

**S1C31 Family
emWin
Application Note**

Evaluation board/kit and Development tool important notice

1. This evaluation board/kit or development tool is designed for use for engineering evaluation, demonstration, or development purposes only. Do not use it for other purposes. It is not intended to meet the requirements of design for finished products.
2. This evaluation board/kit or development tool is intended for use by an electronics engineer and is not a consumer product. The user should use it properly and in a safe manner. Seiko Epson does not assume any responsibility or liability of any kind of damage and/or fire caused by the use of it. The user should cease to use it when any abnormal issue occurs even during proper and safe use.
3. The part used for this evaluation board/kit or development tool may be changed without any notice.

NOTICE

No part of this material may be reproduced or duplicated in any form or by any means without the written permission of Seiko Epson. Seiko Epson reserves the right to make changes to this material without notice. Seiko Epson does not assume any liability of any kind arising out of any inaccuracies contained in this material or due to its application or use in any product or circuit and, further, there is no representation that this material is applicable to products requiring high level reliability, such as, medical products. Moreover, no license to any intellectual property rights is granted by implication or otherwise, and there is no representation or warranty that anything made in accordance with this material will be free from any patent or copyright infringement of a third party. When exporting the products or technology described in this material, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You are requested not to use, to resell, to export and/or to otherwise dispose of the products (and any technical information furnished, if any) for the development and/or manufacture of weapon of mass destruction or for other military purposes.

All brands or product names mentioned herein are trademarks and/or registered trademarks of their respective companies.

Table of Contents

1. Description	2
1.1 System Requirements.....	2
2. emWin Setup	3
2.1 emWin Package Structure	3
2.2 Target Hardware and Development Environment Setup.....	4
2.3 Building an Existing Project.....	4
2.3.1 IAR EWARM	4
2.3.2 KEIL MDK-ARM (uVision).....	5
3. S1C31 Family emWin Configuration.....	6
3.1 LCDConf.c / LCDConf.h	6
3.1.1 Display Panel Type	6
3.1.2 Color Conversion	6
3.1.3 Display Driver	6
3.2 GUIConf.c / GUIConf.h	6
3.3 Epson_C31.c / Epson_C31.h.....	7
4. Sample Software	8
4.1 c31_demo1	8
4.2 c31_demo2.....	8
4.3 c31_demo3.....	8
4.4 c31_basic_performance	8
4.5 c31_sample	8
5. Supplementary Information for Using emWin with the S1C31 Family	9
5.1 Display Updates.....	9
5.2 2D Library.....	9
5.3 Widgets.....	9
5.4 GUIBuilder.....	11
5.5 Creating Bitmaps.....	11
5.6 Using Existing emWin Sample Code.....	16
Revision History	17

1. Description

This document describes how to use emWin for the S1C31 Family and includes information on sample software operations. emWin is a graphic software and graphical user interface (GUI) product owned by Segger Microcontroller.

The version of emWin included in this package has been customized for use with the S1C31D01.

1.1 System Requirements

The following system environment is required to create and run emWin programs on an S1C31D01:

- S1C31D01 evaluation board
Contact you nearest Epson sales office for information on how to acquire an evaluation board.
- Hardware debugger interface
Recommended is the Segger J-Link Plus.
- IAR Embedded Workbench for ARM
This package has been confirmed with IAR Ver. 8.32.
- Keil MDK for ARM
This package has been confirmed with MDK Ver. 5.25.
- S1C31D01 emWin package (this package)
The version of emWin in this package is Ver. 5.48m.

