

1. 08a_lcd

2024/7/18 Table of Contents

08a_lcd

目的

構成データ

LCD制御

例題 lcdclear

課題1 colorbar

課題2 dispbmp

1.1. 目的

組込みアプリケーション開発 08a_lcd

1.2. 構成データ

1.2.1. /media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice ディレクトリ

▼ …/share/ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice/ の構成

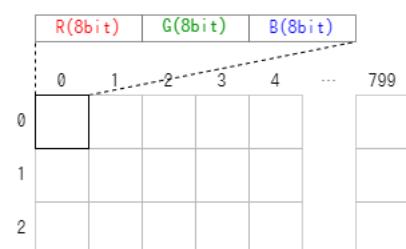
```
1 user@1204PC-Z490M:/mnt/v/VirtualBoxWork/share/ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice$ tr
2 ./
3   └─ 08_lcd/
4     └─ bitmap/
5       └─ 1-sea.bmp*           <---- 課題2用ビットマップファイル
6       └─ 2-park.bmp*          <---- 課題2用ビットマップファイル
7       └─ 3-church.bmp*         <---- 課題2用ビットマップファイル
8       └─ 4-factory.bmp*        <---- 課題2用ビットマップファイル
9       └─ colorbar.c*           <---- 課題1 デバイス制御用ソース
10      └─ dispbmp.c*            <---- 課題2 デバイス制御用ソース
11      └─ lcdclear.c*           <---- 例題 デバイス制御用ソース
12      └─ Makefile*              <---- デバイス制御用Makefile
13      └─ primcol.rgb*           <---- 24ビット非圧縮ビットマップ画像ファイル
```

1.3. LCD制御

1.3.1. デバイス仕様

LCDのハードウェア仕様

- 横800 X 縦480 pixel
- 1pixel 24bit RGB88形式



- フレームバッファデバイスはファイルではなくメモリ
- システムには画像データを格納するためのフレームバッファと呼ばれるメモリ領域がある
- LCD上の各pixelはフレームバッファの中にカラーデータが格納
- カラーデータをRGB88形式でリトルエンディアンの形でフレームバッファに書き込むと LCD 上の対応するpixel を点灯
- プログラムの中でフレームバッファのアドレスを取得するには mmap システムコール

1.3.2. デバイスファイル

"/dev/fb0/"

primcol.rgb を ArmadilloX1へコピー

▼ sudo cp primcol.rgb /work/linux/nfsroot/debug/04_practice/

```
1 | atmark@atde8:/media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/08a.lcd$ sudo cp primcol.rgb /work/linux/nfsroot/debug/04_practice/
```

