FAQ・テクニカルガイド

emWin ストリームビットマップ

作成・使用方法

● ストリームビットマップに変換

Bitmap Converter ツールを起動して画像ファイル (PNG、BMP、JPEG 等)をロードします。



「File → Save as..」メニューコマンドでイメージファイルをストリームビットマップ「*.dta」ファイルとして保存します。



Save file as	
 空理 ▼ 新しいフォルダー 	
RX65N-emWin Barrow Min_RX65N Barrow Min_RX65N Sample1 Tool Simulation CodeBlocks VS2017 Barrow Min	Car_400x205.dt a
ファイル名(N): Printer_200x200_ARGB.dta ファイルの種類(I): DTA files (*.dta)	・ ・ 保存(S) キャンセル

出力フォーマットを選択して、ファイルを作成します。





● ストリームビットマップをC言語ファイルに変換

「emWin→Tool」フォルダ下の「Bin2C. exe」ツールを起動します。

	Bin2C		×
BIN 2C	File:		Select file]
	Convert	About	Close
Bin2C.exe			

「Select File」ボタンを押してストリームビットマップDTA(*.dta)ファイルを指定します。

Bin2C	
File: C:\SEGGER_DEMO\emWin\RX65N-emWin'	Select file
Convert	Close

「Convert」ボタンを押して C 言語ファイルを作成します。DTA ファイルと同じファイル名の(*.c) ファイルを 確認して「Bin2C. exe」ツールを終了します。

бб С		
Printer_200x20	Printer_200x20	Printer_200x20
0_ARGB.c	0_ARGB.dta	0_ARGB.png

	/*↓	
2	C-file generated by Bin2C↓	
- 3	Compiled: Jun 11 2019 at 16:50:55↓	
- 4	1	
- 5	Copyright (C) 2018↓	
6	Segger Microcontroller GmbH↓	
- 7	www.segger.com↓	
8		
9	The Embedded Experts↓	
10	*/↓	
11		
12	static const unsigned char _acPrinter_200x200_ARGB[1600160L + 1] = {↓	
13	0x42, 0x40, 0x10, 0x00, 0x08, 0x00, 0x68, 0x00, 0x20, 0x03, 0x20, 0x00, 0x00, 0x00, 0x00	, 0
	OXBF, OXED, OXFF, OX63, OXBF, OXFF, ↓	
14	0X63, 0X8F, 0XED, 0XFF, 0X63, 0X8F, 0XED, 0XFF, 0X63, 0X8F, 0XED, 0XFF, 0X63, 0X8F, 0XED	, 0
	UXEF, UXEU, UXFF, UXES, UXEF, UXEF, ↓	~
15	L UX63, UXBF, UXED, UXFF, UX63, UXBF, UXED, UXFF, UX63, UXBF, UXED, UXFF, UX63, UXBF, UXED	, 0.



● ストリームビットマップを描画

「Bin2C. exe」ツールで変換しましたイメージデータファイル(C 言語文字列)を GUI プロジェクトに追加しま す。「static const」式の文字列ですので、C 言語ファイルとしてプロジェクトに追加する場合は、以下のよう に「static const」→「const」に変更してください。

DTAから変換したファイル: Printer_200x200_ARGB.c

const unsigned char _acPrinter_200x200_ARGB[120016UL + 1] = {

アプリケーションファイル:

extern const unsigned char _acPrinter_200x200_ARGB[120016UL + 1];

別の方法としては、DTA から作成した C 言語ファイルをヘッダファイル(*.h) に変更して GUI アプリケーションにインクルード可能です。

アプリケーションファイル:

#include "Printer_200x200_ARGB.h"

ストリームビットマップイメージの描画は GUI_DrawStreamedBitmapExAuto() 関数を使用します。

void GUI_DrawStreamedBitmapAuto(const void * p, int x, int y);

アプリケーション事例:

GUI_SetBkColor(GUI_RED); GUI_Clear(); GUI_DrawStreamedBitmapAuto(_acCar_400x205, 20, 20); GUI_DrawStreamedBitmapAuto(_acPrinter_200x200_ARGB, 180, 60);