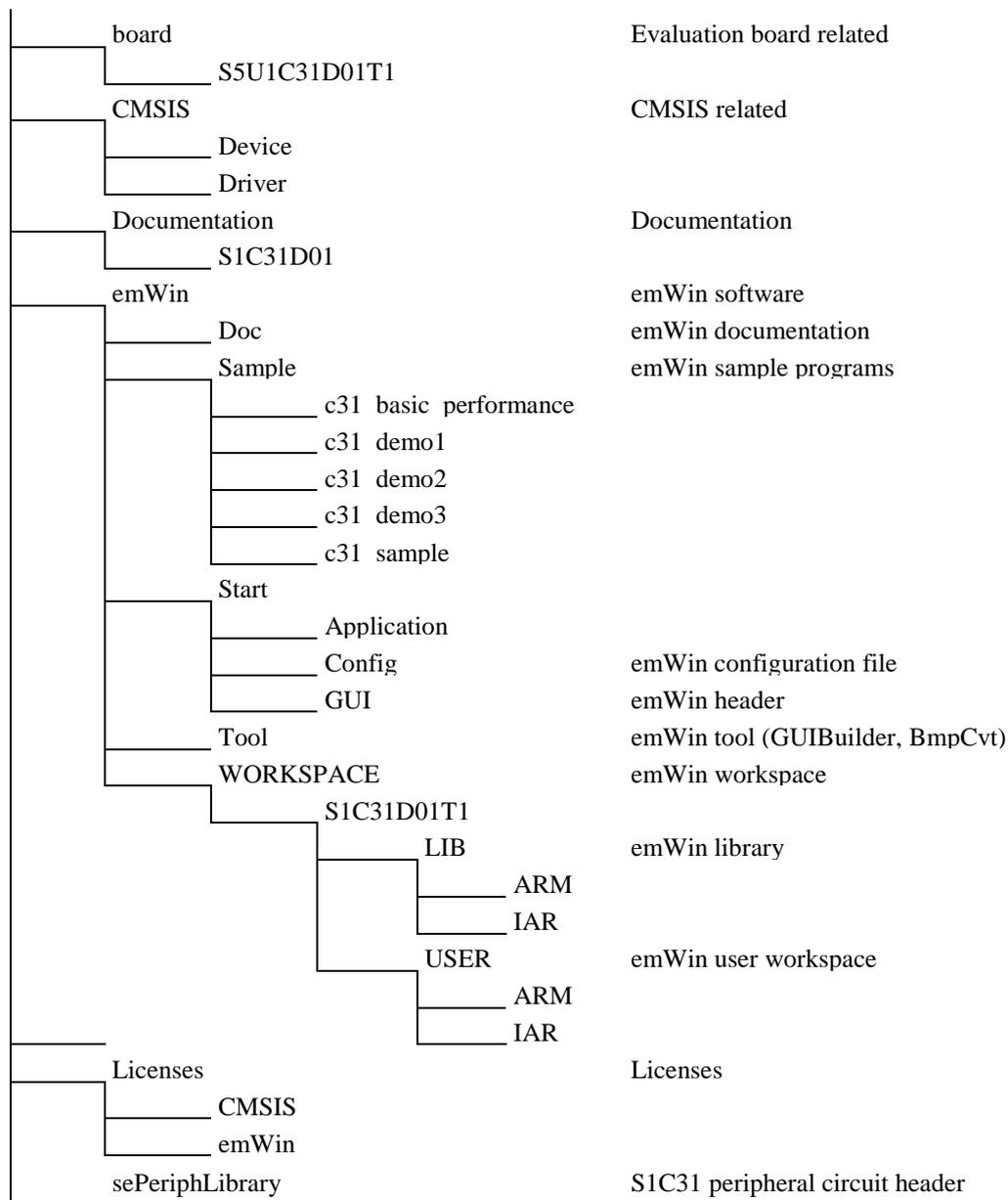
2. emWin Setup

This section describes how to build and debug software using an evaluation board and the emWin package supplied by Epson.

2.1 emWin Package Structure

The directory structure of the emWin package is as follows:

\\SxC31xxx_emWin_ver_x_xx_yymmdd



2.2 Target Hardware and Development Environment Setup

Refer to the following documents for detailed information on how to install the IAR or MDK development environment and connect to the evaluation board:

- S1C31 Family Software Development Setup Guide
- S1C31D01 Sample Software Manual
- S5U1C31D01T1 Manual

2.3 Building an Existing Project

After installing and configuring the hardware and software, build a sample project, load it onto the S1C31D01, and confirm that the system functions correctly. This section describes how to build the “c31_demo1” project.

2.3.1 IAR EWARM

- 1) Open the emWin workspace.

Start the IAR EWARM IDE and click “File” => “Open” => “Workspace”. On the file selection dialog displayed, select the following file within the folder to which the emWin package was decompressed:

```
\\SxC31xx_emWin_ver_x_xx_yymmdd\emWin\WORKSPACE\S1C31D01T1\USER\IAR\emWin.eww
```

- 2) Build peripheral library.

Right-click on the “sePeriphLibrary” project name and select “Set as active”. Next, right-click on the project name and select “Rebuild all” to build.

- 3) Build a sample project.

Right-click on the “c31_demo1” project name and select “Set as active”. Next, right-click on the project name and select “Rebuild all” to build.

As an alternative method for building a project, press F8 after selecting the project, select “Debug - All”, then click “Build all”.

Ensure there are no errors in the “Build” tab of the messages window.

- 4) Download the file to the S1C31D01.

Press Ctrl+D or select “Project” => “Download and Debug”.

When the download is complete, press F5 to start the program.

NOTE: The first time you download a project to the S1C31D01, you will see the following dialog:

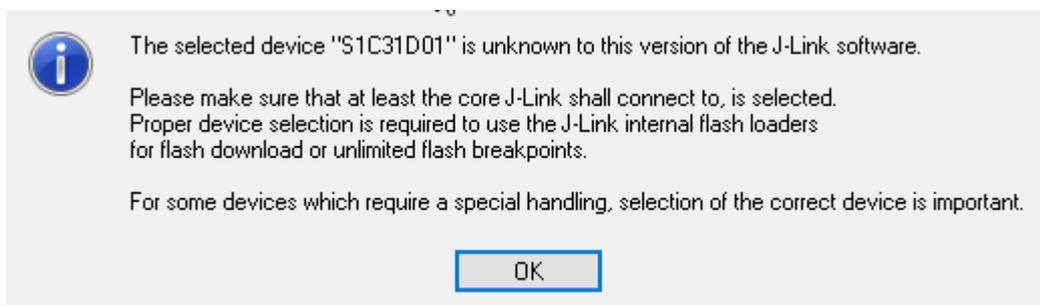


Figure 2.1 Warning dialog with MCU unselected

Click “OK”. In the next dialog, select the “Cortex-M0+” option, as shown.