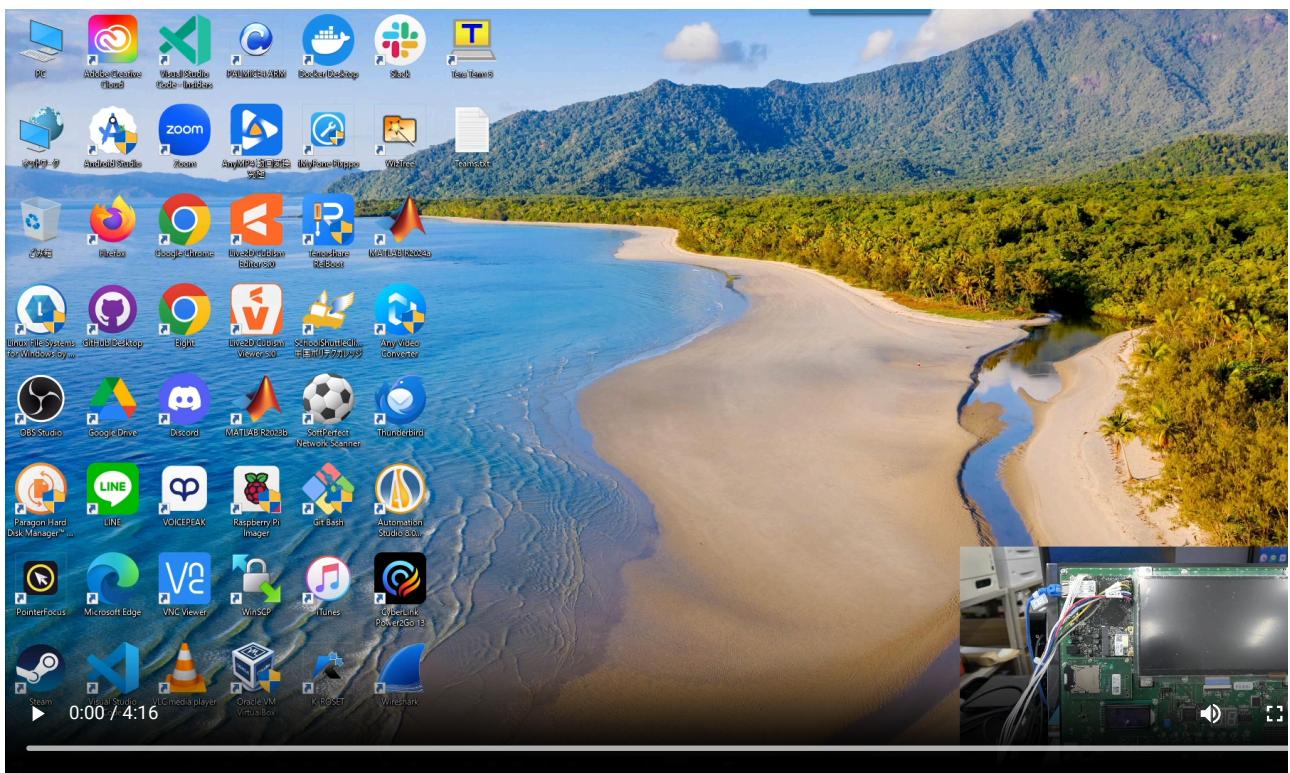
▼ フレームバッファによる LCD 描画

```
1 | root@armadillo:/debug/04_practice# cat primcol.rgb > /dev/fb0
```

1.3.3. 実行している様子

▼ フレームバッファによるLCDへの描画している動画

<https://youtu.be/cgvoJdAWhgY>



1.4. 例題 lcdclear

1.4.1. ソース

lcdclear.c

▼ 08a.lcd/lcdclear.c

```

1 #include <stdio.h>
2 #include <fcntl.h>
3 #include <sys/mman.h>
4 #include <string.h>
5 #include <unistd.h>
6
7 #define SCREENWIDTH          800
8 #define SCREENHEIGHT         480
9 #define BYTES_PER_PIXCEL    4
10 #define SCREENSIZE          (SCREENWIDTH * SCREENHEIGHT * BYTES_PER_PIXCEL)
11
12 int main(void)
13 {
14     int fd;
15     unsigned int *pfb;
16
17     // フレームバッファをオープンします。
18     // オープンに失敗した場合はエラーで終了します。
19     if ((fd = open("/dev/fb0", O_RDWR)) < 0) {
20         perror("open(fb)");
21         return 1;
22     }
23
24     /* mmapによりバッファの先頭アドレスを取得します。 */
25     pfb = mmap(0, SCREENSIZE, PROT_READ|PROT_WRITE, MAP_SHARED, fd, 0);
26     // 取得に失敗した場合はエラーで終了します。
27     if (pfb == MAP_FAILED){
28         perror("mmap");
29         return 1;
30     }
31
32     // LCDの画面表示を全て消去するために、
33     // 取得したアドレスから確保領域を全て0で初期化します。
34     memset(pfb, 0, SCREENSIZE);
35
36     // フレームバッファのために確保した領域を開放します。
37     munmap(pfb, SCREENSIZE);
38     // フレームバッファをクローズします。
39     close(fd);
40
41     return 0;
42 }

