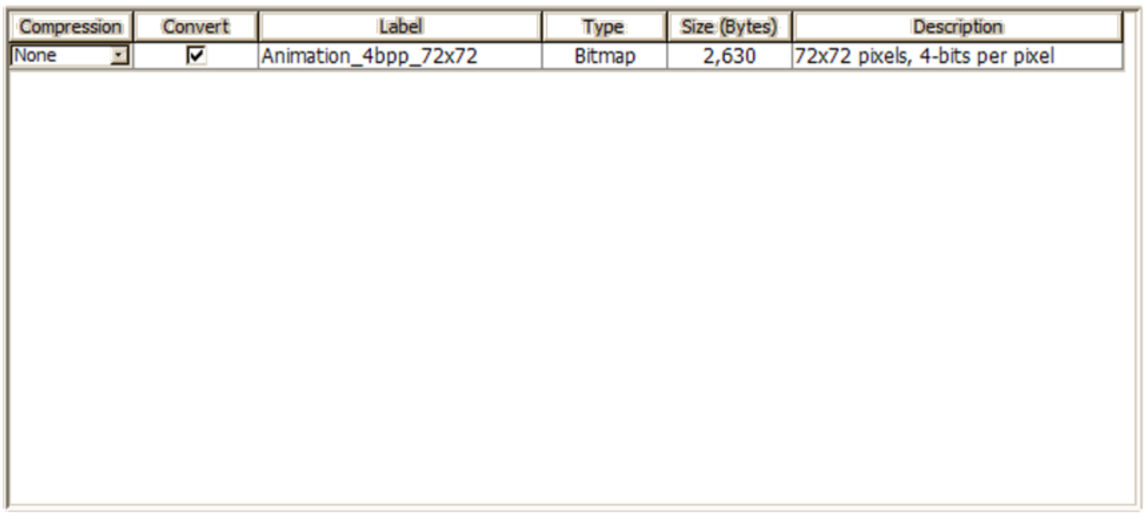
Convert and Settings



Convert - Converts the selected resources.

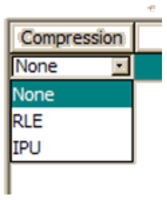
Settings (see page 31) - Sets the compiler, C30 or C32, graphics module, converted resource output and the color depth output.

4.3 Resource Table



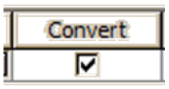
Compression	Convert	Label	Type	Size (Bytes)	Description
None	✓	Animation_4bpp_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel

Compression



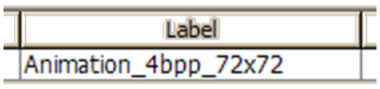
Indicates if any compression is used to reduce the size of a resource. Not all of the compression options are available for every compiler or module setting. For example, the IPU option is only available for the C30 with a graphics module.

Convert



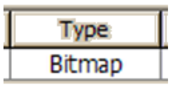
Converting resources, if this box is not checked, the resource will not be converted.

Label



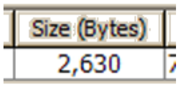
The label that will be given to the resource that the Microchip Graphics Library will reference. For example, the label, *Animation_4bpp_72x72*, will be used by the Microchip Graphics Library.

Type



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

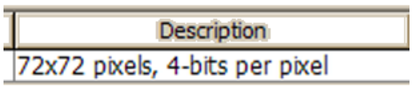
Size



Size (Bytes)
2,630

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

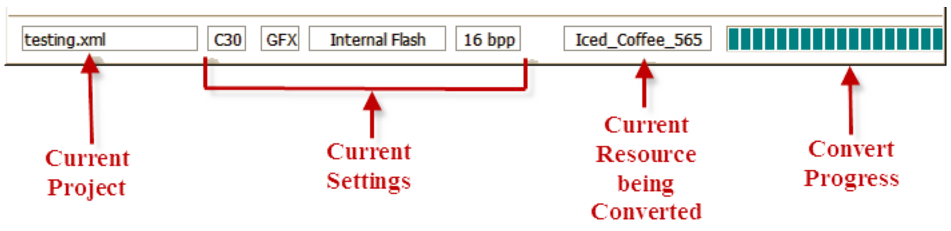
Description



Description
72x72 pixels, 4-bits per pixel

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.

4.4 Status


testing.xml | C30 | GFX | Internal Flash | 16 bpp | Iced_Coffee_565

Current Project

Shows the current project of the Graphics Resource Converter.

Current Settings

Shows the current settings of the Graphics Resource Converter, the compiler, C30 or C32, graphics module, what format the converted resources will be saved as and the display bits per pixel.

Current Resource Being Converted

An indicator showing the current resource that is being converted.

Convert Progress

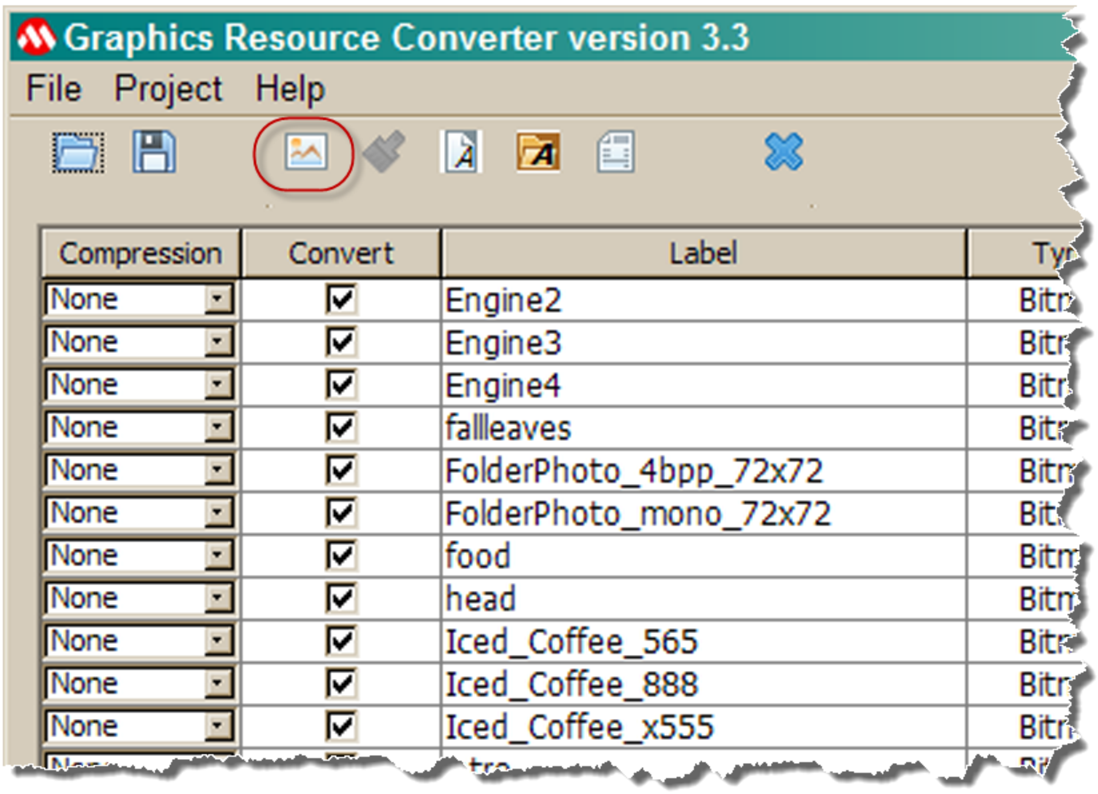
The converting resource progress measured in bytes converted to total bytes to convert.

5 Using the Utility

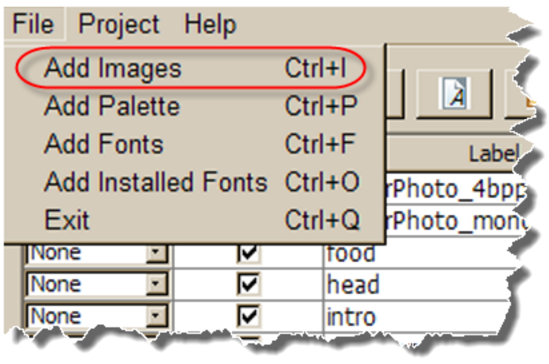
5.1 Adding Images to the Conversion List

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

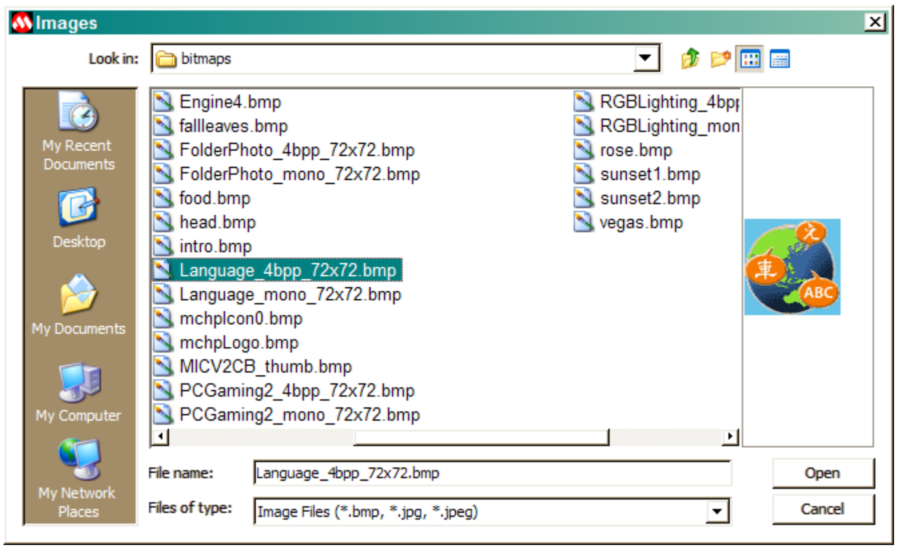
1. Press **Add Images** button or File menu item.



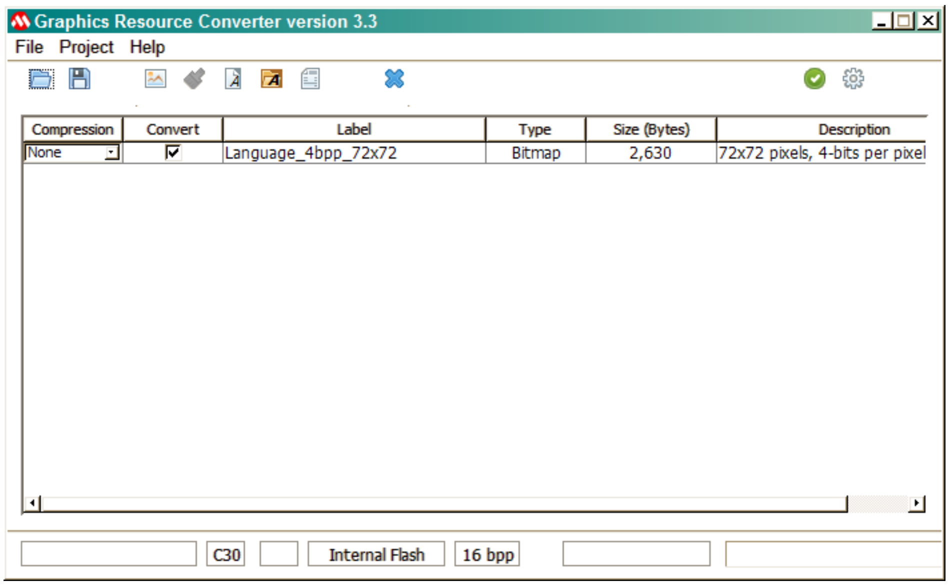
Compression	Convert	Label	Ty
None	✓	Engine2	Bitr
None	✓	Engine3	Bitr
None	✓	Engine4	Bitr
None	✓	fallleaves	Bitr
None	✓	FolderPhoto_4bpp_72x72	Bitr
None	✓	FolderPhoto_mono_72x72	Bit
None	✓	food	Bitm
None	✓	head	Bitm
None	✓	Iced_Coffee_565	Bitr
None	✓	Iced_Coffee_888	Bitr
None	✓	Iced_Coffee_x555	Bitr



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.