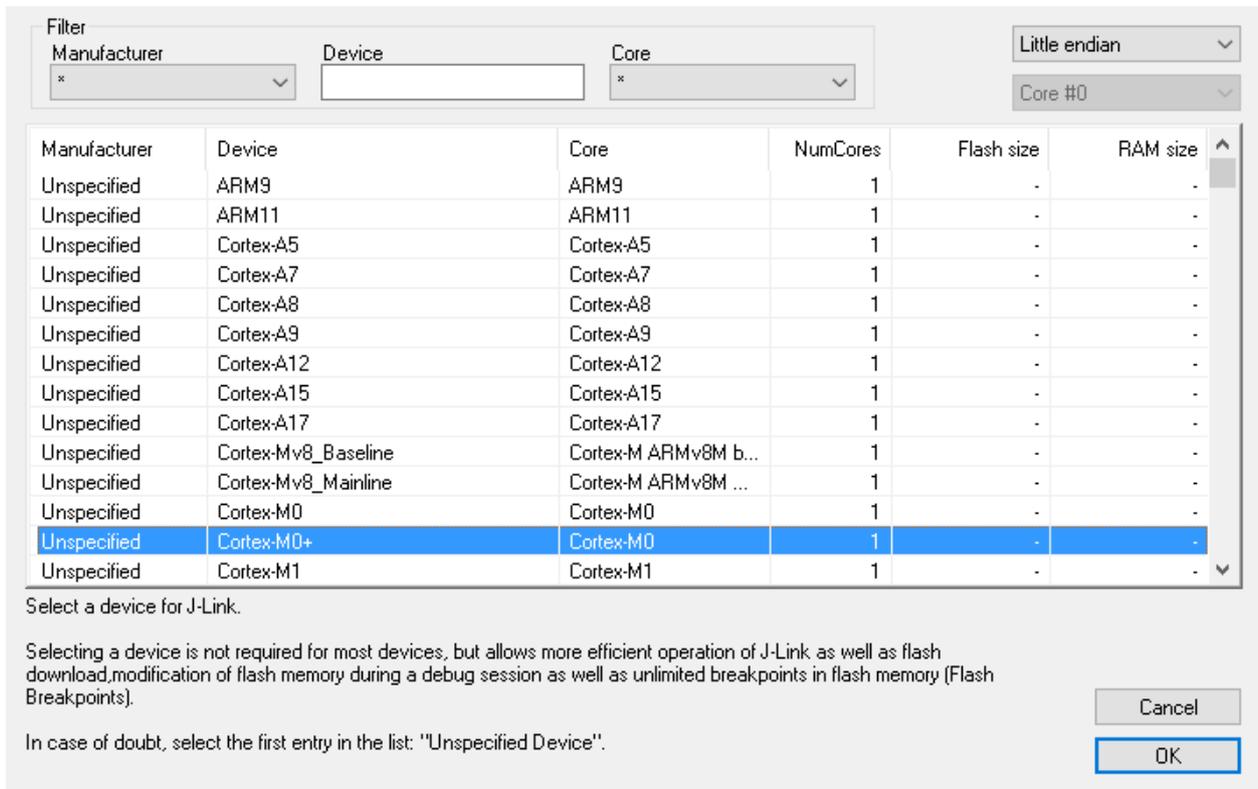


Figure 2.2 MCU selection dialog

- 4) Check the demo program on the display.

The display will vary depending on the demo program selected.

2.3.2 KEIL MDK-ARM (uVision)

- 1) Open the emWin workspace.

Start the Keil MDK and click “Project” => “Open Project”. On the file selection dialog displayed, select the following file within the folder to which the emWin package was decompressed:

\SxC31xx_emWin_ver_x_xx_yymmdd\emWin\WORKSPACE\S1C31D01T1\USER\ARM\emWin.uvmpw

- 2) Build peripheral library.

Right-click on the “sePeriphLibrary” project name and select “Set as Active Project”. Next, click on “Project” => “Rebuild ‘sePeriphLibrary’” to build.

Ensure there are no errors in the “Build Output” tab of the messages window.

- 3) Build a sample project.

Right-click on the “c31_demo1” project name and select “Set as Active Project”. Next, click on “Project” => “Rebuild ‘c31_demo1’” to build.

Ensure there are no errors in the “Build Output” tab of the messages window.

- 4) Download the file to the S1C31D01.

Press Ctrl+F5 or select “Debug” => “Start/Stop Debug Session”.

When the download is complete, press F5 to start the program.

3. S1C31 Family emWin Configuration

This section describes the configuration file when using emWin with the S1C31 Family.

All emWin projects contain a “config” project group. This group contains the following files for either configuring the attached project or providing emWin hardware interface support:

3.1 LCDConf.c / LCDConf.h

The file “LCDConf.h” is where the display panel type, the color conversion method and the display driver are selected.

3.1.1 Display Panel Type

Eight display panel types are currently defined in LCDConf.h. Selecting the panel type configures the panel properties (width, height, color depth) and configures the proper panel initialization sequence.

The following panels are currently supported:

LPM010R030B	- 180x180, 6-bit color, 0.99”, round
LPM012M134B	- 240x240, 6-bit color, 1.19”, round
LPM014T262C	- 205x148, 6-bit color, 1.39”, rectangular
LPM027M128A	- 400x240, SPI, 3-bit color, 2.7”, rectangular
LS010B7DH01	- 128x128, SPI, 1-bit BW, 1.0”, round
LS012B7DH02	- 240x240, SPI, 1-bit BW, 1.2”, round
LS013B4DN04	- 96x96, SPI, 1-bit BW, 1.3”, square
LS013B7DH06	- 128x128, SPI, 3-bit color, 1.33”, square

Select one, and only one, of the panel definitions from the list and leave it uncommented.

3.1.2 Color Conversion

The following color conversion types are available:

<code>COLOR_CONVERSION GUICC_2222</code>	- Supports 6-bit color (RrGgBb) and 2 bits of alpha.
<code>COLOR_CONVERSION GUICC_111</code>	- Supports 1-bit color (monochrome).

The color conversion type is automatically selected to correspond to the panel definition. No changes are needed.

3.1.3 Display Driver

The display driver is selected here. At this time, only the following driver is defined (leave unchanged):

```
GUIDRV_EPSON_MCU
```

3.2 GUIConf.c / GUIConf.h

Refer to the emWin manual for detailed information on the settings used in these files.

