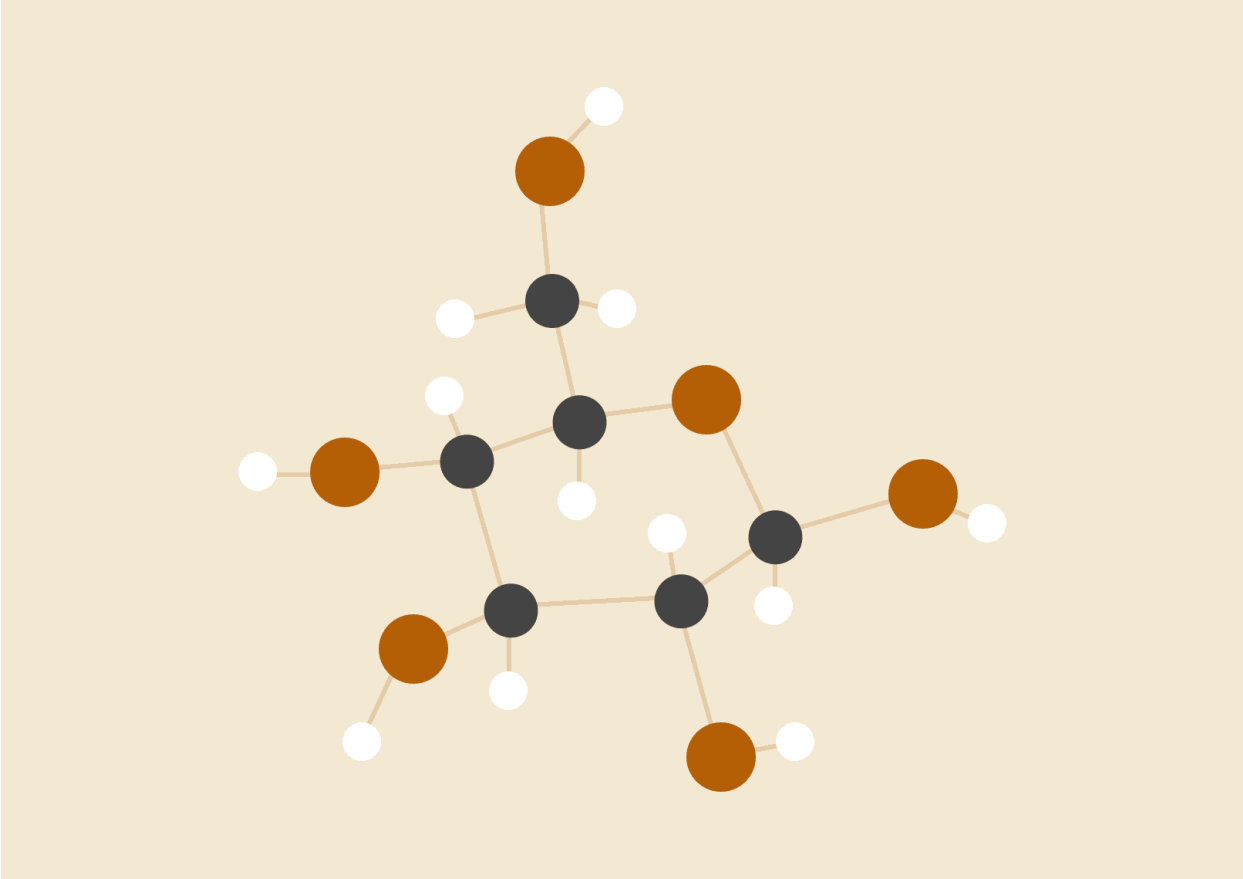
EIN PROGRAMM LESEN

*Ein Bild sagt mehr als tausend Worte, sagt man, tausend Worte sind viele Worte und danach ist man verwirrt, da ist **ein** Bild viel besser.*
Das Bild unten … sagt ?



engelbert

20.02.2022
1ST GRADE SCIENCE

THE PROBLEM

Man kann sich alles herunterladen (siehe Hitchhikers Guide to the Galaxy), sagen mir Schüler. Man sollte mit dem heruntergeladenen auch umgehen können, sage ich dann.