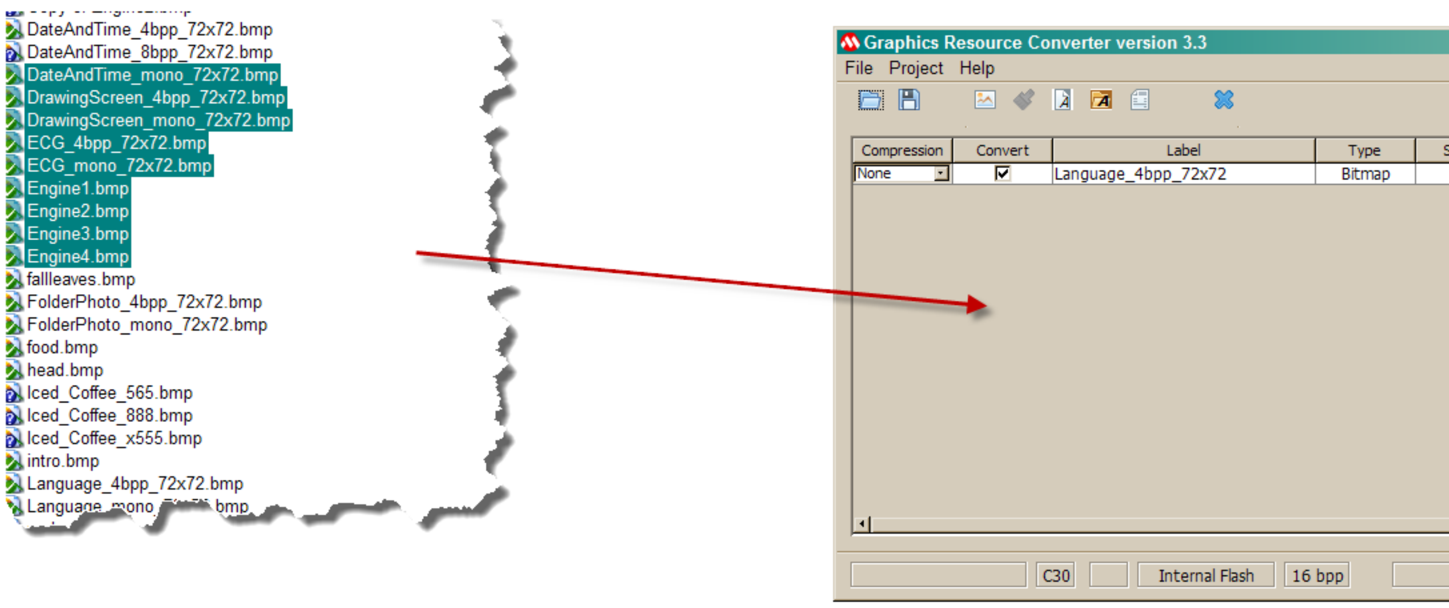


3. If selected file will be imported successfully the information about it (label, type, size and description) will be added in the list box of the main window. Label is generated from the file name, type defines that imported object is using a bitmap or a JPEG format, size of data is shown in bytes,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



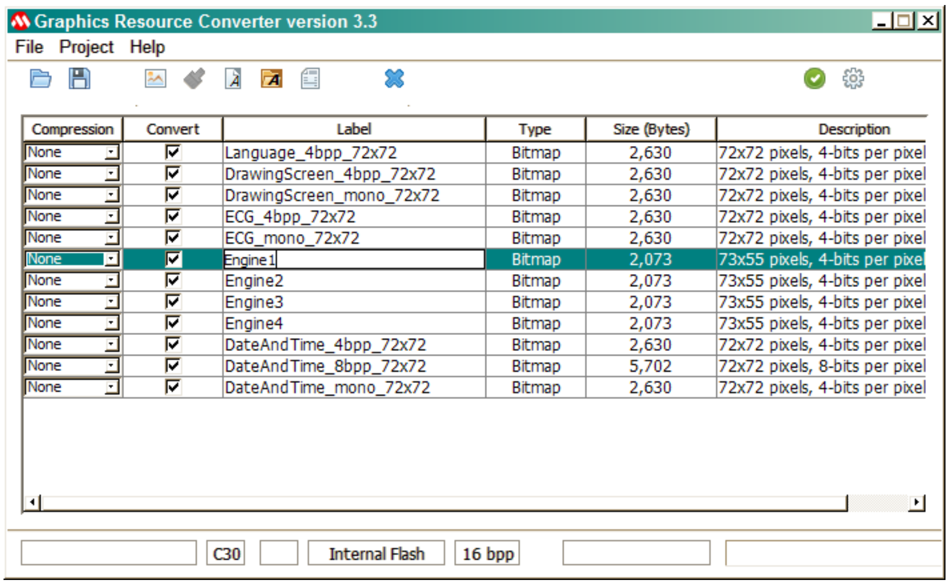
Compression	Convert	Label	Type	Size (Bytes)	Description
None	✓	Language_4bpp_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel

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


DateAndTime_4bpp_72x72.bmp
DateAndTime_8bpp_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
FolderPhoto_mono_72x72.bmp
food.bmp
head.bmp
Iced_Coffee_565.bmp
Iced_Coffee_888.bmp
Iced_Coffee_x555.bmp
intro.bmp
Language_4bpp_72x72.bmp

Compression	Convert	Label	Type
None	✓	Language_4bpp_72x72	Bitmap

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_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	✓	Engine4	Bitmap	2,073	73x55 pixels, 4-bits per pixel
None	✓	DateAndTime_4bpp_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel
None	✓	DateAndTime_8bpp_72x72	Bitmap	5,702	72x72 pixels, 8-bits per pixel
None	✓	DateAndTime_mono_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel

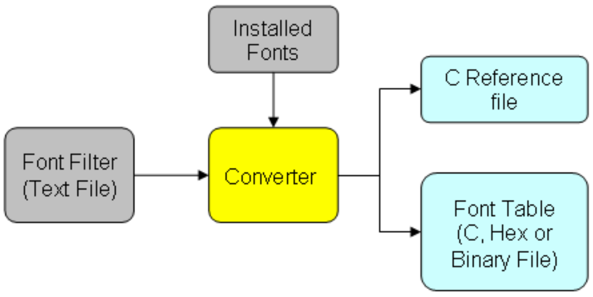
5.2 Adding Installed Fonts to the Conversion List

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.

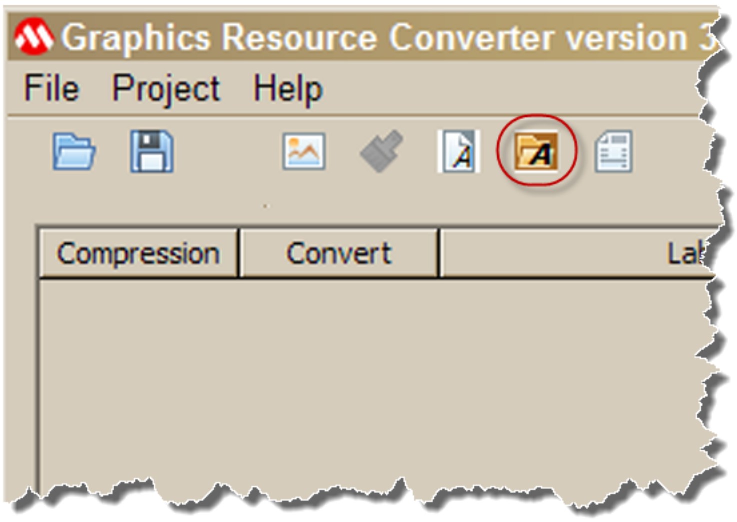
Importing Fonts using Character ID Filter

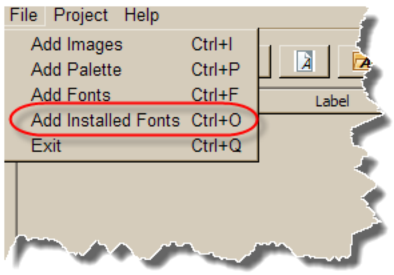
Another way to overcome the problem of memory requirements for fonts with thousands of character indexes is filtering the font table and recreating a reduced character set table which contains only the pre-defined characters. This approach will need a filter text file (described in Font Filter File Format (see page 48) 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.



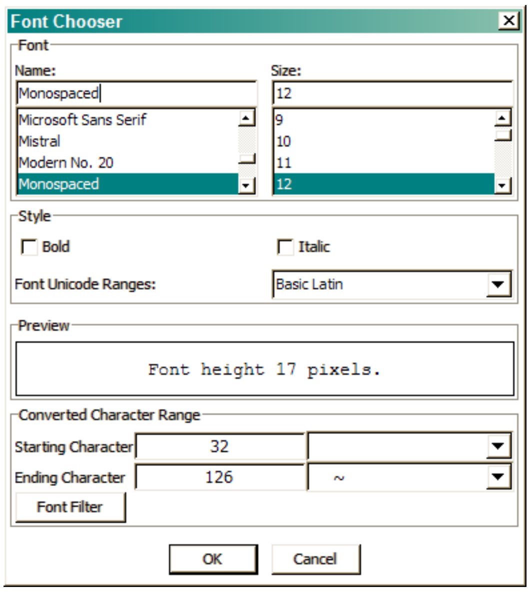
To add an installed font for conversion the following steps should be done:

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.



Name: Monospaced
Microsoft Sans Serif
Mistral
Modern No. 20
Monospaced

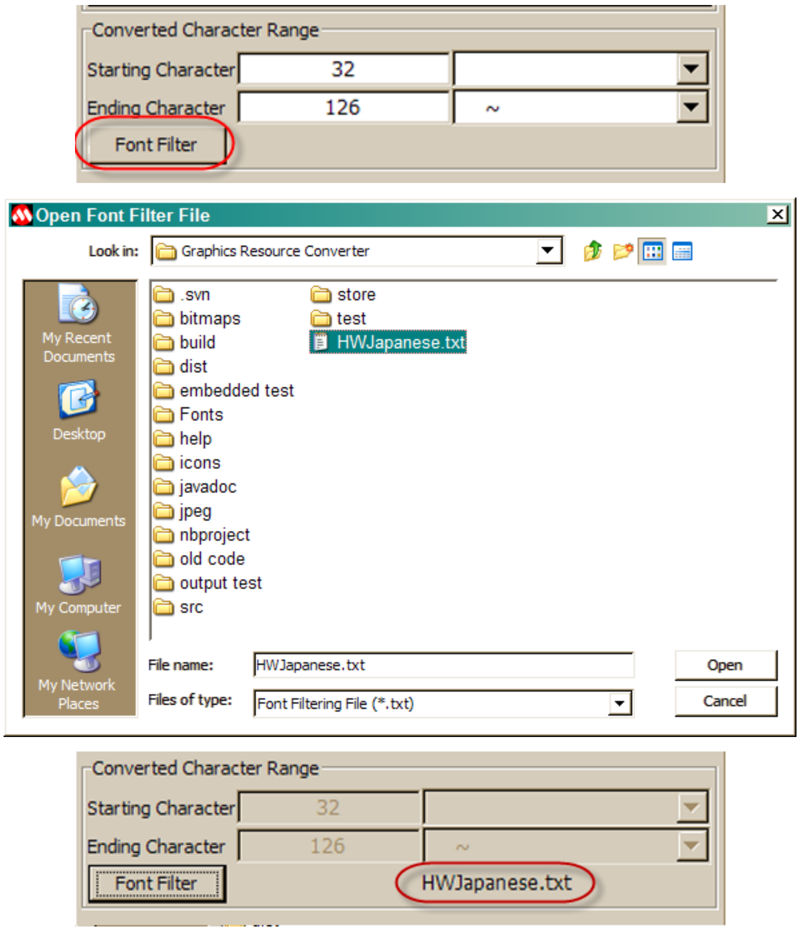
Size: 12
9
10
11
12

Font Unicode Ranges: Basic Latin

Font height 17 pixels.

Starting Character 32
Ending Character 126 ~

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.



Starting Character 32
Ending Character 126 ~

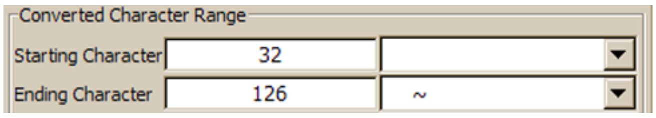
Look in: Graphics Resource Converter

.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)

Starting Character 32
Ending Character 126 ~
HWJapanese.txt

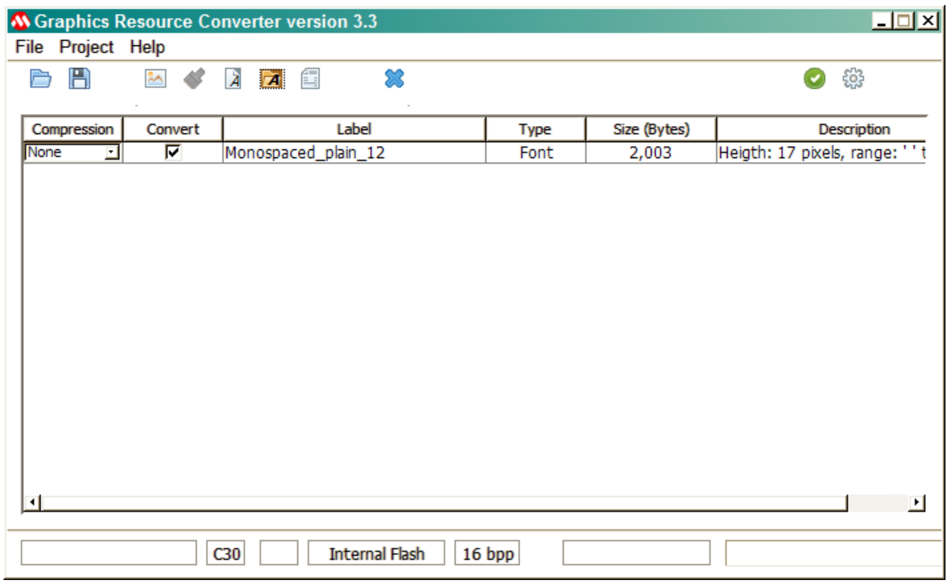
4. If filtering is not used, it is assumed by the utility that the user wants to use a range of characters instead. The user can choose a range of the characters that will be used in the font table. Starting Character sets the character index of the first character in the table. Ending Character sets 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	2,003	Heigth: 17 pixels, range: ' ' t

C30 | Internal Flash | 16 bpp

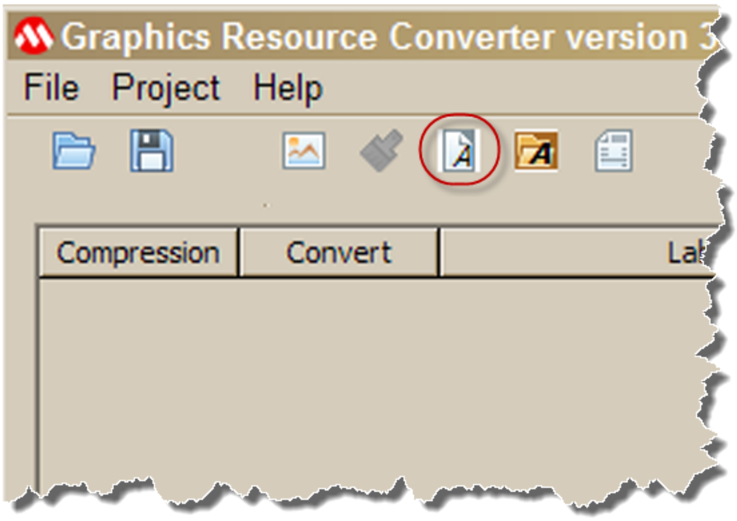
5.3 Adding Fonts from Files to the Conversion List

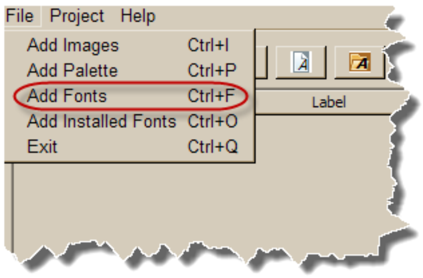
Importing Fonts from Files

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.