3.3 Epson_C31.c / Epson_C31.h

These files are used when GUIDRV_EPSON_MCU is defined in LCDConf.c and do not contain configuration settings.

4. Sample Software

This section describes the sample software included in the emWin package.

4.1 c31_demo1

This sample program demonstrates 2D drawing techniques such as text drawing and polygon use, as well as use of some basic widgets in the form of icons and checkboxes.

4.2 c31_demo2

This sample program demonstrates the use of emWin widgets. Refer to the source code for examples of how to subclass emWin widgets for display updates.

4.3 c31_demo3

This sample program demonstrates various images created using the bin2c.exe and bmpcvt.exe tools.

4.4 c31_basic_performance

This sample program has been ported from the emWin sample program BASIC_Performance project and displays the number of milliseconds required to calculate the number of prime numbers.

4.5 c31_sample

This sample program displays the Epson logo and “Welcome” text.

Use this minimum basic project when creating your own new projects.

5. Supplementary Information for Using emWin with the S1C31 Family

This section provides information to assist developers when using emWin with the S1C31 Family.

5.1 Display Updates

In most graphical systems, the display is updated immediately when new data is written into display memory.

The S1C31 Family operates in a slightly different way. Drawing is performed in a display memory buffer, but the panel is not updated until a panel update sequence is initiated.

The process of updating the display panel is left to the application, allowing updating to be performed at the optimal timing.

Since panel updating is controlled by the application, there is no need for emWin to call memory devices. The device interface draws to a safe area of memory and updates the display once all drawing is complete. The S1C31 Family hardware interface already imposes these conventions on the system.

5.2 2D Library

The emWin 2D library is a collection of drawing routines that can be called at any time. Similarly, the display update routine, `seMDC_PanelUpdate()`, can be called at any time to update the display.

Since the application is in full control of when to upload the panel, there should be no drawing artifacts or incomplete images on the display.

5.3 Widgets

Widgets are predefined graphical items that add controls and visual effects to simplify the creation of a user interface. Widgets include items like buttons, checkboxes, frame windows, knobs, and listboxes.

One widget characteristic is that they are drawn when they receive a `WM_PAINT` message. On most systems, when a widget is drawn, it will appear on the display immediately. On the S1C31 Family display architecture, when a widget, or any other GUI element, is written to the display buffer, it does not instantly appear on the display.

To ensure that widgets are properly updated on the display, programs written for the S1C31 Family must connect to the message chain for each widget used and call the `seMDC_PanelUpdate()` display update routine for the `WM_PAINT` message.

As a programmer, you will need to override the window message handler for most widgets to ensure that they are drawn as expected.

When a widget receives a `WM_PAINT` message, its appearance will be updated as necessary. After the `WM_PAINT` message is received, the system will issue a `WM_POST_PAINT` message.

Overrides are created as shown below.

```
void aFunction( void )
{
    WM_HWIN hCal;
    . . .
    // Create a calendar.
    // Note the callback for the calendar to the _cbCalendar() routine
    hCal = CALENDAR_Create( WM_HBKWIN, 0, 0, 2016, 11, 30, 1, ID_CALENDAR, 0 );
    WM_SetCallback( hCal, _cbCalendar );
    // Show the calendar, pause, then delete the calendar
    WM_InvalidateWindow( hCal );
    GUI_Delay( 5000 );
    WM_DeleteWindow( hCal );
    . . .
}
```

The calendar callback routine passes all messages to the default calendar callback. If the message is WM_POST_PAINT, the panel update routine is called.

```
//-----
// _cbCalendar
// The common dialogs handle most message internally. However updating the
// display requires a panel update sequence and that is where this routine
// comes in.
//
// _cbCalendar connects to the CALENDAR message chain and adds functionality to
// the WM_POST_PAINT message. All other messages are processed by the default
// calendar handler.
//-----
static void _cbCalendar( WM_MESSAGE * pMsg )
{
    GUI_RECT invalidRect;
    CALENDAR_Callback( pMsg );
    // WM_POST_PAINT is sent after WM_PAINT completes.
    // This is an excellent time to update the panel.
    if ( pMsg->MsgId == WM_POST_PAINT ) {
        WM_GetInvalidRect( pMsg->hWin, &invalidRect );
        seMDC_PanelUpdate( invalidRect.y0, invalidRect.y1 );
    }
}
```

The same technique is used to override the update function for all other widgets. The default widget callback will have the form `WIDGETNAME_Callback`. Thus, for the `MULTIEDIT` widget, the default callback is `MULTIEDIT_Callback()`; for the `TREEVIEW` widget, the default callback is `TREEVIEW_Callback()`.

5.4 GUIBuilder

emWin includes a tool called GUIBuilder, which can be used to create a user interface graphically using emWin widgets.

The sample code in the file “demo2_dialog2.c” demonstrates how to use a dialog box generated by GUIBuilder. Using this tool simplifies the creation of a windowed user interface, but it may not always be possible to use the generated code without modifications.

The source code contained in the file “demo2_dialog2.c” was generated by the GUIBuilder tool. All of the code required to make the demo work was added to the areas bracketed by the following comments:

```
// USER START (Optionally insert . . .)
. . .
// USER END
```

The sample contains a frame window, a graph display, and two buttons. Since the sample does not include mouse input, the buttons are not functional. They are simply drawn using the default button code and displayed by the dialog callback routine within the `WM_POST_PAINT` message.

5.5 Creating Bitmaps

If there is no file system for saving and loading bitmaps, images must be created and saved as character arrays.