Wie finde ich in einem Haufen Beispiel-Code den Teil den ich brauche.

MATERIALS

1. Interesse
2. Zeit
3. Internet mit Download Volumen > 0

EIN BEISPIEL

Als Beispiel nehme ich das E-Paper von Waveshare. Hier https://www.waveshare.com/wiki/7.5inch_HD_e-Paper_HAT, gibt es Beispiel-Code zum herunterladen (die Schüler haben recht).

Eine Datei E-Paper_ESP32_Driver_Board_Code.7z mit 6,6 MB. Beim Entpacken mit 7z erfahre ich, es sind:

```
Folders: 628
Files:   1353
Size:    36959309
```

also einiges an … Information/Spaß.

PROCEDURE

Das oberste Verzeichnis

Enthält die Dateien : `app-release.apk`, `Version_CN.txt`, `Version_EN.txt`

Und die Verzeichnisse : `ePape_Esp32_Loader_APP`, `examples`, `Loader_esp32bt`, `Loader_esp32wf`

examples

Wir schauen in **`examples`** ob da ein kleines Beispiel drin ist.

Da gibt es nur **`esp32-waveshare-epd`** , in dem ist dann :

```
examples/  extras/  keywords.txt  library.properties
readme_CN.txt  readme_EN.txt  src/
```

.txt-Dateien … wir lesen Code, keinen Text … noch nicht.

Also noch einmal in

examples / esp32-waveshare-epd / examples

Da gibt es:

epd1in54b-demo epd2in13d-demo epd2in7b_V2-demo epd3in7-demo
epd5in83b_V2-demo epd7in5_HD-demo epd1in54b_V2-demo epd2in13-demo
epd2in7-demo epd4in01f-demo epd5in83-demo epd7in5_V2-demo
epd1in54c-demo epd2in13_V2-demo epd2in9bc-demo epd4in2bc-demo
epd5in83_V2-demo epd1in54-demo epd2in13_V3-demo epd2in9b_V3-demo
epd4in2b_V2-demo epd7in5bc-demo epd1in54_V2-demo epd2in66b-demo
epd2in9d-demo epd4in2-demo epd7in5b_HD-demoepd2in13bc-demo
epd2in66-demo epd2in9-demo epd5in65f-demo epd7in5b_V2-demo
epd2in13b_V3-demo epd2in7b-demo epd2in9_V2-demo epd5in83bc-demo
epd7in5-demo

Ziemlich viel. Wir haben ein 7,5 Zoll E-Paper und es gibt Verzeichnisse mit “7in5” im Namen, “in” steht vielleicht für “inch”, das ist “Zoll” in english. Dann wird es nicht weniger, aber leichter. Wahrscheinlich immer derselbe Code nur für verschieden große Displays.

Wir nehmen

epd7in5-demo

Da sind nur noch drei Dateien drin: `epd7in5-demo.ino  ImageData.c ImageData.h`

Die “ino”-Datei weist daraufhin, dass es ein arduino-Sketch ist.

Wir öffnen das mit der arduino-IDE.