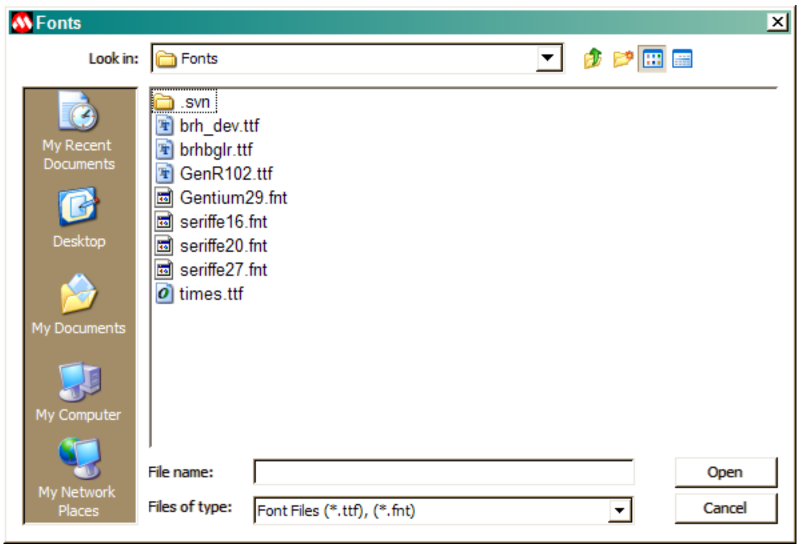
To add fonts using font files as inputs the following steps should be done:

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

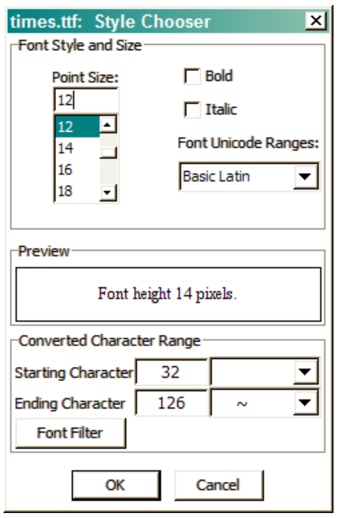




2. In the dialog select the file type extension to select importing raster font (*.fnt) or true type font (*.ttf). Browse and select the file and press **Open**.
   - Go to step 6 to generate raster fonts.
   - Go to step 3 to generate true type fonts.



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.

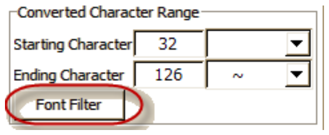


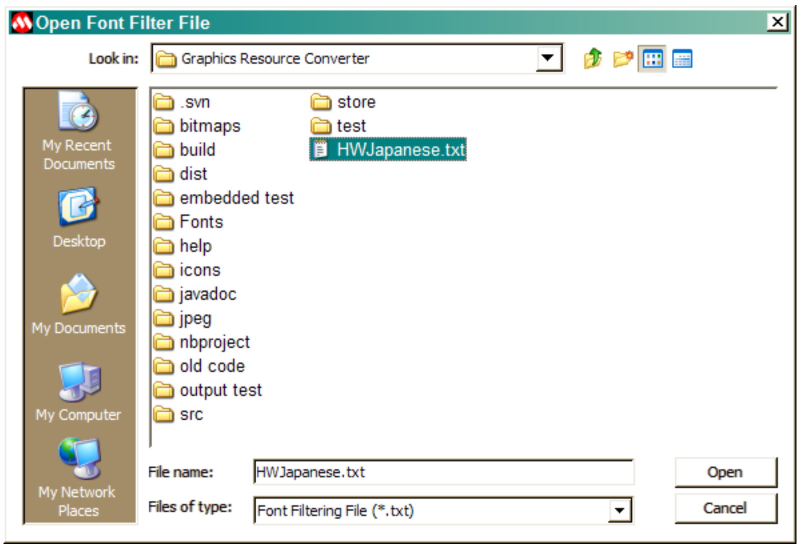
Point Size: 12

Starting Character: 32

Ending Character: 126 ~

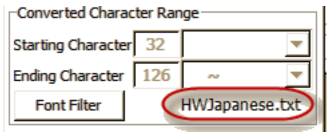
3. A font filter can be selected by pressing the "Font Filter" button. When selecting the "Font Filter" button, a file dialog will appear. This gives the user the opportunity to assign a font character filter to the selected font. This filter will be used to extract character ID's that will comprise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





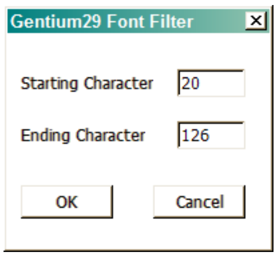
File name: HWJapanese.txt

Files of type: Font Filtering File (*.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.



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

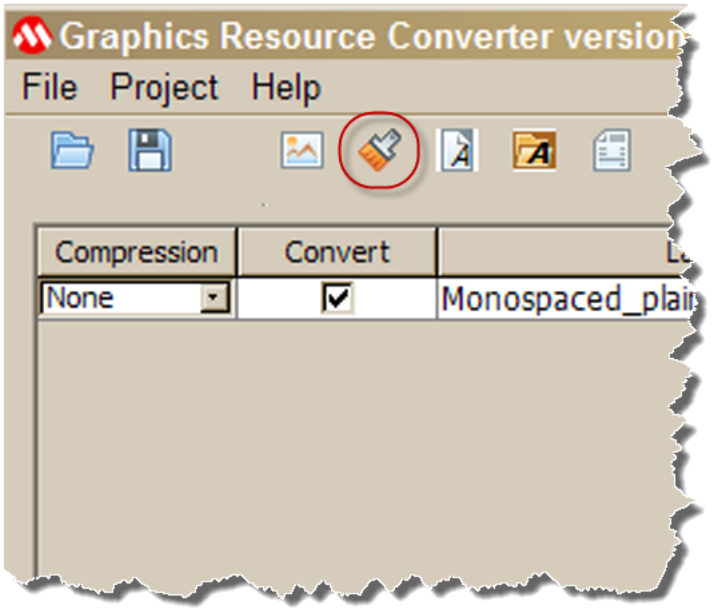
5.4 Adding Palettes to the Conversion List

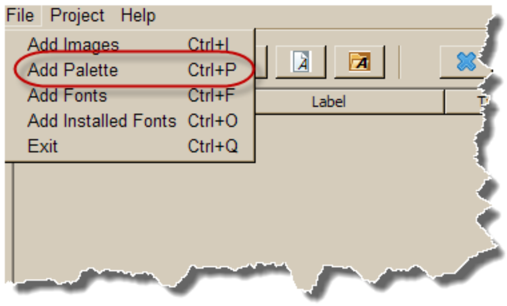
Importing Palettes

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.

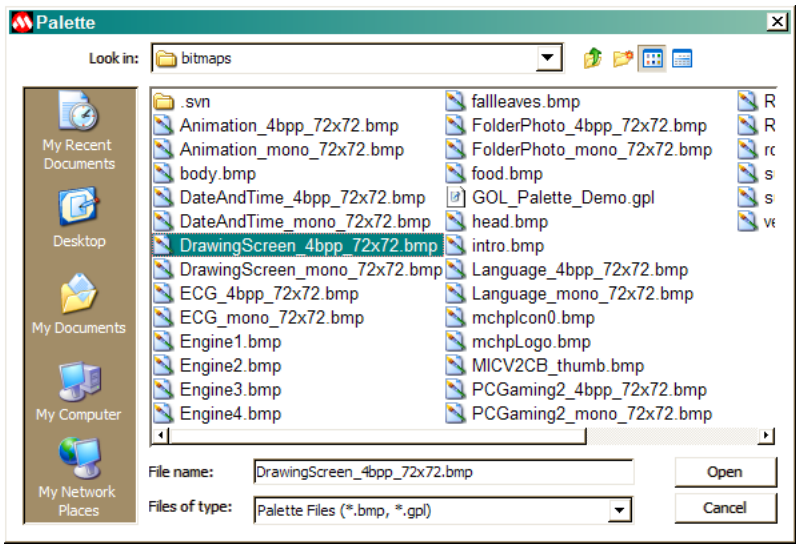
To load the color table of a bitmap file the following steps should be done:

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.


Palette

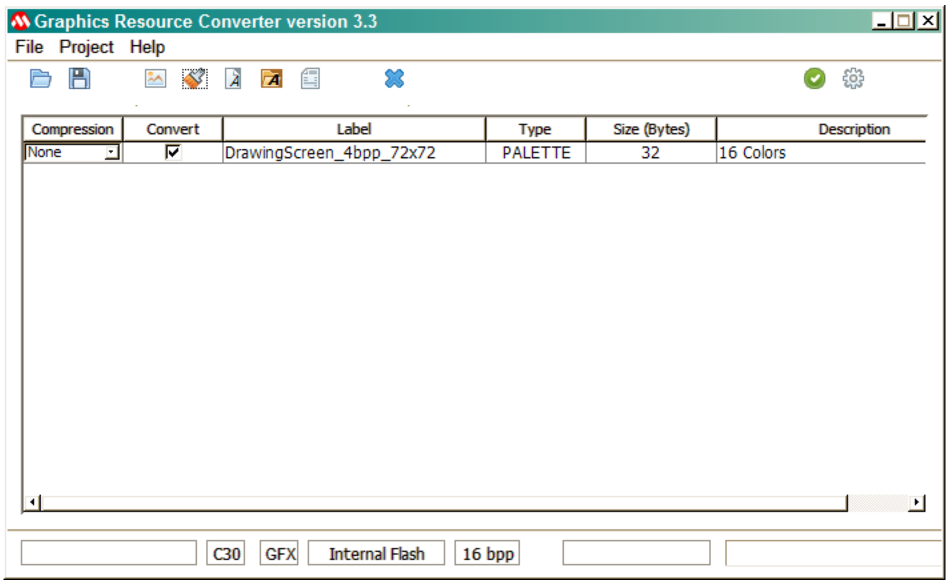
Look in: bitmaps

.svn, 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, FolderPhoto_mono_72x72.bmp, food.bmp, GOL_Palette_Demo.gpl, 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

File name: DrawingScreen_4bpp_72x72.bmp

Files of type: Palette Files (*.bmp, *.gpl)

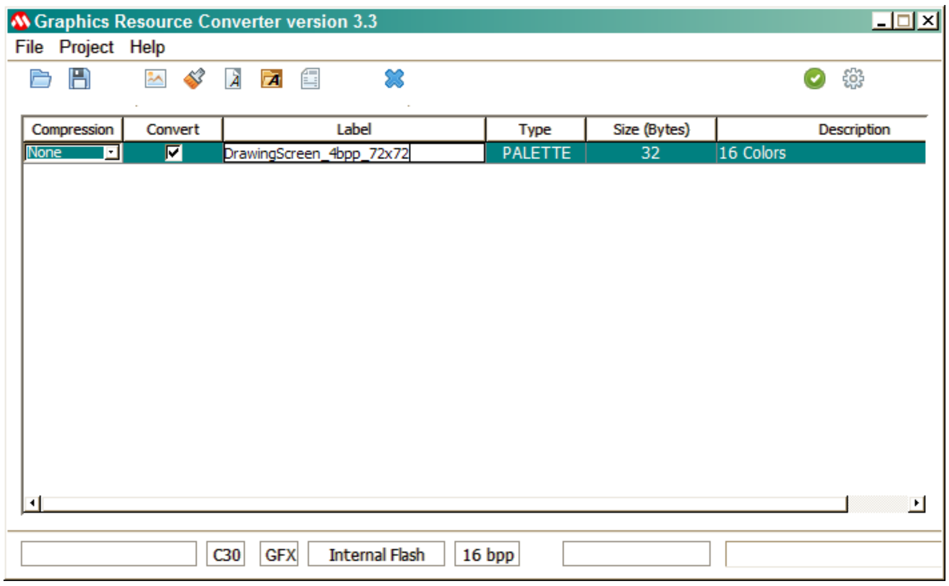
3. If selected file will be imported successfully the information about it (label, type, size and description) will be added in the list box of the main window. Label is generated from the file name, type defines that imported object is a palette, size of data is shown in bytes, and description contains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


Graphics Resource Converter version 3.3

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

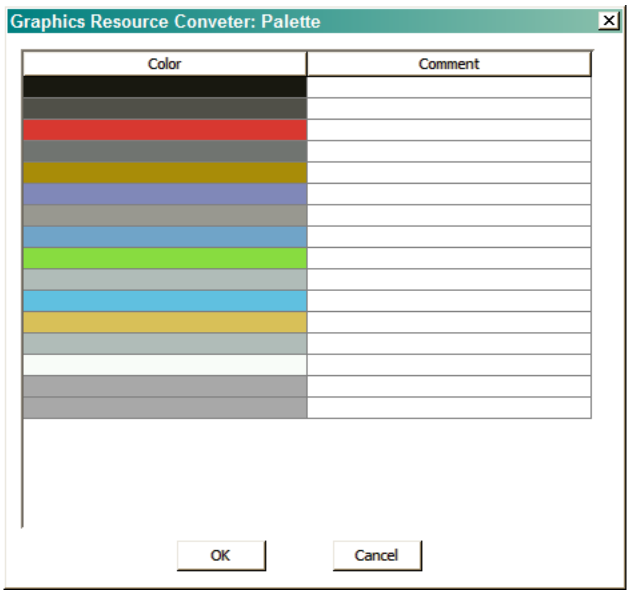
C30 | GFX | Internal Flash | 16 bpp

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.