There are two ways to convert image files into character arrays, each with its pros and cons. The included sample code demonstrates image handling using both methods.

In project “c31_demo2”, the images in `demo2_images.c` were created with `Bin2C.exe`. The first image in the file is the header information for a .BMP file, 0x42 (b) and 0x4D (m). The remainder of the images are derived from .GIF images. A quick inspection of the character arrays shows the characteristic 0x47, 0x49, and 0x46 (GIF) of a .GIF image.

In project “c31_demo3”, the Epson logo image in the file “epsonlogo.c” was created using the `BmpCvt` program. The data in the image array does not contain any file header information.

Bin2C

`Bin2C.exe` is available for download from the Segger website.

(https://www.segger.com/downloads/free_tools)

The program “`Bin2C.exe`” converts any binary file to a character array. It can be used to convert a binary format image file (.BMP, .JPG, .GIF, .PNG) to a character array.

The advantage of directly converting binary images to character arrays is that the image will retain the same characteristics of the image file: for example, if the image file is compressed, the size of the converted file is also smaller.

This advantage is offset by the requirement of including routines to decode the image. With lossy compression image formats (e.g., .JPG files), it may be difficult to match the decoded image color to an expected color.

The procedure for creating an image using Bin2C is as follows:

- 1) Open the application.

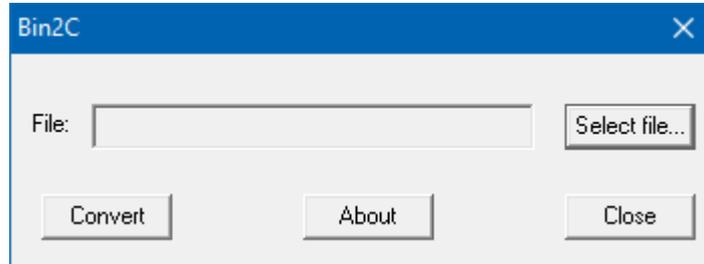


Figure 5.1 Bin2C launch window

- 2) Click “Select file...” and browse to the folder containing the file you wish to convert, select the file, and click “Open”.

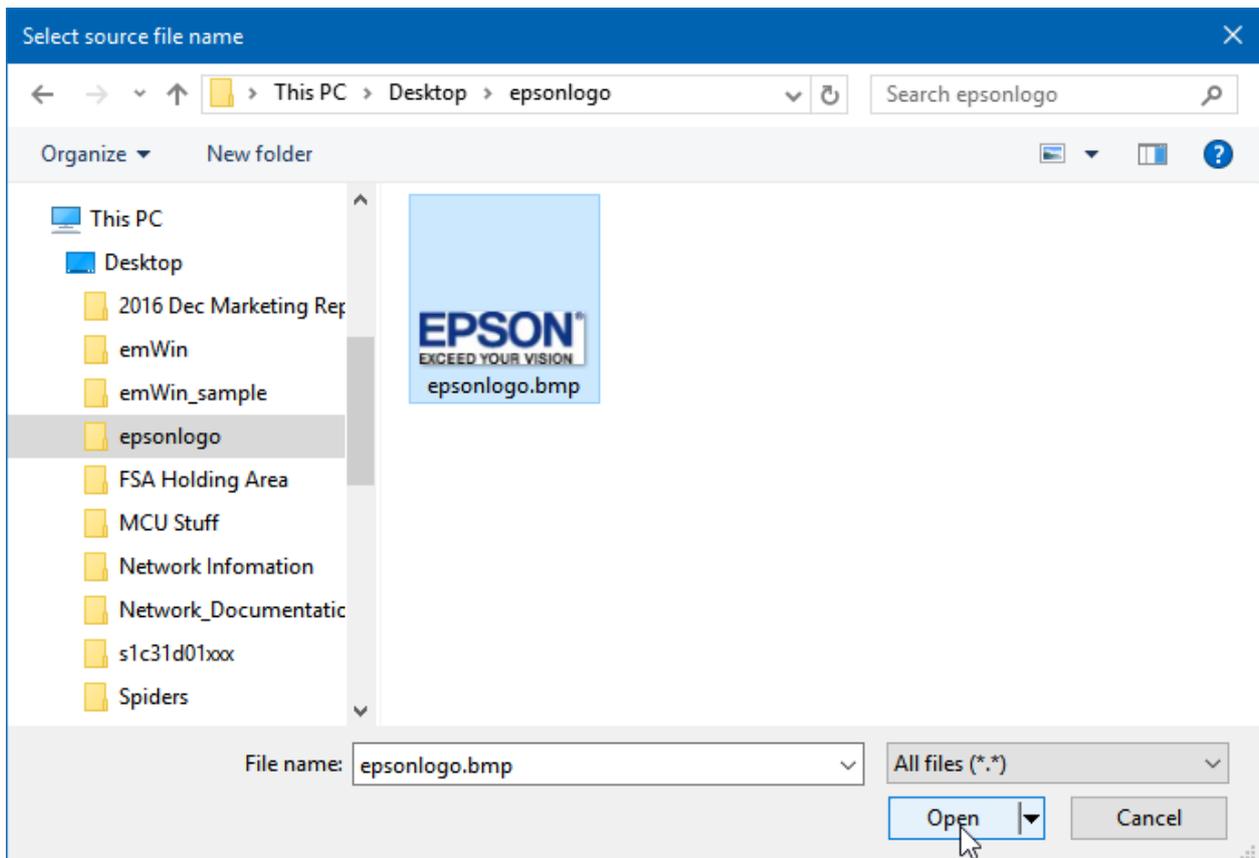


Figure 5.2 File selection dialog

-
- 3) Click “Convert”. An ASCII representation of the file will be saved into the same folder as the binary image.