```

Makefile

⚠ \$(CFLAGS) と cp -p ./*.bmp を追加している

▼ 08a_lcd/Makefile

```

1 CC = arm-linux-gnueabihf-gcc
2 #TARGET = lcdclear colorbar dispbmp
3 TARGET = lcdclear
4 CFLAGS = -gdwarf-2 -O0
5
6 all: $(TARGET)
7
8 lcdclear: lcdclear.c
9     $(CC) $(CFLAGS) -o $@ $<
10
11 colorbar: colorbar.c
12     $(CC) $(CFLAGS) -o $@ $<
13
14 dispbmp: dispbmp.c
15     $(CC) $(CFLAGS) -o $@ $<
16
17 install :
18     cp -p $(TARGET) /work/linux/nfsroot/debug/04_practice
19     cp -p $(TARGET) /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice
20     cp -p $(TARGET).c /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice
21     cp -p bitmap/*.bmp /work/linux/nfsroot/debug/04_practice
22     cp -p bitmap/*.bmp /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice
23
24 clean:
25     rm -f $(TARGET)
26
27 .PHONY: clean

```

1.4.2. 動作確認

make clean

▼ \$ make clean

```

1 | atmark@atde8:/media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/08a_lcd$ make clean
2 | rm -f lcdclear

```

make

▼ \$ make

```

1 | atmark@atde8:/media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/08a_lcd$ make lcd
2 | arm-linux-gnueabihf-gcc -gdwarf-2 -O0 -o lcdclear lcdclear.c

```

sudo make install

▼ \$ sudo make install

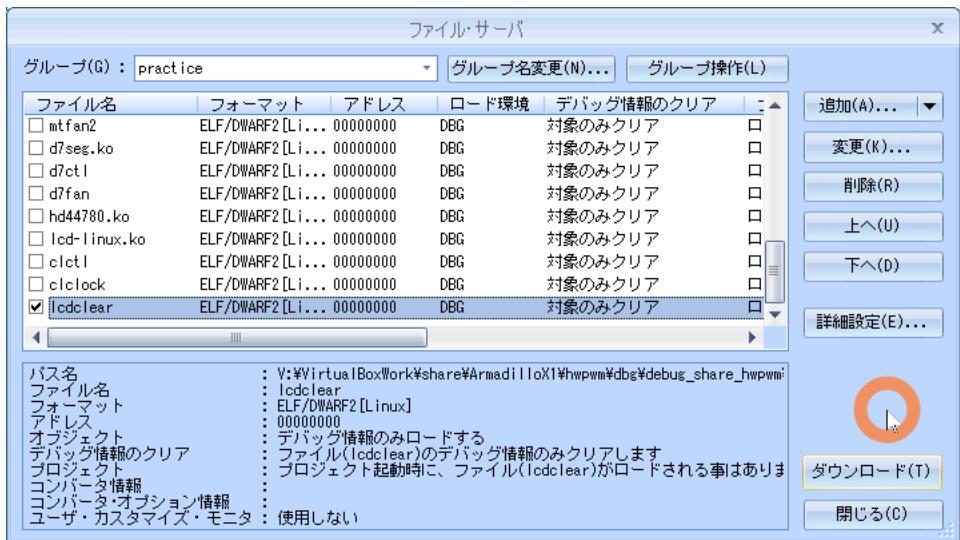
```

1 | atmark@atde8:/media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/08a_lcd$ sudo make
2 | [sudo] atmark のパスワード:
3 | cp -p lcdclear /work/linux/nfsroot/debug/04_practice
4 | cp -p lcdclear /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice
5 | cp -p lcdclear.c /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice
6 | cp -p bitmap/*.bmp /work/linux/nfsroot/debug/04_practice
7 | cp -p bitmap/*.bmp /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice

```

CSIDEでロード

▼ メニュー「ファイル」 - 「ロード」



実行結果

▼ root@armadillo:/debug/04_practice# ./lcdclear

1 | root@armadillo:/debug/04_practice# ./lcdclear

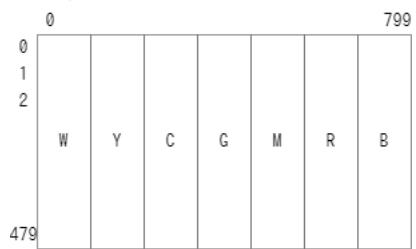
実行している様子

▼ lcdclear を実行している動画

<https://youtu.be/uC1v1coHhP8>



1.5. 課題1 colorbar



1.5.1. ソース

colorbar.c

▼ 08a.lcd/colorbar.c

```

1 #include <stdio.h>
2 #include <fcntl.h>
3 #include <sys/mman.h>
4 #include <unistd.h>
5
6 #define SCREENWIDTH           800
7 #define SCREENHEIGHT          480
8 #define BYTES_PER_PIXCEL      4
9 #define SCREENSIZE            (SCREENWIDTH * SCREENHEIGHT * BYTES_PER_PIXCEL)
10 #define RGB888(r, g, b)        (((r) & 0xff) << 16 | \
11                                         ((g) & 0xff) << 8 | \
12                                         ((b) & 0xff))
13
14 #define WHITE                RGB888(0xff, 0xff, 0xff)
15 #define YELLOW               RGB888(0xff, 0xff, 0x00)
16 #define CYAN                 RGB888(0x00, 0xff, 0xff)
17 #define GREEN                RGB888(0x00, 0xff, 0x00)
18 #define MAGENTA              RGB888(0xff, 0x00, 0xff)
19 #define RED                 RGB888(0xff, 0x00, 0x00)
20 #define BLUE                RGB888(0x00, 0x00, 0xff)
21
22 // LCDに画像を描画する関数。
23 void fill_rect(unsigned int *pf, int x0, int y0, int w, int h, unsigned int color)
24 {
25
26
27
28
29
30
31
32
33
34
35
36 }
37
38 int main(void)
39 {
40
41
42     // フレームバッファをオープンします。
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64

```

```
65
66
67 // フレームバッファのために確保した領域を開放します。
68
69 // フレームバッファをクローズします。
70
71
72     return 0;
73 }
```

Makefile

▼ 08a_lcd/Makefile

```
1 CC = arm-linux-gnueabihf-gcc
2 #TARGET = lcdclear colorbar dispbmp
3 TARGET = colorbar
4 CFLAGS = -gdwarf-2 -O0
5
6 all: $(TARGET)
7
8 lcdclear: lcdclear.c
9     $(CC) $(CFLAGS) -o $@ $<
10
11 colorbar: colorbar.c
12     $(CC) $(CFLAGS) -o $@ $<
13
14 dispbmp: dispbmp.c
15     $(CC) $(CFLAGS) -o $@ $<
16
17 install :
18     cp -p $(TARGET) /work/linux/nfsroot/debug/04_practice
19     cp -p $(TARGET) /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice
20     cp -p $(TARGET).c /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice
21     cp -p bitmap/*.bmp /work/linux/nfsroot/debug/04_practice
22     cp -p bitmap/*.bmp /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice
23
24 clean:
25     rm -f $(TARGET)
26
27 .PHONY: clean
```

1.5.2. 動作確認

make clean

▼ \$ make clean

```
1 atmark@atde8:/media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/08a_lcd$ make clean
2 rm -f colorbar
```

make

▼ \$ make

```
1 atmark@atde8:/media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/08a_lcd$ make colorbar
2 arm-linux-gnueabihf-gcc -gdwarf-2 -O0 -o colorbar colorbar.c
```