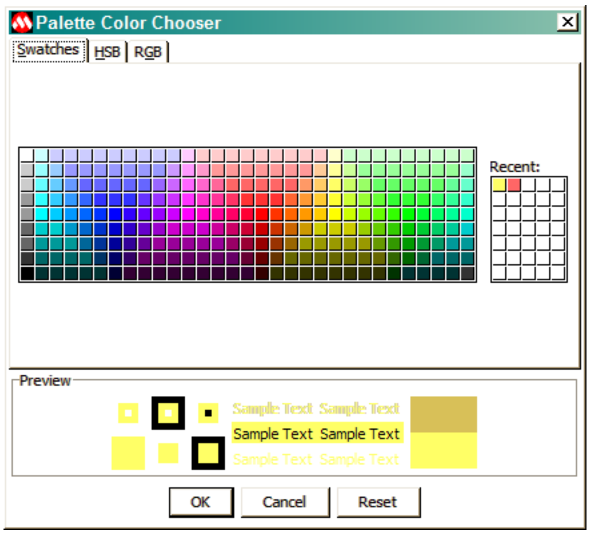
Compression	Convert	Label	Type	Size (Bytes)	Description
None	✓	DrawingScreen_4bpp_72x72	PALETTE	32	16 Colors

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

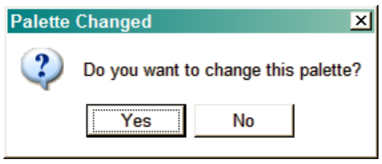


Color	Comment
⬛	
⬛	
🟥	
⬛	
🟨	
🟦	
⬜	
🟦	
🟩	
⬜	
🟦	
🟨	
⬜	
⬜	
⬜	
⬜	

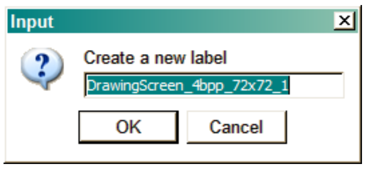
6. Double click on the color entry in the table to edit it. A color chooser will appear. Select the new color and select **OK**.

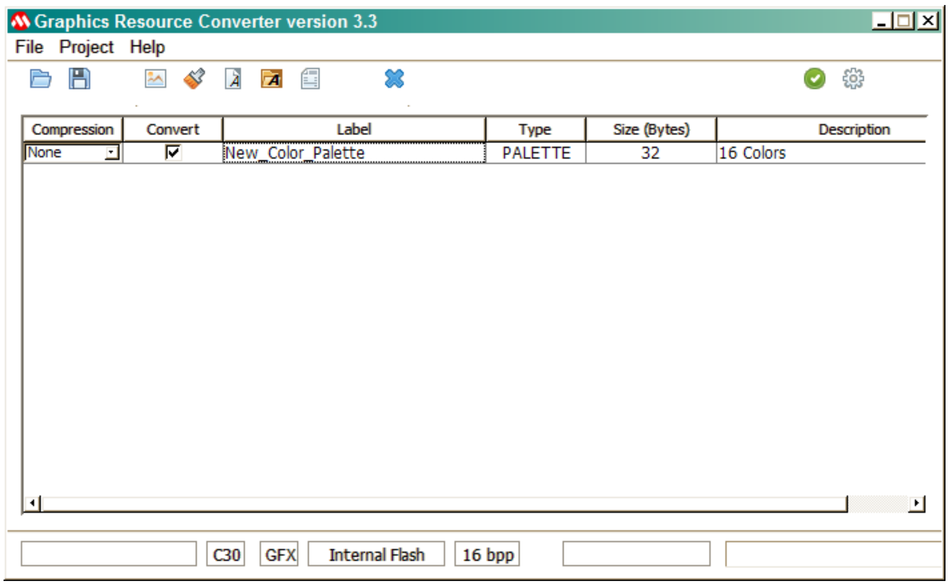


7. After done making edits, select **OK.** If you want to change the palette, select **YES**.



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

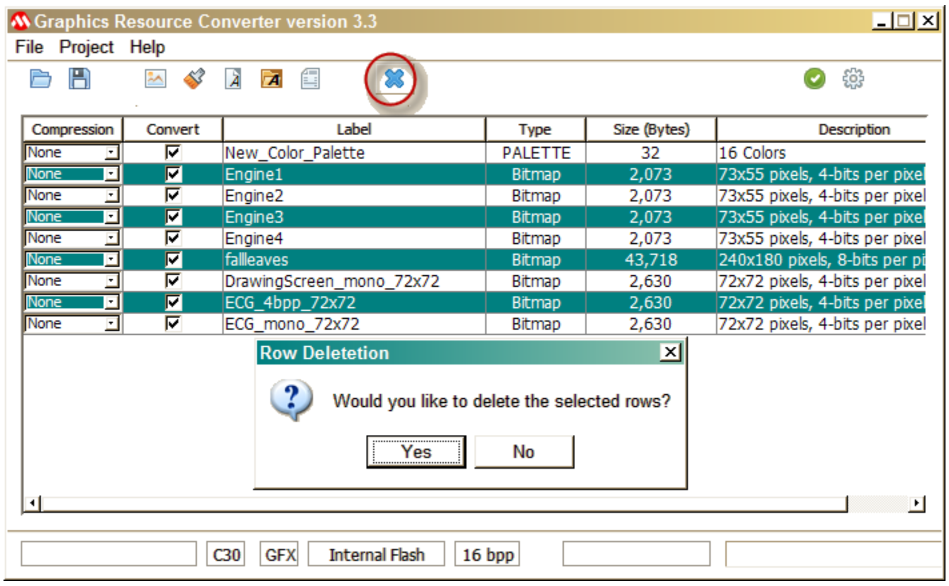




Compression	Convert	Label	Type	Size (Bytes)	Description
None	✓	New_Color_Palette	PALETTE	32	16 Colors

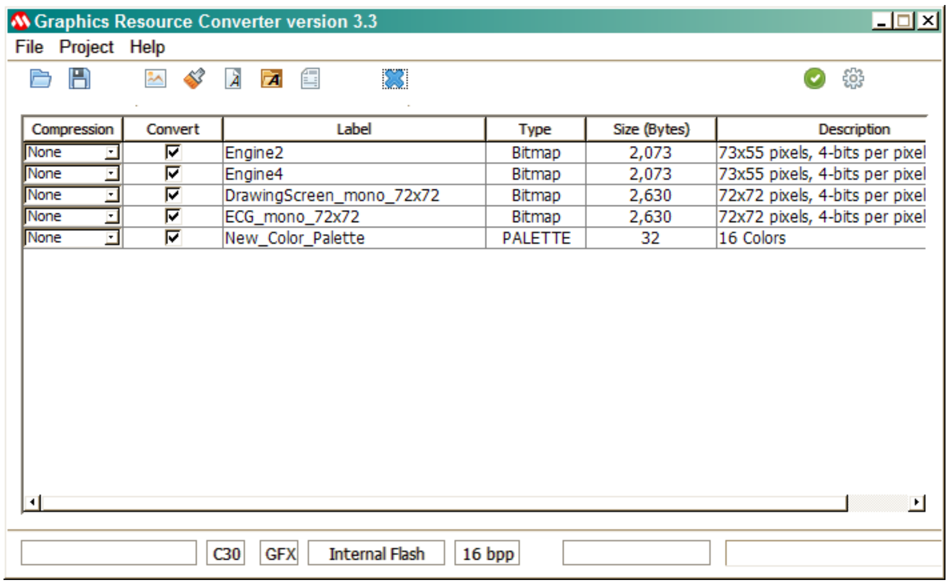
5.5 Removing Item from the Conversion List

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	✓	New_Color_Palette	PALETTE	32	16 Colors
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	✓	Engine4	Bitmap	2,073	73x55 pixels, 4-bits per pixel
None	✓	fallleaves	Bitmap	43,718	240x180 pixels, 8-bits per pi
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

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	✓	DrawingScreen_mono_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel
None	✓	ECG_mono_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel
None	✓	New_Color_Palette	PALETTE	32	16 Colors

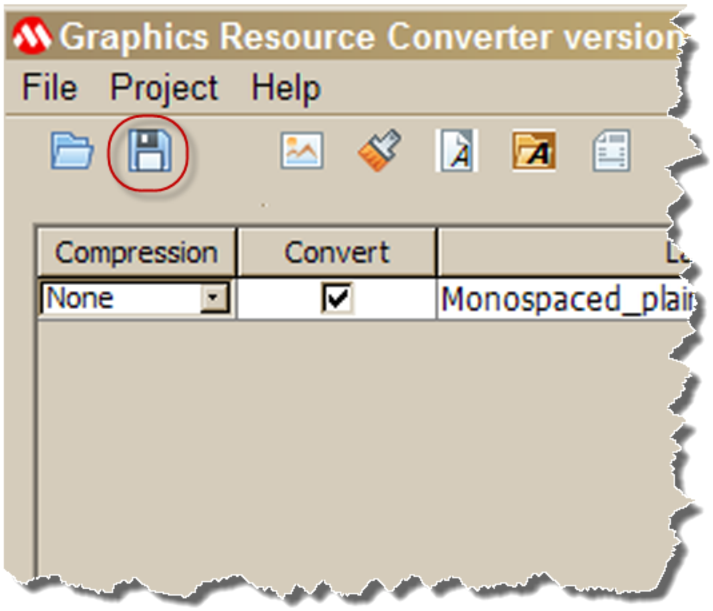
5.6 Saving and Loading Projects

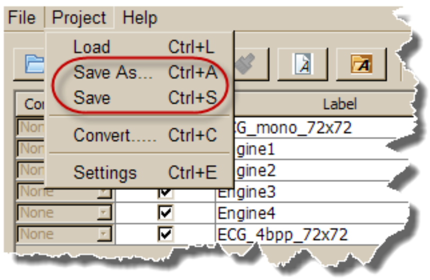
The utility allows users to save the loaded images, palettes and fonts into a project and to load a previously saved project.

Saving a Project

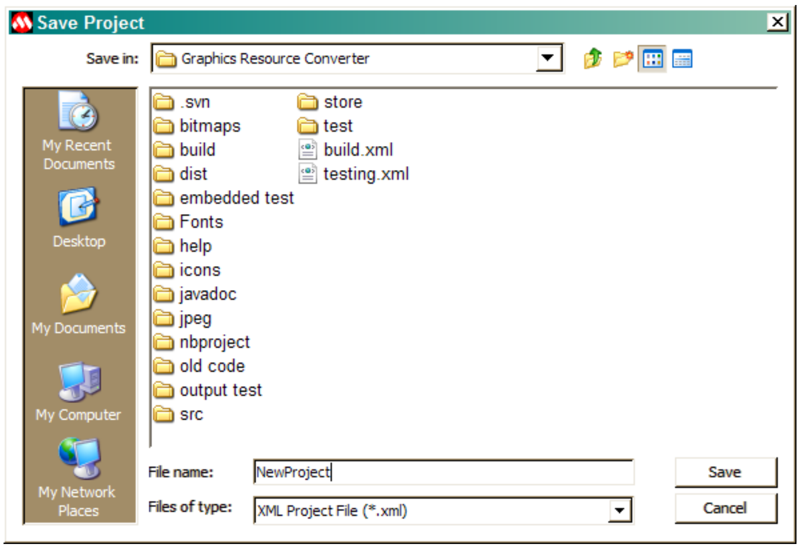
To save a project the following steps should be done:

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





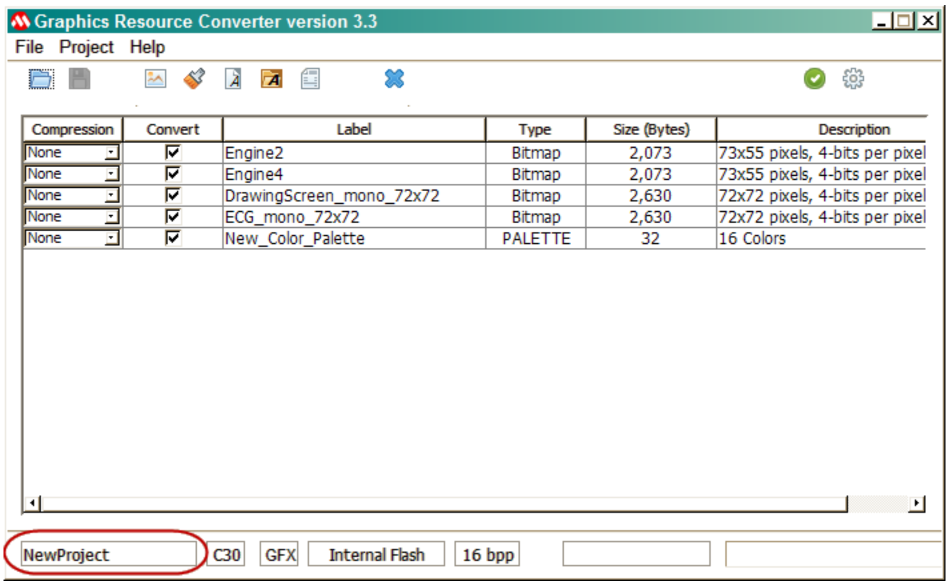
2. A Save As dialog will appear and, by default, filters the project file (*.xml filter).Type the new project file name or browse for the project file that will be overwritten and and press **Save** button.



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.