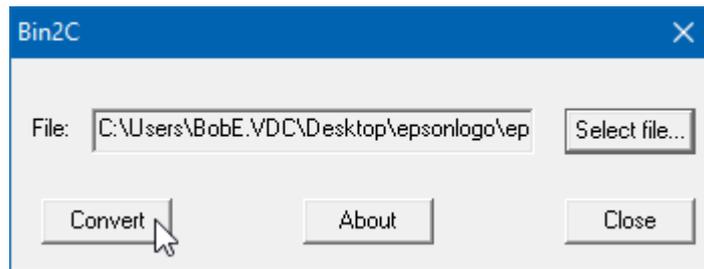


Figure 5.3 Image conversion dialog

“BmpCvt”

Images created with BmpCvt.exe consist of a color palette and an array of values representing each pixel in the image. Images created in this way may be slightly larger than images created with Bin2C.exe, but since there is no code space for the image decode routines and there is no image decode overhead, images should load slightly faster.

The procedure for creating a bitmap image for use in emWin applications is as follows:

- 1) Start by creating an image and reducing the color depth to 8 bits per pixel.



Refer to the “epsonlogo.bmp” image in the

“\SxC31xx-emWin_ver_x_xx\emWin\Sample\c31_demo3\artwork” folder.

- 2) Launch the “BmpCvt.exe” program and drag the image onto the BmpCvt window.

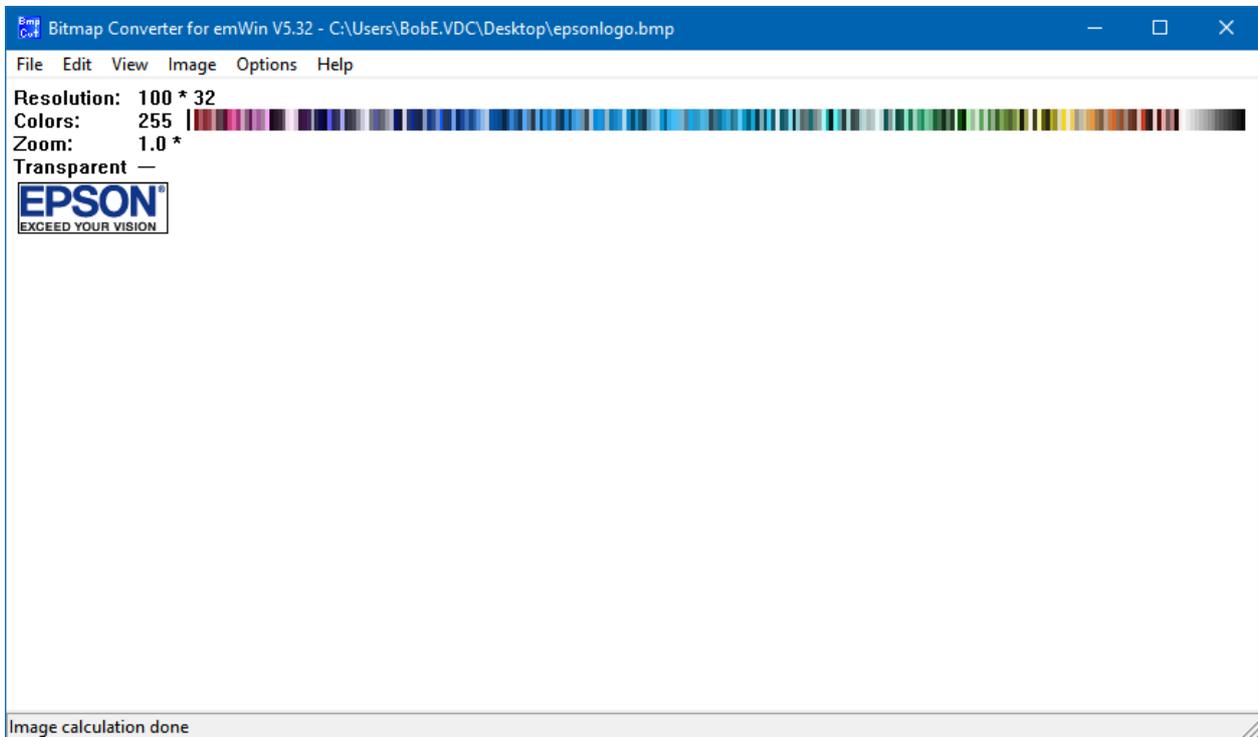


Figure 5.4 Image file importing

3) Convert the image to 6-bit color using the menu “Image” => “Convert to” => “6 bit color 222”.