sudo make install

▼ \$ sudo make install

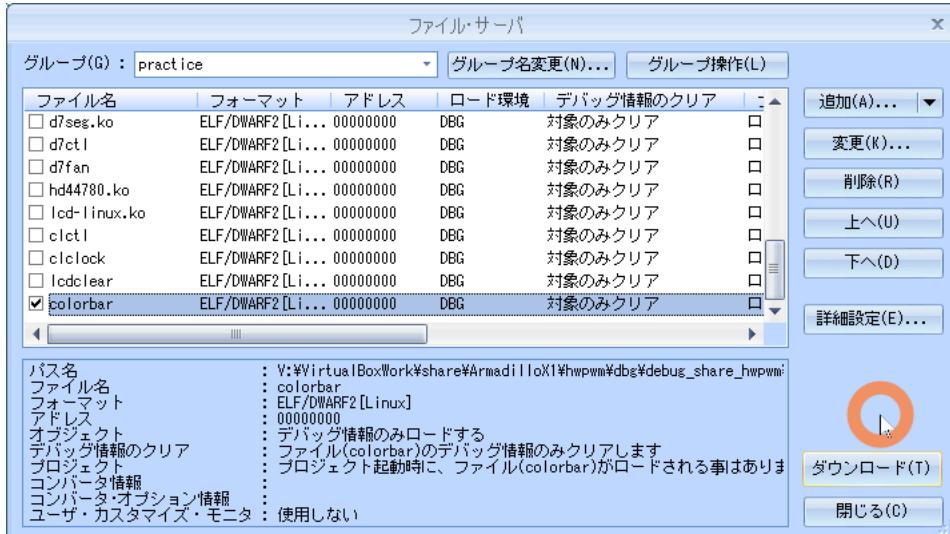
```

1 atmark@atde8:/media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/08a_lcd$ sudo make
2 cp -p colorbar /work/linux/nfsroot/debug/04_practice
3 cp -p colorbar /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice
4 cp -p colorbar.c /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice
5 cp -p bitmap/*.bmp /work/linux/nfsroot/debug/04_practice
6 cp -p bitmap/*.bmp /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice

```

CSIDEでロード

▼ メニュー「ファイル」 - 「ロード」



実行

▼ root@armadillo:/debug/04_practice# ./colorbar

```
1 | root@armadillo:/debug/04_practice# ./colorbar
```

実行している様子

▼

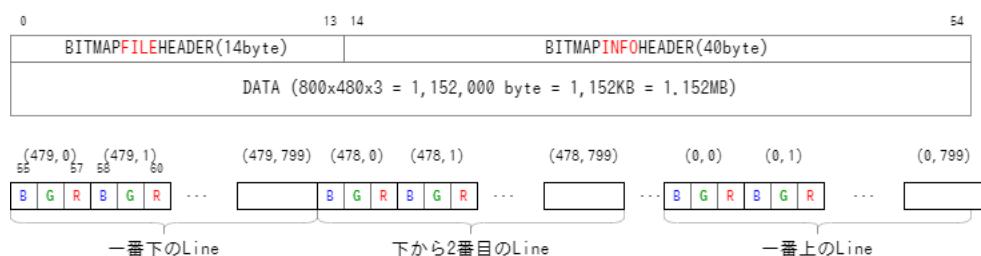
<https://youtu.be/bwBskFgUOkg>



1.6. 課題2 dispbmp

- *.bmp を LCD へ表示
 - 読み込んだ画像が LCD よりも小さい時は LCD 中央に表示
 - 反対に LCD よりも大きな画像の場合はエラーメッセージを出力してプログラムを終了

24ビット非圧縮ビットマップ画像ファイルフォーマット



ビットマップヘッダー (BIMAPFILEHEADER & BITMAPINFOHEADER)