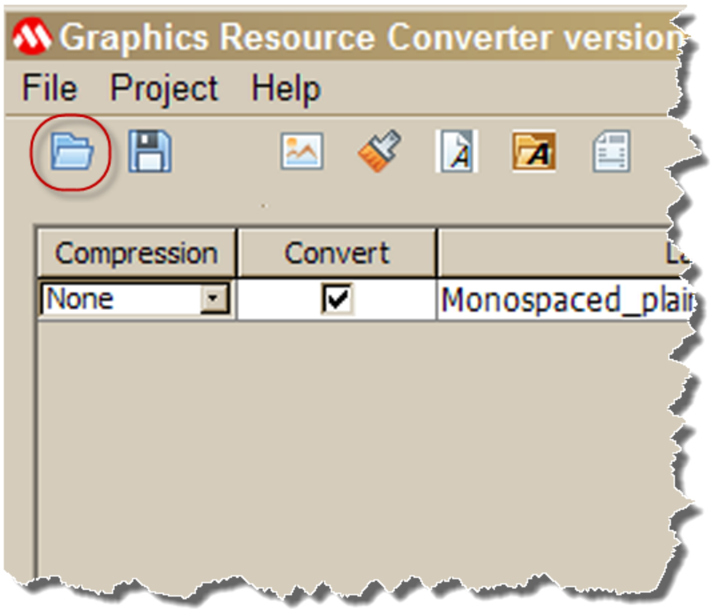
Compression	Convert	Label	Type	Size (Bytes)	Description
None	✓	Engine2	Bitmap	2,073	73x55 pixels, 4-bits per pixel
None	✓	Engine4	Bitmap	2,073	73x55 pixels, 4-bits per pixel
None	✓	DrawingScreen_mono_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel
None	✓	ECG_mono_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel
None	✓	New_Color_Palette	PALETTE	32	16 Colors

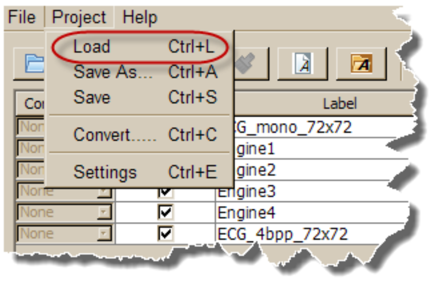
NewProject C30 GFX Internal Flash 16 bpp

Loading a Project

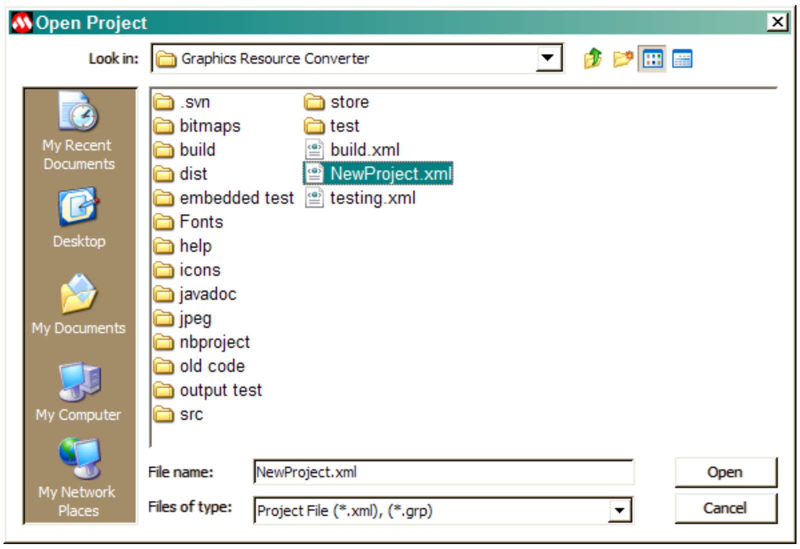
To load a project the following steps should be done:

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.



Open Project

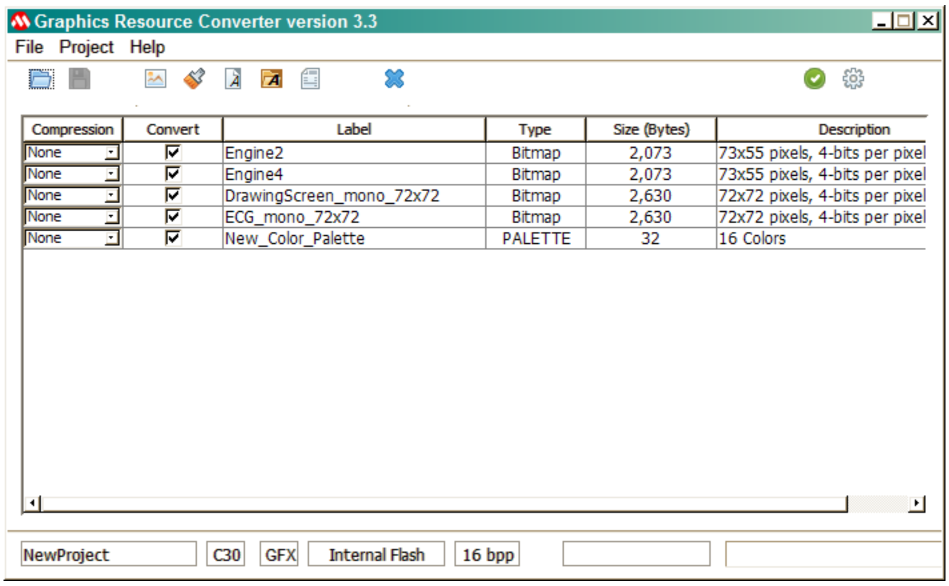
Look in: Graphics Resource Converter

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

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.



Graphics Resource Converter version 3.3

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	✓	DrawingScreen_mono_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel
None	✓	ECG_mono_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel
None	✓	New_Color_Palette	PALETTE	32	16 Colors

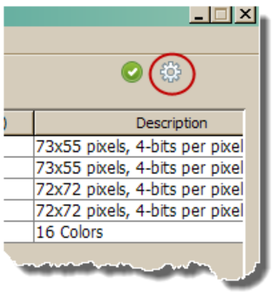
NewProject | C30 | GFX | Internal Flash | 16 bpp

5.7 Settings

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

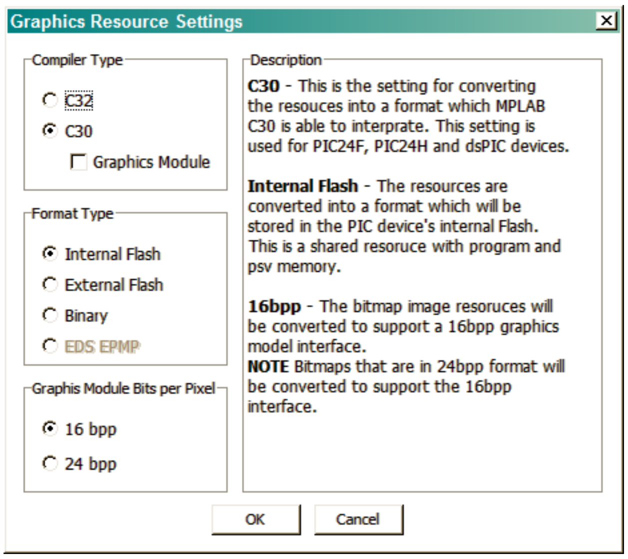
To change the Graphics Resource Converter's settings

1. Press the **Settings** button.



Description
73x55 pixels, 4-bits per pixel
73x55 pixels, 4-bits per pixel
72x72 pixels, 4-bits per pixel
72x72 pixels, 4-bits per pixel
16 Colors

2. The Graphics Resource Setting dialog box will appear. The left side of the dialog has all of the configuration options, while the right side give a description of the configuration options that are currently selected.



Graphics Resource Settings

Compiler Type
- ( ) C32
- (●) C30
  - [ ] Graphics Module

Format Type
- (●) Internal Flash
- ( ) External Flash
- ( ) Binary
- ( ) EDS EPMP

Graphis Module Bits per Pixel
- (●) 16 bpp
- ( ) 24 bpp

Description

C30 - This is the setting for converting the resouces into a format which MPLAB C30 is able to interprate. This setting is used for PIC24F, PIC24H and dsPIC devices.

Internal Flash - The resources are converted into a format which will be stored in the PIC device's internal Flash. This is a shared resoruce with program and psv memory.

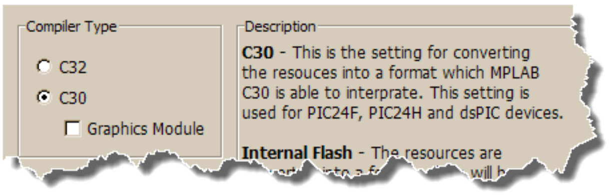
16bpp - The bitmap image resources will be converted to support a 16bpp graphics model interface.
NOTE Bitmaps that are in 24bpp format will be converted to support the 16bpp interface.

3. The Compiler Type panel contains the compiler build type

   1. C32 - Select this to convert for a format which MPLAB C32 is able to interpret. This setting is used for PIC32MX devices.

   2. C30 - Select this to convert for a format which MPLAB C30 is able to interpret. This setting is used for PIC24F, PIC24H and dsPIC devices.

   3. Checking the **Graphics Module** will indicate that the device also has an internal graphics module. Please refer to the device's datasheet or Family Reference Manual to determine if it has an internal graphics module.



Compiler Type
- ( ) C32
- (●) C30
  - [ ] Graphics Module

Description

C30 - This is the setting for converting the resouces into a format which MPLAB C30 is able to interprate. This setting is used for PIC24F, PIC24H and dsPIC devices.

Internal Flash - The resources are

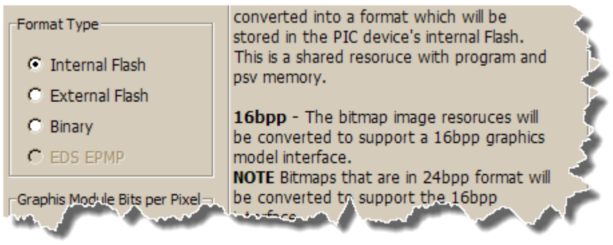
4. The Format Type panel contains the medium for the converted resources

   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

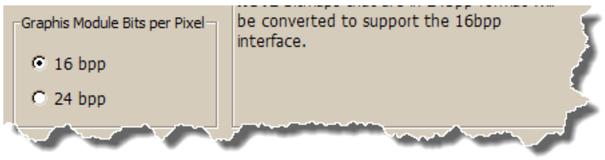
[✓ Internal Flash]
External Flash
Binary
EDS EPMP

Graphis Module Bits per Pixel

converted into a format which will be stored in the PIC device's internal Flash. This is a shared resoruce with program and psv memory.

16bpp - The bitmap image resoruces will be converted to support a 16bpp graphics model interface.
NOTE Bitmaps that are in 24bpp format will be converted to support the 16bpp

5. The Graphics Module Bits per Pixel option is for selecting the output of the converted image resources. Some graphics devices support different color depth. This option will allow the suer to select the color depth of 16bpp or 24bpp. It is import to note that this setting does not determine the color depth of the input resources. The application will down covert if needed. For example, a bitmap image that is 24bpp will be converted as a 16bpp if the Graphics Module Bits per Pixel setting is 16bpp.


Graphis Module Bits per Pixel

[✓ 16 bpp]
24 bpp

be converted to support the 16bpp interface.

6. After selecting the desired configuration settings, select **OK** and the status bar will populated with the current configuration settings. These configuration settings are saved with the project.

5.8 Converting

5.8.1 Converting into C file

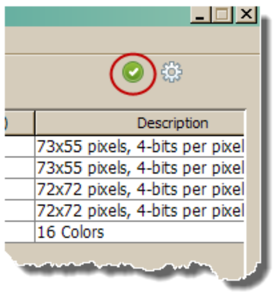
There is a limitation on the memory used when generating font images in C files (to be stored in internal flash for PIC24 devices). The font images are placed in the const section. The const section has a maximum size of 32 Kbytes. Therefore, the maximum size that the font image can have is 32 Kbyte 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.

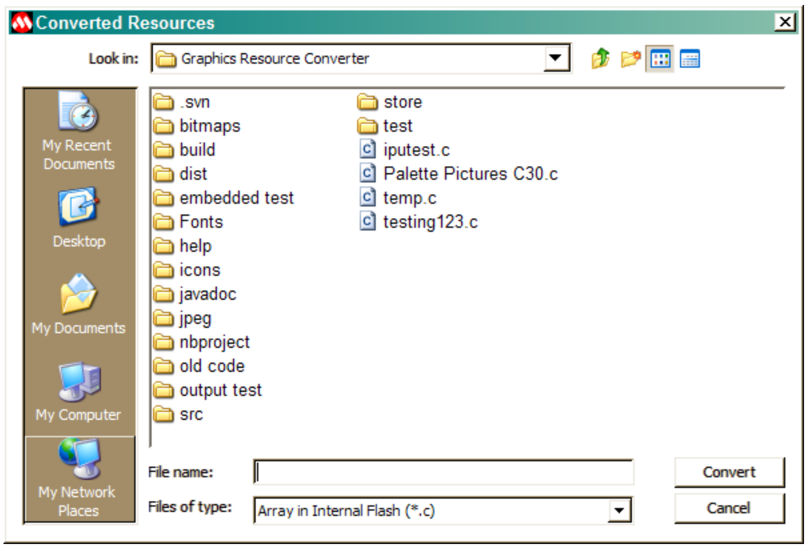
Conversion of C file containing arrays to be located in internal flash memory is similar to the conversion of the Hex file.

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

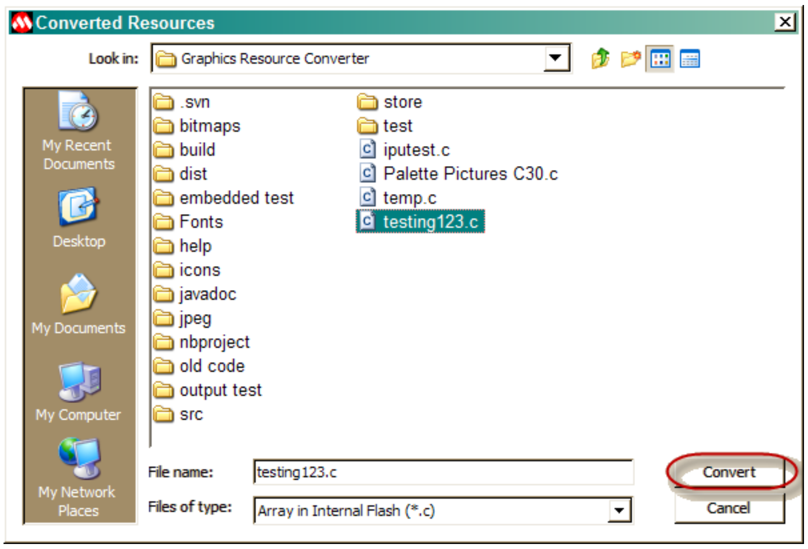
1. Press **Convert** button.