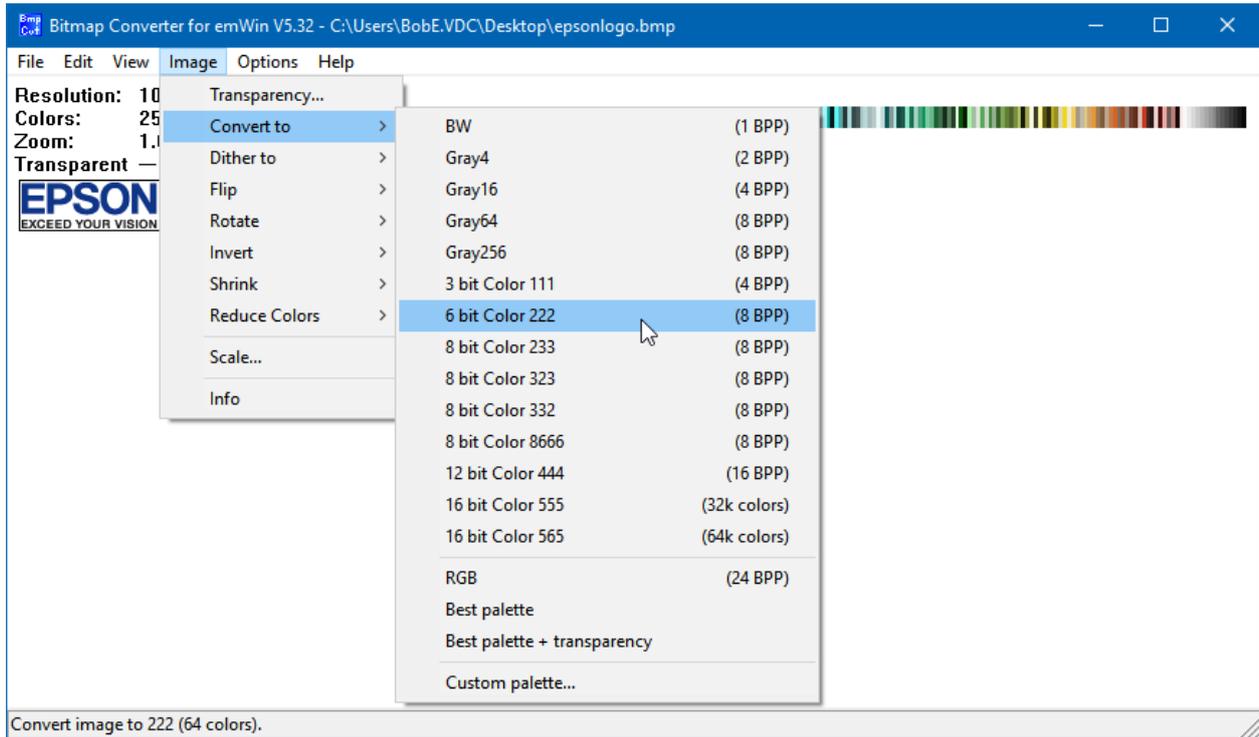


Figure 5.5 Image color reduction

4) Select “File” => “Save As...” to save the image as a character array.

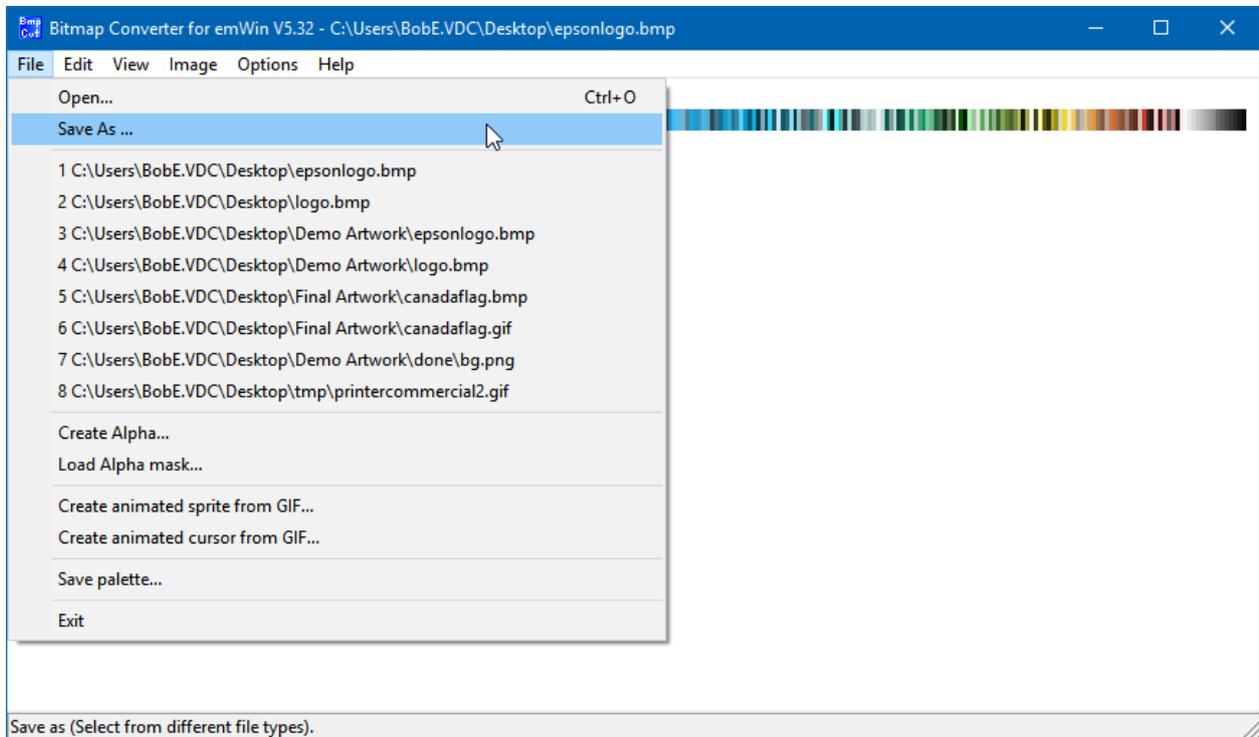


Figure 5.6 Image saving

- 5) Set “Save as type:” to ““C” bitmap file (*.c)”, and enter the desired name for the destination file. Click “Save” to save the image file.