▼

```

1  typedef struct tagBITMAPFILEHEADER{
2      unsigned short bfType;           // ビットマップファイルヘッダ
3      unsigned long  bfSize;          // 識別子0x4d42('B','M')
4      unsigned short bfReserved1;     // ファイルサイズ
5      unsigned short bfReserved2;     // 使わない
6      unsigned long  bfOffBits;       // 使わない
7  } __attribute__((packed)) BITMAPFILEHEADER;
8
9  typedef struct tagBITMAPINFOHEADER{
10     unsigned long   biSize;          // ビットマップ情報ヘッダ
11     long            biWidth;         // 情報ヘッダサイズ
12     long            biHeight;        // 画像の幅
13     unsigned short biPlanes;        // 画像の高さ
14     unsigned short biBitCount;      // プレーン数（1に固定）
15     unsigned long   biCompression;   // 1ピクセルあたりのビット数
16     unsigned long   biSizeImage;    // 圧縮タイプ
17     long            biXPixPerMeter;  // 画像データサイズ
18     long            biYPixPerMeter;  // 横1mあたりのピクセル数
19     unsigned long   biClrUsed;      // 縦1mあたりのピクセル数
20     unsigned long   biClrImportant; // パレット数
21 } __attribute__((packed)) BITMAPINFOHEADER;
22

```

1.6.1. ソース

dispbmp.c

▼ 08a_lcd/dispbmp.c

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <fcntl.h>
4 #include <sys/mman.h>
5 #include <string.h>
6 #include <unistd.h>
7
8 #define SCREENWIDTH          800
9 #define SCREENHEIGHT         480
10 #define BYTES_PER_PIXCEL    4
11 #define SCREENSIZE           (SCREENWIDTH * SCREENHEIGHT * BYTES_PER_PIXCEL)
12 #define RGB888(r, g, b)      (((r) & 0xff) << 16 | \
13                                ((g) & 0xff) << 8 | \
14                                ((b) & 0xff))
15
16 typedef struct tagBITMAPFILEHEADER{
17     unsigned short bfType;           // ビットマップファイルヘッダ
18     unsigned long  bfSize;          // 識別子0x4d42('B','M')
19     unsigned short bfReserved1;     // ファイルサイズ
20     unsigned short bfReserved2;     // 使わない
21     unsigned long  bfOffBits;       // 使わない
22 } __attribute__((packed)) BITMAPFILEHEADER;
23
24 typedef struct tagBITMAPINFOHEADER{
25     unsigned long  biSize;          // ビットマップ情報ヘッダ
26     long           biWidth;         // 情報ヘッダサイズ
27     long           biHeight;        // 画像の幅
28     unsigned short biPlanes;       // 画像の高さ
29     unsigned short biBitCount;     // プレーン数（1に固定）
30     unsigned long  biCompression;  // 1ピクセルあたりのビット数
31     unsigned long  biSizeImage;    // 圧縮タイプ
32     long           biXPixPerMeter; // 画像データサイズ
33     long           biYPixPerMeter; // 横1mあたりのピクセル数
34     unsigned long  biClrUsed;     // 縦1mあたりのピクセル数
35     unsigned long  biClrImportant; // パレット数
36 } __attribute__((packed)) BITMAPINFOHEADER;
37
38 /*
39  * draw_bmp - draw bitmap image on LCD screen
40  *   pfb: pointer to the framebuffer
41  *   x0: x coordinates of image
42  *   y0: y coordinates of image
43  *   w: image width
44  *   h: image height
45  *   bmpdata: pointer to the image data
46  */
47 void draw_bmp(unsigned int *pfb, int x0, int y0, int w, int h, unsigned char *bmpdata)
48 {
49     XXXXX...
50 }
51
52 int main(int argc, char **argv) {
53     int fd_in, fd_fb;
54     unsigned int *pfb;
55     unsigned char *bmpdata;
56     int datasize;
57     int x, y;
58
59     struct bmpheader_t{
60         BITMAPFILEHEADER fh;
61         BITMAPINFOHEADER ih;
62     } bmp;
63
64     // 引数が設定されていなかった場合はエラーで終了します。

```