Description
73x55 pixels, 4-bits per pixel
73x55 pixels, 4-bits per pixel
72x72 pixels, 4-bits per pixel
72x72 pixels, 4-bits per pixel
16 Colors

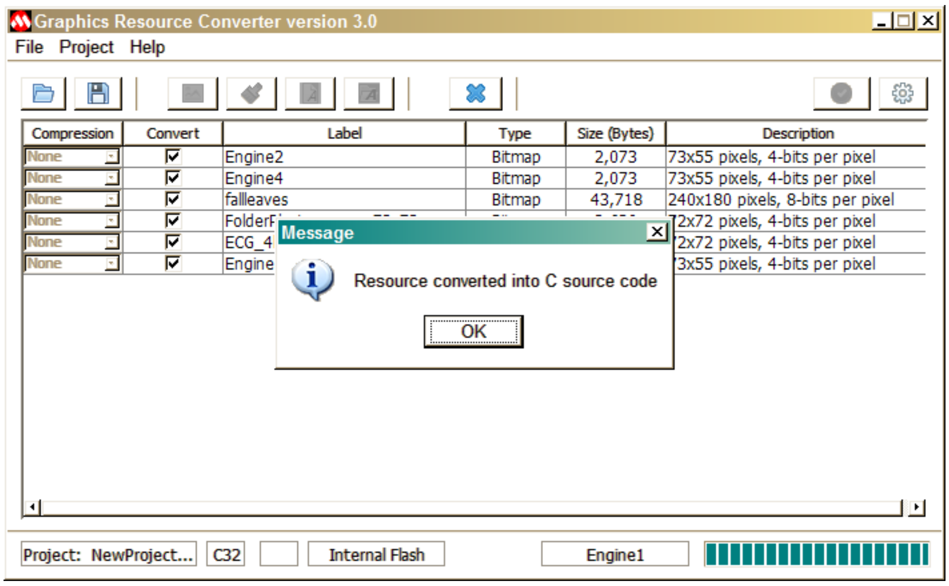
2. “Converted Resources” dialog will appear.



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	✓	FolderF			2x72 pixels, 4-bits per pixel
None	✓	ECG_4			2x72 pixels, 4-bits per pixel
None	✓	Engine			3x55 pixels, 4-bits per pixel

Message

Resource converted into C source code

OK

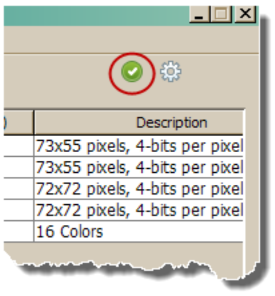
Project: NewProject... | C32 | Internal Flash | Engine1

5.8.2 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 49) description.

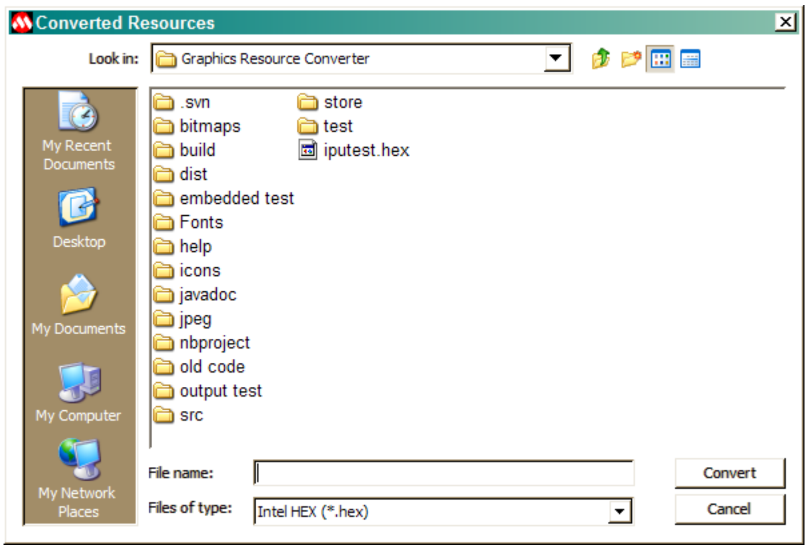
To create Intel Hex and reference C files following steps must be performed:

1. Press **Convert** button.

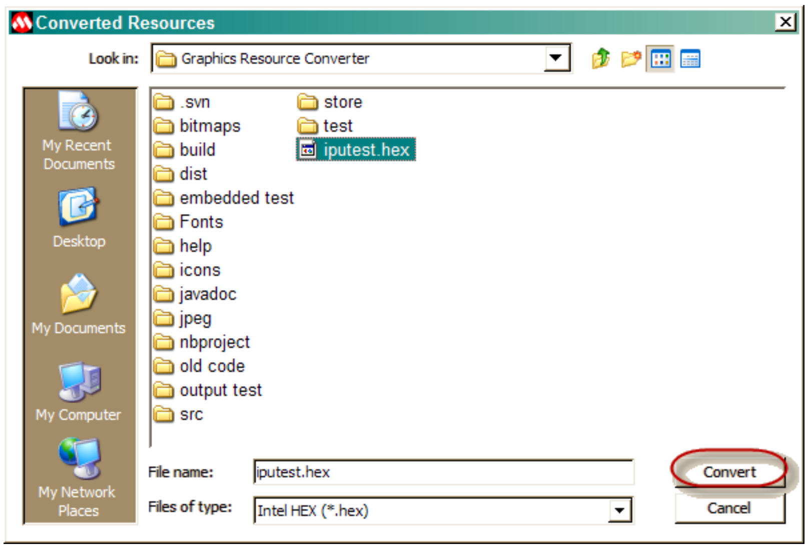


Description
73x55 pixels, 4-bits per pixel
73x55 pixels, 4-bits per pixel
72x72 pixels, 4-bits per pixel
72x72 pixels, 4-bits per pixel
16 Colors

2. "Convert Resources" dialog will appear.

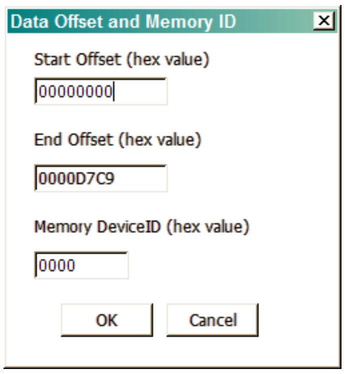


3. Press **Convert** button.



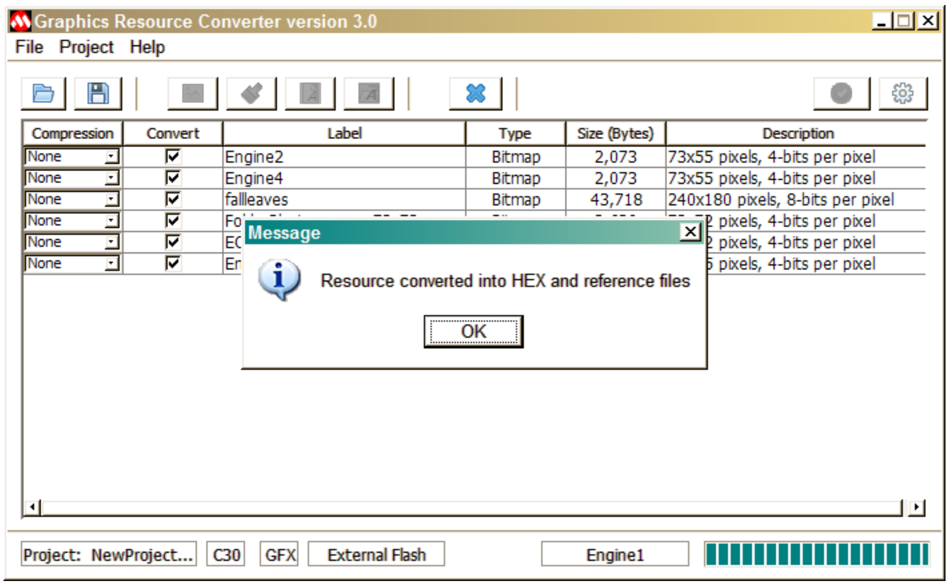
File name: iputest.hex
Files of type: Intel HEX (*.hex)

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.

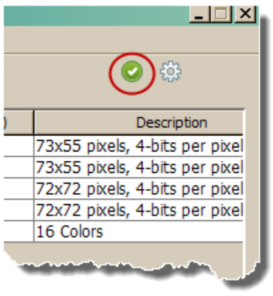

Graphics Resource Converter version 3.0
File Project Help
Message
Resource converted into HEX and reference files
OK
Project: NewProject... C30 GFX External Flash Engine1

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

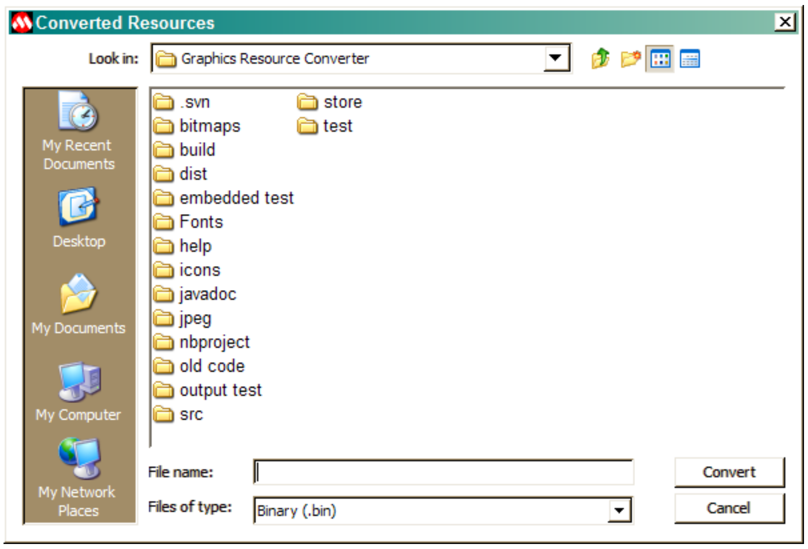
5.8.3 Converting into Binary file

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:

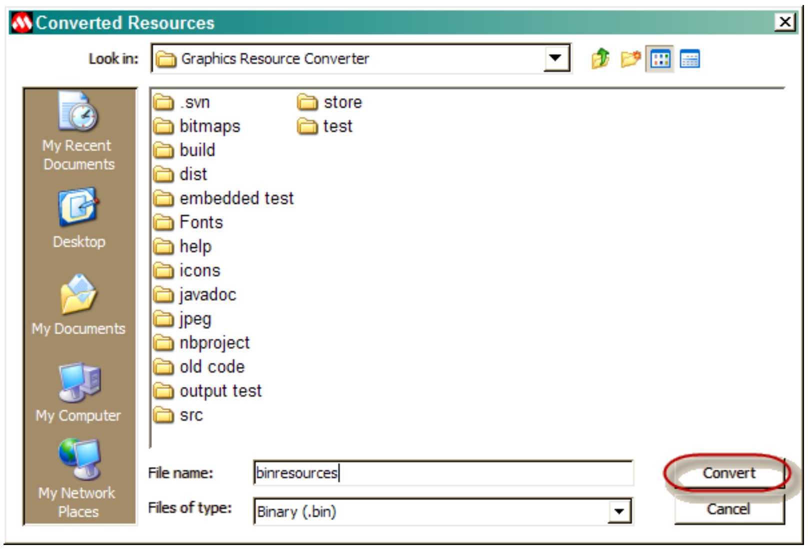
1. Press **Convert** button.


Description
73x55 pixels, 4-bits per pixel
73x55 pixels, 4-bits per pixel
72x72 pixels, 4-bits per pixel
72x72 pixels, 4-bits per pixel
16 Colors

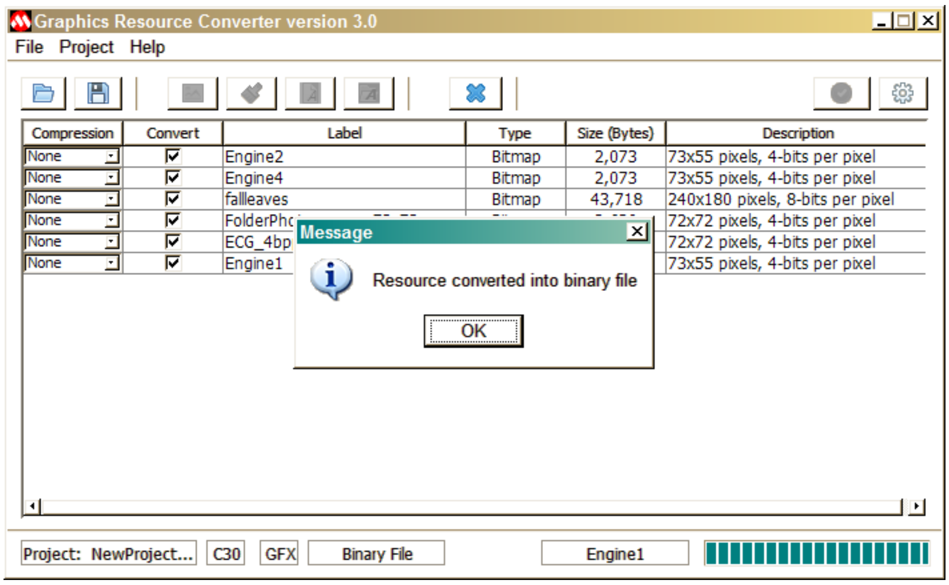
2. “Converted Resources” dialog will appear.



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

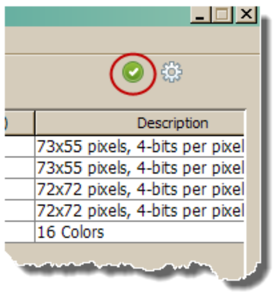
Resource converted into binary file

5.8.4 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 49) 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.

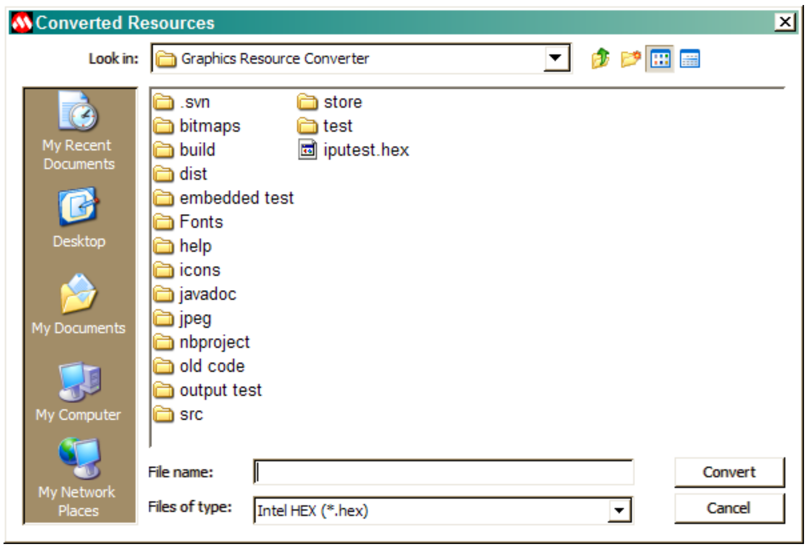
To create Intel Hex and reference C files following steps must be performed:

1. Press **Convert** button.



Description
73x55 pixels, 4-bits per pixel
73x55 pixels, 4-bits per pixel
72x72 pixels, 4-bits per pixel
72x72 pixels, 4-bits per pixel
16 Colors

2. “Convert Resources” dialog will appear.