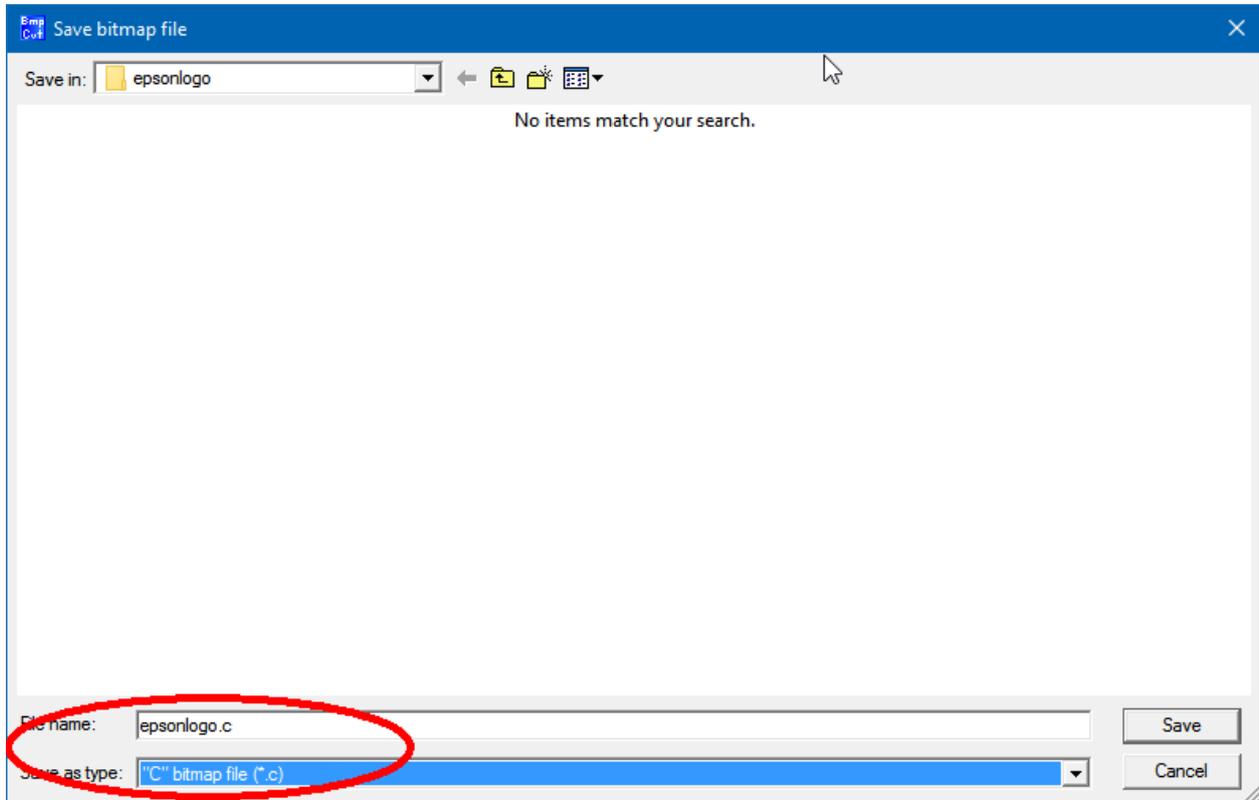


Figure 5.7 Image saving (save format selection)

5.6 Using Existing emWin Sample Code

Segger provides a large collection of sample code for the emWin GUI. Most of the sample code will work on the S1C31D01 platform, but tweaks and changes may be required in some cases.

The Segger sample code is written for use with a Windows-based emulator. This adds an extra layer of overhead that must be removed for use on the S1C31D01.

America

Epson America, Inc.

Headquarter:
3840 Kilroy Airport Way
Long Beach, California 90806-2452 USA
Phone: +1-562-290-4677

San Jose Office:
214 Devcon Drive
San Jose, CA 95112 USA
Phone: +1-800-228-3964 or +1-408-922-0200

Europe

Epson Europe Electronics GmbH

Riesstrasse 15, 80992 Munich,
Germany
Phone: +49-89-14005-0 FAX: +49-89-14005-110

Asia

Epson (China) Co., Ltd.

4th Floor, Tower 1 of China Central Place, 81 Jianguo Road, Chaoyang
District, Beijing 100025 China
Phone: +86-10-8522-1199 FAX: +86-10-8522-1120

Shanghai Branch

Room 1701 & 1704, 17 Floor, Greenland Center II,
562 Dong An Road, Xu Hui District, Shanghai, China
Phone: +86-21-5330-4888 FAX: +86-21-5423-4677

Shenzhen Branch

Room 804-805, 8 Floor, Tower 2, Ali Center, No.3331
Keyuan South RD(Shenzhen bay), Nanshan District, Shenzhen
518054, China
Phone: +86-10-3299-0588 FAX: +86-10-3299-0560

Epson Taiwan Technology & Trading Ltd.

15F, No.100, Songren Rd, Sinyi Dist, Taipei City 110. Taiwan
Phone: +886-2-8786-6688

Epson Singapore Pte., Ltd.

1 HarbourFront Place,
#03-02 HarbourFront Tower One, Singapore 098633
Phone: +65-6586-5500 FAX: +65-6271-3182

Seiko Epson Corp.

Korea Office

19F, KLI 63 Bldg, 60 Yoido-dong,
Youngdeungpo-Ku, Seoul 150-763, Korea
Phone: +82-2-784-6027 FAX: +82-2-767-3677

Seiko Epson Corp.

Sales & Marketing Division

Device Sales & Marketing Department

421-8, Hino, Hino-shi, Tokyo 191-8501, Japan
Phone: +81-42-587-5816 FAX: +81-42-587-5116