```

65     if (argc != 2){
66         fprintf(stderr, "Usage: %s bmpfile\n", argv[0]);
67         return 1;
68     }
69
70     // 引数に設定されたファイルをオープンします。
71     // オープンに失敗した場合はエラーで終了します。
72     if ((fd_in = open(argv[1], O_RDONLY)) < 0) {
73         perror("open(file)");
74         return 1;
75     }
76
77     // ピットマップヘッダに、画像データを読み込みます。
78     // 読み込みに失敗した場合はエラーで終了します。
79     if (read(fd_in, &bmp, sizeof(bmp)) != sizeof(bmp)){
80         perror("read(file)");
81         return 1;
82     }
83
84     // 取得した画像データより、
85     // 識別子、1ピクセルあたりのピット数、圧縮タイプ、画像の高さをチェックします。
86     if (bmp.fh.bfType != 0x4d42 || bmp.ih.biBitCount != 24
87         || bmp.ih.biCompression != 0 || bmp.ih.biHeight < 0){
88         fprintf(stderr, "unsupported bitmap format\n");
89         return 1;
90     }
91
92     // 取得した画像データより、
93     // 画像データの幅と高さが画面サイズよりも大きい場合はエラーで終了します。
94     if (XXXXX){
95         fprintf(stderr, "image size too big\n");
96         return 1;
97     }
98
99     // 画像データから、ピットマップファイルのヘッダ情報のデータサイズを引いた値を
100    // データサイズとして、メモリ領域を確保します。
101    datasize = bmp.fh.bfSize - sizeof(bmp);
102    // 必要なメモリ領域を確保できない場合はエラーで終了します。
103    if (!(bmpdata = malloc(datasize))){
104        perror("malloc");
105        return 1;
106    }
107
108    // 確保したメモリ領域に画像データを読み込みます。
109    // 読み込みに失敗した場合はエラーで終了します。
110    if (read(fd_in, bmpdata, datasize) != datasize){
111        perror("read(file)");
112        free(bmpdata);
113        return 1;
114    }
115
116    // 画像ファイルをクローズします。
117    close(fd_in);
118
119    // フレームバッファをオープンします。
120    // オープンに失敗した場合はエラーで終了します。
121    if ((fd_fb = open("/dev/fb0", O_RDWR)) < 0) {
122        perror("open(fb)");
123        free(bmpdata);
124        return 1;
125    }
126
127    // mmapによりバッファの先頭アドレスを取得します。
128    pfb = mmap(0, SCREENSIZE, PROT_READ|PROT_WRITE, MAP_SHARED, fd_fb, 0);
129    // 取得に失敗した場合はエラーで終了します。
130

```

```

130     if (pfb == MAP_FAILED){
131         perror("mmap");
132         free(bmpdata);
133         return 1;
134     }
135
136     // LCDの画面表示を全て消去するために、
137     // 取得したアドレスから確保領域を0で初期化します。
138     memset(pfb, 0, SCREENSIZE);
139
140     // 画像がLCDの中央に表示されるようにx, yを設定します。
141     x = XXXXX;
142     y = XXXXX;
143     // LCDに画像を表示します。
144     draw_bmp(pfb, x, y, XXXXX, XXXXX, bmpdata);
145
146     // フレームバッファのために確保した領域を開放します。
147     munmap(pfb, SCREENSIZE);
148     // フレームバッファをクローズします。
149     close(fd_fb);
150     // 画像表示に必要なメモリ領域を開放します。
151     free(bmpdata);
152
153     return 0;
154 }
155
156

```

Makefile

▼ 08a_lcd/Makefile

```

1 CC = arm-linux-gnueabihf-gcc
2 #TARGET = lcdclear colorbar dispbmp
3 TARGET = dispbmp
4 CFLAGS = -gdwarf-2 -O0
5
6 all: $(TARGET)
7
8 lcdclear: lcdclear.c
9     $(CC) $(CFLAGS) -o $@ $<
10
11 colorbar: colorbar.c
12     $(CC) $(CFLAGS) -o $@ $<
13
14 dispbmp: dispbmp.c
15     $(CC) $(CFLAGS) -o $@ $<
16
17 install :
18     cp -p $(TARGET) /work/linux/nfsroot/debug/04_practice
19     cp -p $(TARGET) /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice
20     cp -p $(TARGET).c /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice
21     cp -p bitmap/*.bmp /work/linux/nfsroot/debug/04_practice
22     cp -p bitmap/*.bmp /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice
23
24 clean:
25     rm -f $(TARGET)
26
27 .PHONY: clean

```

1.6.2. 動作確認

make clean

▼ \$ make clean