Converted Resources

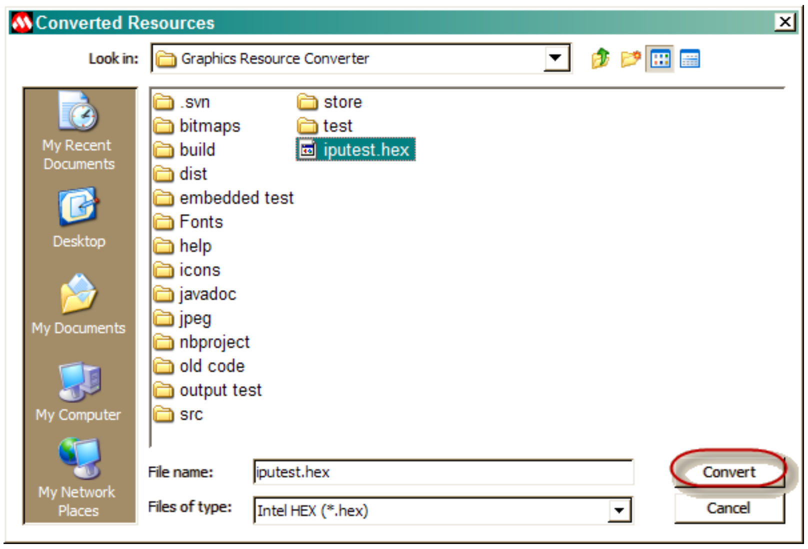
Look in: Graphics Resource Converter

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

File name:

Files of type: Intel HEX (*.hex)

3. Press **Convert** button.



Converted Resources

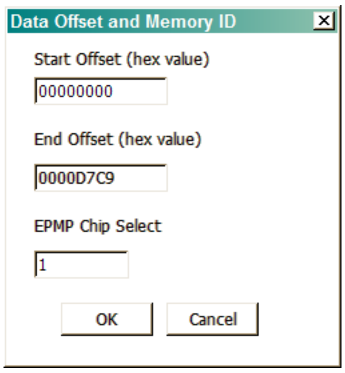
Look in: Graphics Resource Converter

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

File name: iputest.hex

Files of type: Intel HEX (*.hex)

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

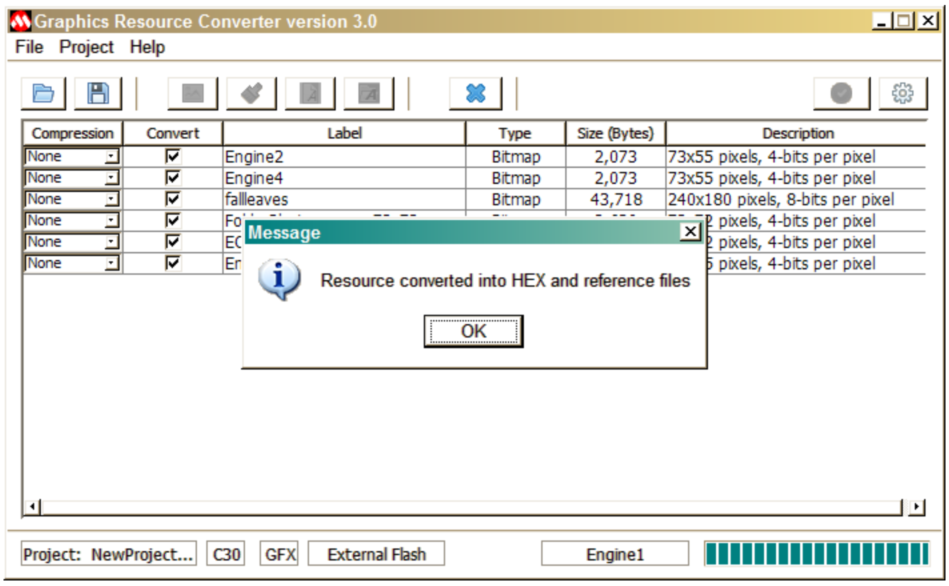


Start Offset (hex value)
00000000

End Offset (hex value)
0000D7C9

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.



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

Resource converted into HEX and reference files

Project: NewProject... C30 GFX External Flash Engine1

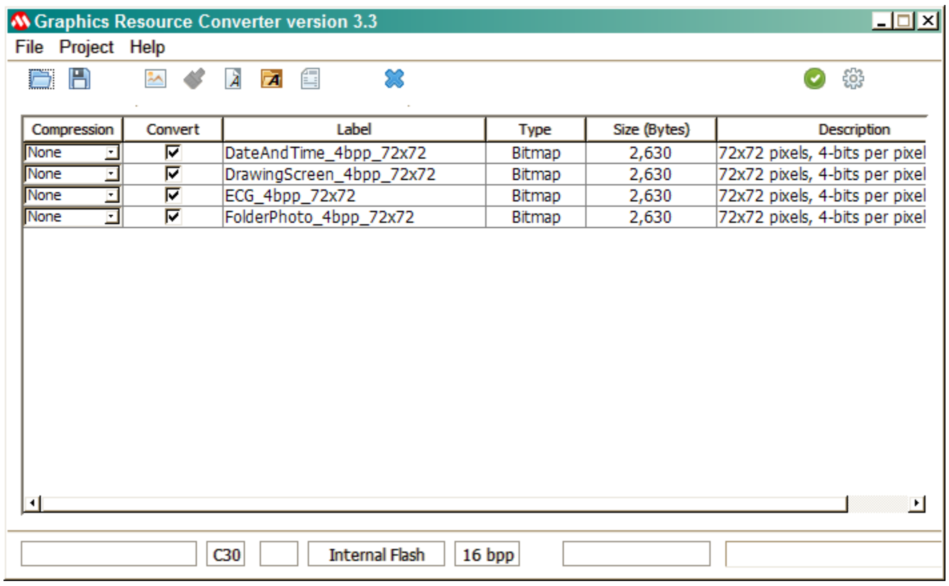
5.9 Generating a Palette from Bitmap Images

The converter can generate a palette based on bitmap images that are loaded into the resource table. If there are any JPEG images, the palette obtain will not be available. The palette that is generated will be up to 256 colors and the bitmap images will be reformatted to use the generated palette.

Generating a Palette

To generate a palette follow these steps:

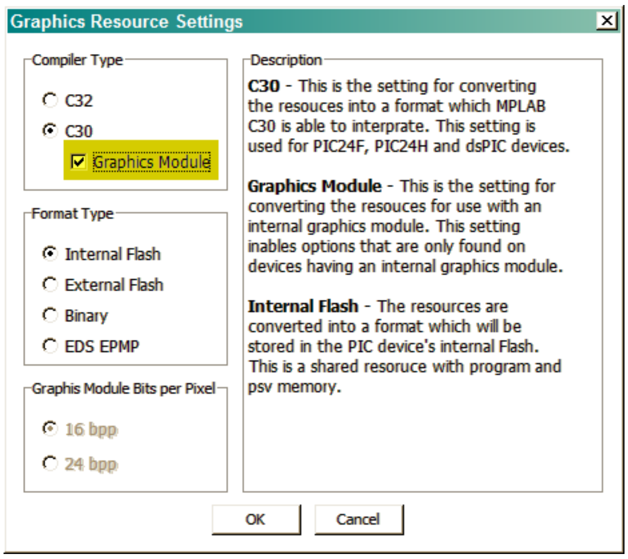
1. Add all of the bitmap images to the converter for the project.



Compression	Convert	Label	Type	Size (Bytes)	Description
None	✓	DateAndTime_4bpp_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel
None	✓	DrawingScreen_4bpp_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel
None	✓	ECG_4bpp_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel
None	✓	FolderPhoto_4bpp_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel

C30 | Internal Flash | 16 bpp

2. Under the settings dialog box, make sure that the graphics module option is selected. Palettes will not be generated for projects that do not have a graphics module.



Compiler Type: C32 [✓ C30], [✓ Graphics Module]

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

Graphis Module Bits per Pixel: [✓ 16 bpp], 24 bpp

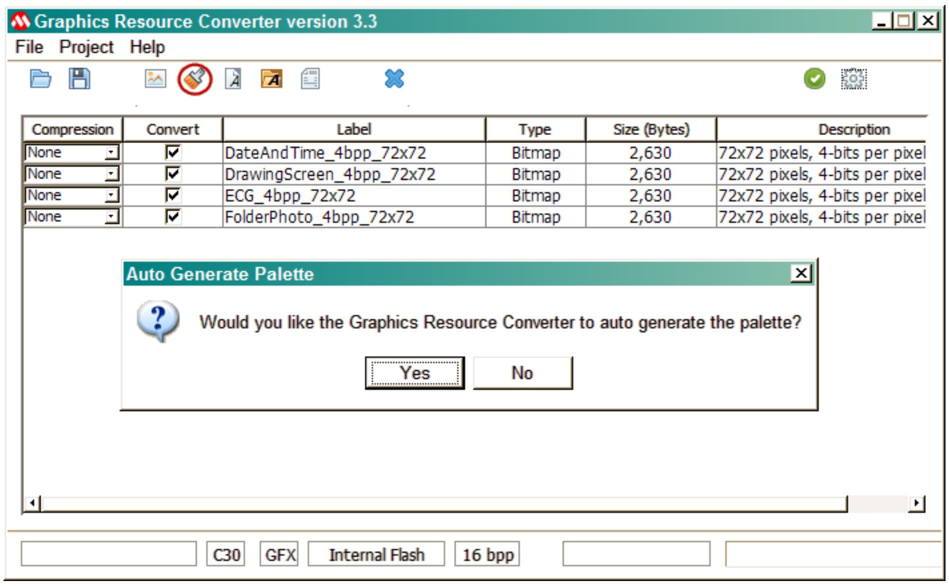
Description

C30 - This is the setting for converting the resouces into a format which MPLAB C30 is able to interprate. This setting is used for PIC24F, PIC24H and dsPIC devices.

Graphics Module - This is the setting for converting the resouces for use with an internal graphics module. This setting inables options that are only found on devices having an internal graphics module.

Internal Flash - The resources are converted into a format which will be stored in the PIC device's internal Flash. This is a shared resoruce with program and psv memory.

3. When prompted, choose **YES** to allow the converter to auto generate the palette.

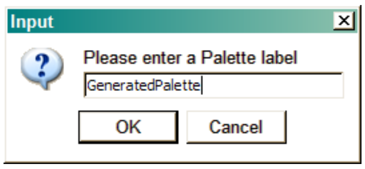


Compression	Convert	Label	Type	Size (Bytes)	Description
None	☑	DateAndTime_4bpp_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel
None	☑	DrawingScreen_4bpp_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel
None	☑	ECG_4bpp_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel
None	☑	FolderPhoto_4bpp_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel

Auto Generate Palette

Would you like the Graphics Resource Converter to auto generate the palette?

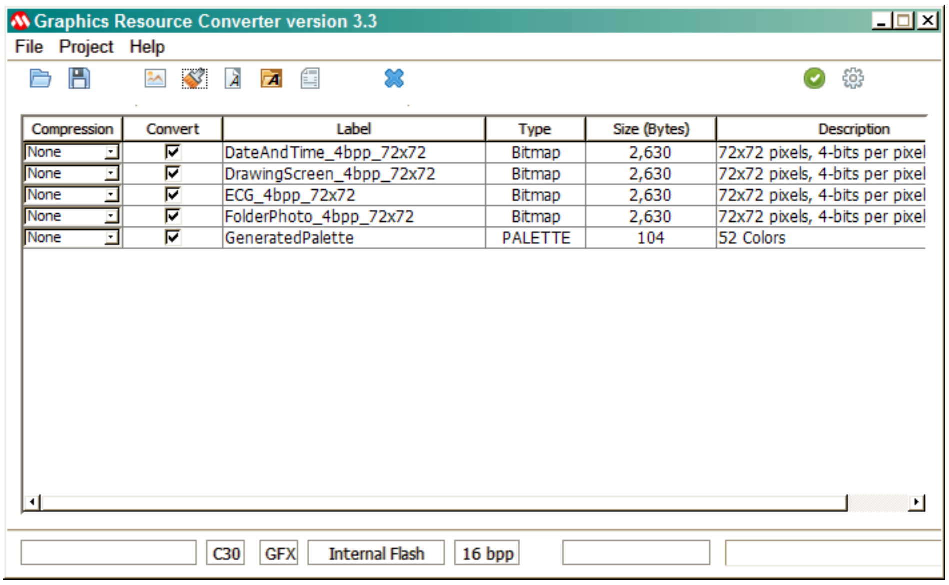
4. Enter the palette label that you wish to use.