直接メモリアクセス出来ない QSPI フラッシュ、SD カード、EEPROM 等の外部 ROM メディアにイメージデータが保存されている場合は、GUI_DrawStreamedBitmapExAuto() 関数を使用します。

```
アプリケーション事例:
```

```
static int _GetData(void * p, const U8 ** ppData, unsigned NumBytes, U32 Off) {
    // 外部 SD カード/QSPI フラッシュから Off オフセットの NumBytes サイズのデータブロックを
    // 読み出して指定バッファ (*ppData) に保存します。
    .....
    return NumBytes;
}
void MainTask(void) {
    GUI_Init();
    GUI_DrawStreamedBitmapExAuto(_GetData, _acCar_400x205, 20, 20);
    GUI_DrawStreamedBitmapExAuto(_GetData, _acPrinter_200x200_ARGB, 180, 60);
    .....
}
```

ストリームビットマップのフォーマットに合わせて以下の関数も使用可能です。

関数	フォーマット
GUI_DrawStreamedBitmapA555Ex()	A555:16bpp の high color ビットマップ、5 ビットの青、5 ビットの 緑、5 ビットの赤、8 ビットのアルファチャンネル
GUI_DrawStreamedBitmapAM555Ex()	AM555:16bpp の high color ビットマップ、5 ビットの赤、5 ビットの 緑、5 ビットの青、8 ビットのアルファチャンネル
GUI_DrawStreamedBitmapA565Ex()	A565:16bpp の high color ビットマップ、5 ビットの青、6 ビットの 緑、5 ビットの赤、8 ビットのアルファチャンネル
GUI_DrawStreamedBitmapAM565Ex()	AM565:16bppのhigh color ビットマップ、5 ビットの赤、5 ビットの 緑、6 ビットの青、8 ビットのアルファチャンネル
GUI_DrawStreamedBitmap555Ex()	555: 16bpp の high color ビットマップ、5 ビットの赤、5 ビットの緑、 5 ビットの青
GUI_DrawStreamedBitmapM555Ex()	M555:16bppのhigh colorビットマップ、5ビットの青、5ビットの 緑、5ビットの赤
GUI_DrawStreamedBitmap565Ex()	565: 16bpp の high color ビットマップ、5 ビットの青、5 ビットの緑、 5 ビットの赤
GUI_DrawStreamedBitmapM565Ex()	M565: 16bpp の high color ビットマップ、5 ビットの赤、5 ビットの 緑、5 ビットの青
GUI_DrawStreamedBitmap24Ex()	24: 24bpp の true color ビットマップ、8 ビットの青、8 ビットの緑、 8 ビットの赤



IMAGE ウィジェットとしてストリームビットマップイメージを描画する場合は、は IMAGE_SetDTAEx() 関数を使用します。

void IMAGE_SetDTAEx(IMAGE_Handle hObj, GUI_GET_DATA_FUNC * pfGetData, void * pVoid);

```
アプリケーション事例:
```

```
#include "DIALOG.h"
                      (GUI_ID_USER + 0x00)
#define ID_IMAGE_0
#define ID_IMAGE_1
                       (GUI_ID_USER + 0x01)
static int _GetData(void * p, const U8 ** ppData, unsigned NumBytes, U32 Off) {
 // 外部 QSPI フラッシュから Off オフセットの NumBytes サイズのデータブロックを
 // 読み出して指定バッファに保存します。
 return NumBytes;
}
void MainTask(void) {
 IMAGE_Handle hImg1;
 IMAGE_Handle hImg2;
 hImg1 = IMAGE_CreateEx(20, 20, 400, 205, WM_HBKWIN, WM_CF_SHOW | WM_CF_HASTRANS, 0, ID_IMAGE_0);
 hImg2 = IMAGE_CreateEx(180, 60, 200, 200, WM_HBKWIN, WM_CF_SHOW | WM_CF_HASTRANS, 0, ID_IMAGE_1);
 IMAGE_SetDTAEx(hImg1, _GetData, _acCar_400x205);
 IMAGE_SetDTAEx(hImg2, _GetData, _acPrinter_200x200_ARGB);
 GUI_Delay(10);
 .....
}
```



代理店販売(デバッガ、RTOS、GUI、ミドルウエア) https://www.embitek.co.jp

日本代理店

SEGGER Microcontroller GmbH



〒130-0021 東京都墨田区緑 4-8-8 中井ビル 4F Phone: 03-6240-2655 Fax: 03-6240-2656

本資料に記載の全ての情報の使用に起因する損害、第三者の知的財産権、権利またはその他の経 費に対して、SEGGER 社及び株式会社エンビテックは一切責任を負いません。 本資料の内容は予告なく変更されることがあります。

商標

「EmblTeK」、EmblTeK ロゴは株式会社エンビテックの商標または登録商標です。 その他、本資料に記載しているプロセッサ名、ツール名および製品名は、それぞれ各社の商標または登録 商標です。