```
1 | atmark@atde8:/media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/08a_lcd$ make clec
2 | rm -f dispbmp
```

make

▼ \$ make

```
1 | atmark@atde8:/media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/08a_lcd$ make disp
2 | arm-linux-gnueabihf-gcc -gdwarf-2 -O0 -o dispbmp dispbmp.c
```

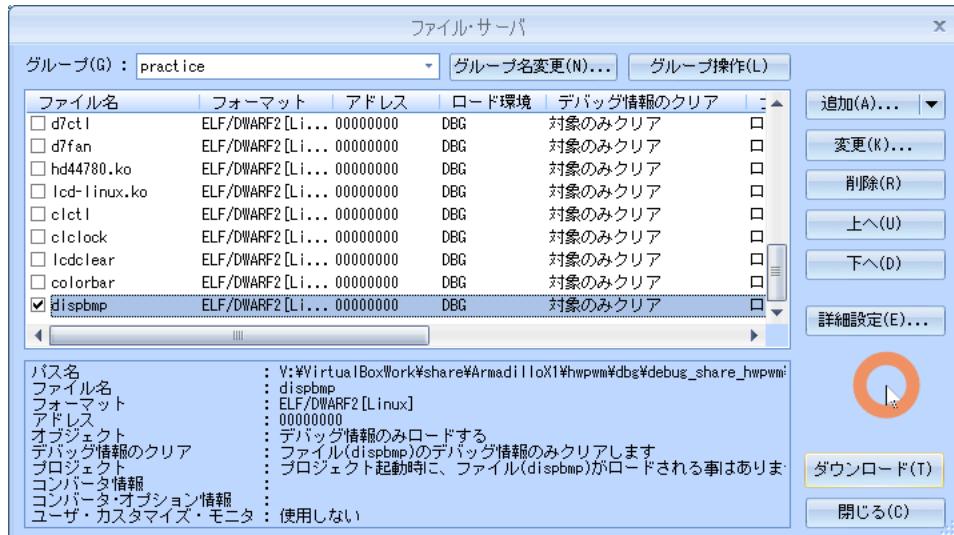
sudo make install

▼ \$ sudo make install

```
1 | atmark@atde8:/media/sf_ArmadilloX1/hwpwm/work/R06_2024/Application_debug/text/practice-example/08a_lcd$ sudo make
2 | cp -p dispbmp /work/linux/nfsroot/debug/04_practice
3 | cp -p dispbmp /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice
4 | cp -p dispbmp.c /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice
5 | cp -p bitmap/*.bmp /work/linux/nfsroot/debug/04_practice
6 | cp -p bitmap/*.bmp /media/sf_ArmadilloX1/hwpwm/dbg/debug_share_hwpwm/R06_2024/04_practice
```

CSIDEでロード

▼ メニュー「ファイル」 - 「ロード」



実行

▼ root@armadillo:/debug/04_practice# ./dispbmp

```
1 | root@armadillo:/debug/04_practice# ./dispbmp 1-sea.bmp
2 | root@armadillo:/debug/04_practice# ./dispbmp 2-park.bmp
3 | root@armadillo:/debug/04_practice# ./dispbmp 3-church.bmp
4 | root@armadillo:/debug/04_practice# ./dispbmp 4-factory.bmp
5 | root@armadillo:/debug/04_practice# ./lcdclear
```

実行している様子

▼ dispbmp で BMP画像ファイルを描画

<https://youtu.be/EPYFJ4yTZWY>