Please enter a Palette label

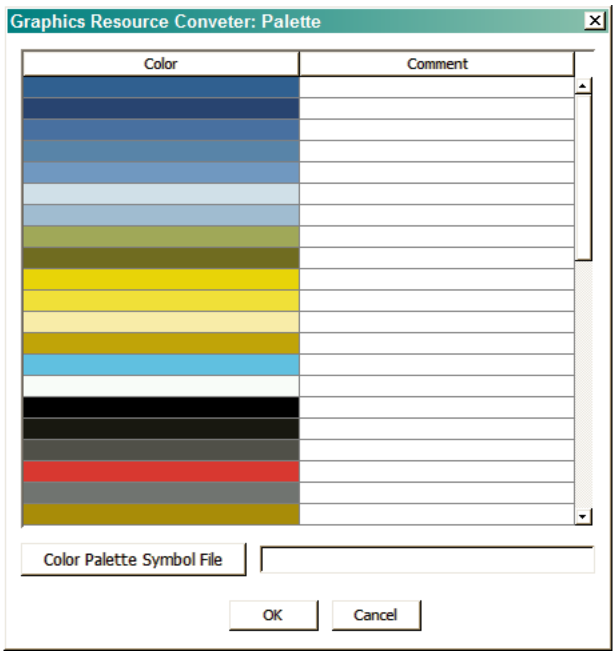
GeneratedPalette

5. You will see that a palette has been generated from the individual bitmap images.

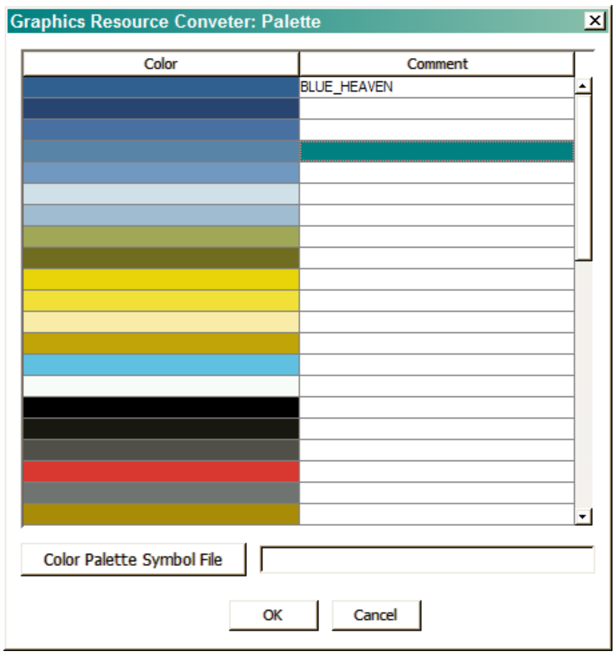


Compression	Convert	Label	Type	Size (Bytes)	Description
None	☑	DateAndTime_4bpp_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel
None	☑	DrawingScreen_4bpp_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel
None	☑	ECG_4bpp_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel
None	☑	FolderPhoto_4bpp_72x72	Bitmap	2,630	72x72 pixels, 4-bits per pixel
None	☑	GeneratedPalette	PALETTE	104	52 Colors

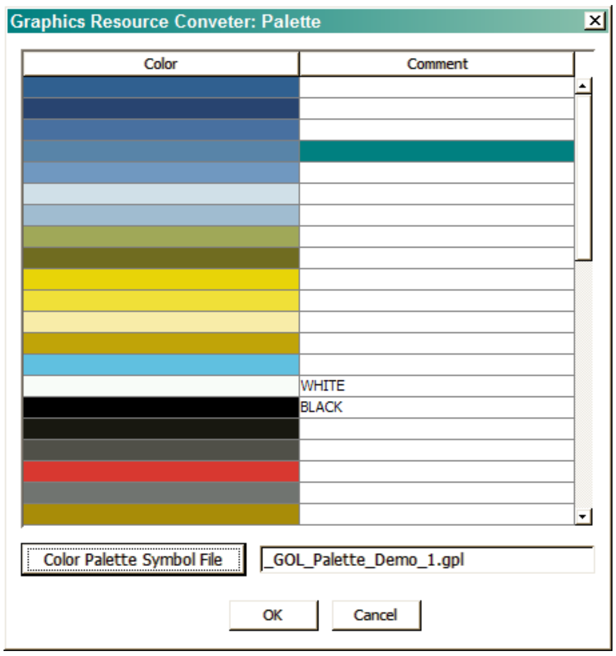
6. By double clocking on the palette resource in the resource table, you will bring up the color chart.


Graphics Resource Conveter: Palette
Color
Comment
Color Palette Symbol File
OK
Cancel

7. You can manually add in comments that, when converted will become part of a palette color defines header file.
    1. The application can include this file to reference the color palette index via a symbol


Graphics Resource Conveter: Palette
Color
Comment
BLUE_HEAVEN
Color Palette Symbol File
OK
Cancel

8. Selecting the Color Palette Symbol File button will load a GIMP palette and use the comments in the file.
    1. The generated color palette will match the colors in the GIMP palette file to place the comments


Graphics Resource Conveter: Palette
Color
Comment
WHITE
BLACK
Color Palette Symbol File
_GOL_Palette_Demo_1.gpl
OK
Cancel

9. Convert the project into the medium of choice.

5.10 Command Line Interface

Starting with Graphics Resource Converter version 3.3, a command line interface is supported. This interface allows for resource conversion to be done without launching the GUI. The JAVA runtime engine (JRE) is still requirement to run the command line interface.

RUNNING FROM THE COMMAND LINE

The Graphics Resource Converter, GRC, can be run through a command line interface. Passing arguments will determine if the GUI is launched or the command line interface is used.

To run the GRC GUI from the command line:

```
>java –jar “<Microchip Applications Library directory>/Microchip/Graphics/bin/grc/grc.jar”
```

To run the GRC without launching the GUI:

```
>java –jar “<Microchip Applications Library directory>/Microchip/Graphics/bin/grc/grc.jar”
<options>
```

where `<options>` are the GRC command line options.

COMMAND LINE OPTIONS

All command line options and associated values must be separated by a space. If the value of the argument contains a space, it must be surrounded by quotes.

For example:

-T I -- This command line switch is –T and the value is I.

-O “output file.c” – This command line switch is –O and the value is “output file.c.” Since the value has a space, it must be surrounded by quotes

Resource Type (-T)

I – An image, either bitmap or JPEG.

F – A font, either an installed or font file.

Resource Sub-type (-U)

I – installed font

T – TTF font file

R – raster font file

Output Format (-F)

C – C array

H – Hex

B – Binary

Input File (-I)

The value will be the file name and path. The path can be relative or absolute. It is recommended to surround the file and path with quotes.

Output File (-O)

The value will be the file name and path. The path can be relative or absolute. It is recommended to surround the file and path with quotes.

Font Start Character (-S)

The value is the starting character of the font.

Font End Character (-E)

The value is the ending character of the font.

Font Style Italic (-A)

This option does not have a value associated with it. The font that is being converted will be italic.

Font Name (-N)

The value is the font name. For example, “Times New Roman” is the name of the font used to write this document.

Font Size (-Z)

The value is the font size.

Font Filter File (-R)

The value will be the file name and path. The path can be relative or absolute. It is recommended to surround the file and path with quotes.

Build Type (-B)

C30 – For PIC24 and dsPIC

C32 – For PIC32

Compression (-P)

IPU – Deflate compression, only for DA210 applications

RLE – Run length encoding

Hex File Starting Address (-HA)

A decimal or hexi-decimal number where the beginning of the resource generate will start. For example, the option -HA 128 will have the resource conversions starting at address 128. The same is for the option -HA 0x80, the resource conversion will start at address 128.

Project File (-XML)

The value will be the relative path of the project file. The project configuration and resources will be converted.

Label (-X)

The value should start with a letter and have not spaces.

Break (;)

Example

Here is an example of some command line options:

```
java  -jar  "../store/grc.jar"  -T  I  -F  C  -I  "images/DateAndTime_4bpp_72x72.bmp"  -O
"compression/commandline_test_carray.c" -B C30 -X DandT_Uncompressed_4bpp ; -T I -F C -I
```

```
"images/DateAndTime_4bpp_72x72.bmp" -O "compression/commandline_test_carray.c" -B C30 -X
DandT_Compressed_RLE_4bpp -P RLE ; -T I -F C -I "images/DateAndTime_8bpp_72x72.bmp" -O
"compression/commandline_test_carray.c" -B C30 -X DandT_Uncompressed_8bpp ; -T I -F C -I
"images/DateAndTime_8bpp_72x72.bmp" -O "compression/commandline_test_carray.c" -B C30 -X
DandT_Compressed_RLE_8bpp -P RLE ; -T I -F C -I "images/sunset1.bmp" -O
"compression/commandline_test_carray.c" -B C30 -X sunset_Uncompressed ; -T I -F C -I
"images/sunset1.bmp" -O "compression/commandline_test_carray.c" -B C30 -X
sunset_Compressed_IPU -P IPU
```

6 Input and Output File Formats

6.1 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.

6.2 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.

6.3 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.

7 Output Image and Font Data Formats

7.1 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 (16bpp)

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	

Color Table Entry Format (24bpp)

Bits	31	30	29	28	27	26	25	24
Value	0	0	0	0	0	0	0	0

Bits	23	22	21	20	19	18	17	16
Value	R	R	R	R	R	R	R	R

Bits	15	14	13	12	11	10	9	8
Value	G	G	G	G	G	G	G	G

Bits	7	6	5	4	3	2	1	0
Value	B	B	B	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 if the Graphics Module configuration setting is set to 16bpp. Every 2bytes / 16bit holds 1 pixel. The pixels are not 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.

7.2 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.

8 Examples

8.1 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 48) 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 49) will contain this declaration:

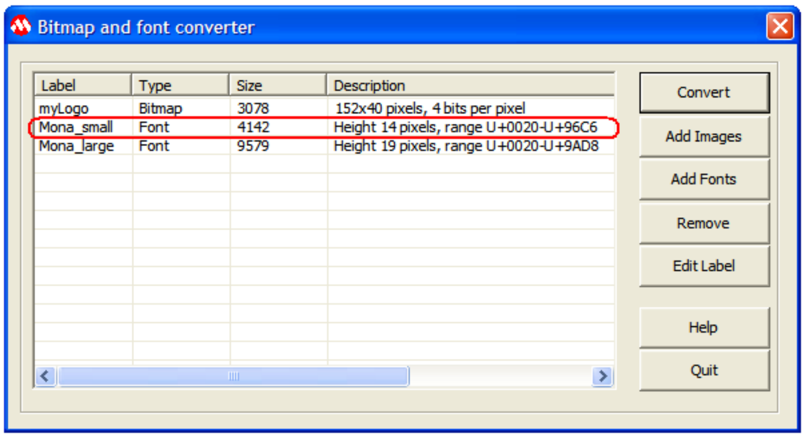
```
XCHAR HelloStr[] = {0x002B,0x0034,0x0037,0x0037,0x0038,0x0020,
                    0x0030,0x0038,0x003A,0x0037,0x0033,0x0000}; // using reduced font table
```

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.

8.2 Generating Bitmaps and Multiple Font Tables

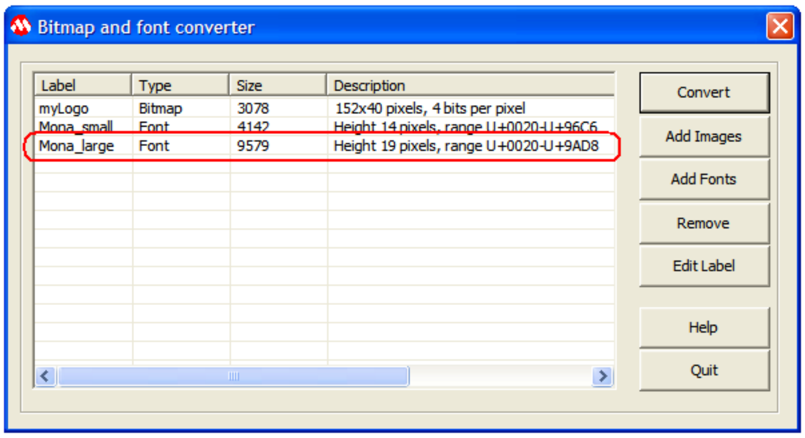
In applications that uses multiple font types and sizes, multiple font tables will be generated. To illustrate, the example below shows two generated font tables and a bitmap.



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

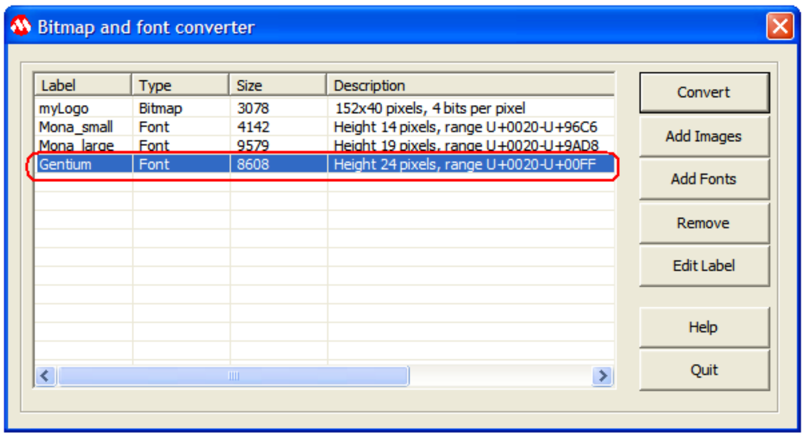
The item encircled in red used Mona font and a font filter file defining the strings and characters that will be included in the table. The font size is shown as 14 pixels. The lowest character ID is 0x0020 and the highest ID is 0x96C6. The font table size that will be generated will depend on the number of characters defined in the font filter file (please see Font Filter File Format (see page 48) 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.

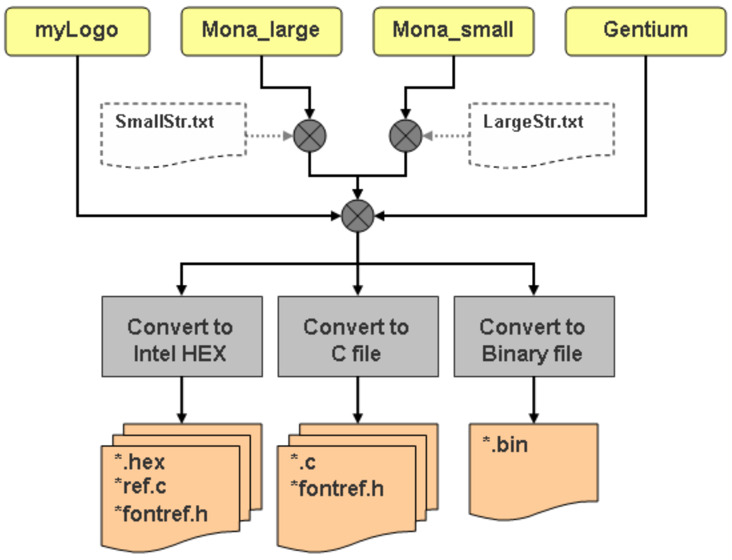


Label	Type	Size	Description
myLogo	Bitmap	3078	152x40 pixels, 4 bits per pixel
Mona_small	Font	4142	Height 14 pixels, range U+0020-U+96C6
Mona_large	Font	9579	Height 19 pixels, range U+0020-U+9AD8
Gentium	Font	8608	Height 24 pixels, range U+0020-U+00FF

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