Source Code - Original

So zwei Seiten beziehungsweise 83 Zeilen.

```
/* Includes
------------------------------------------------------------*/
#include "DEV_Config.h"
#include "EPD.h"
#include "GUI_Paint.h"
#include "imagedata.h"
#include <stdlib.h>

/* Entry point
------------------------------------------------------------*/
void setup()
{
  printf("EPD_7IN5_test Demo\r\n");
  DEV_Module_Init();

  printf("e-Paper Init and Clear...\r\n");
  EPD_7IN5_Init();
  EPD_7IN5_Clear();
  DEV_Delay_ms(500);

  //Create a new image cache
  UBYTE *BlackImage;
  /* you have to edit the startup_stm32fxxx.s file and set a big
enough heap size */
  UWORD Imagesize = ((EPD_7IN5_WIDTH % 8 == 0) ? (EPD_7IN5_WIDTH / 8
) : (EPD_7IN5_WIDTH / 8 + 1)) * EPD_7IN5_HEIGHT;
  if ((BlackImage = (UBYTE *)malloc(Imagesize)) == NULL) {
    printf("Failed to apply for black memory...\r\n");
    while (1);
  }
  printf("Paint_NewImage\r\n");
  Paint_NewImage(BlackImage, EPD_7IN5_WIDTH, EPD_7IN5_HEIGHT, 0,
WHITE);

#if 1   // show image for array
  printf("show image for array\r\n");
  Paint_SelectImage(BlackImage);
  Paint_Clear(WHITE);
  Paint_DrawBitMap(gImage_7in5);
  EPD_7IN5_Display(BlackImage);
  DEV_Delay_ms(500);
```

```
#endif

#if 1   // Drawing on the image
  //1.Select Image
  printf("SelectImage:BlackImage\r\n");
  Paint_SelectImage(BlackImage);
  Paint_Clear(WHITE);

  // 2.Drawing on the image
  printf("Drawing:BlackImage\r\n");
  Paint_DrawPoint(10, 80, BLACK, DOT_PIXEL_1X1, DOT_STYLE_DFT);
  Paint_DrawPoint(10, 90, BLACK, DOT_PIXEL_2X2, DOT_STYLE_DFT);
  Paint_DrawPoint(10, 100, BLACK, DOT_PIXEL_3X3, DOT_STYLE_DFT);
  Paint_DrawLine(20, 70, 70, 120, BLACK, DOT_PIXEL_1X1,
LINE_STYLE_SOLID);
  Paint_DrawLine(70, 70, 20, 120, BLACK, DOT_PIXEL_1X1,
LINE_STYLE_SOLID);
  Paint_DrawRectangle(20, 70, 70, 120, BLACK, DOT_PIXEL_1X1,
DRAW_FILL_EMPTY);
  Paint_DrawRectangle(80, 70, 130, 120, BLACK, DOT_PIXEL_1X1,
DRAW_FILL_FULL);
  Paint_DrawCircle(45, 95, 20, BLACK, DOT_PIXEL_1X1,
DRAW_FILL_EMPTY);
  Paint_DrawCircle(105, 95, 20, WHITE, DOT_PIXEL_1X1,
DRAW_FILL_FULL);
  Paint_DrawLine(85, 95, 125, 95, BLACK, DOT_PIXEL_1X1,
LINE_STYLE_DOTTED);
  Paint_DrawLine(105, 75, 105, 115, BLACK, DOT_PIXEL_1X1,
LINE_STYLE_DOTTED);
  Paint_DrawString_EN(10, 0, "waveshare", &Font16, BLACK, WHITE);
  Paint_DrawString_EN(10, 20, "hello world", &Font12, WHITE, BLACK);
  Paint_DrawNum(10, 33, 123456789, &Font12, BLACK, WHITE);
  Paint_DrawNum(10, 50, 987654321, &Font16, WHITE, BLACK);
  Paint_DrawString_CN(130, 0, " 你好abc", &Font12CN, BLACK, WHITE);
  Paint_DrawString_CN(130, 20, "微雪电子", &Font24CN, WHITE, BLACK);

  printf("EPD_Display\r\n");
  EPD_7IN5_Display(BlackImage);
  DEV_Delay_ms(2000);
#endif

  printf("Clear...\r\n");
  EPD_7IN5_Clear();

  printf("Goto Sleep...\r\n");
  EPD_7IN5_Sleep();
  free(BlackImage);
  BlackImage = NULL;
```

```
}

/* The main loop
------------------------------------------------------------*/
void loop()
{
  //
}
```

Ein Überblick

Es gibt

- Includes: Das sind externe Dateien die im Programm verwendet werden.
- “Entry Point”: Die setup-Funktion des arduino-Sketch.
- Und eine leere loop-Funktion.

Und ab ins Detail

Der ganz Code steht also in der setup-Funktion. Wir lesen.

```
  printf("EPD_7IN5_test Demo\r\n");
  DEV_Module_Init();

  printf("e-Paper Init and Clear...\r\n");
  EPD_7IN5_Init();
  EPD_7IN5_Clear();
  DEV_Delay_ms(500);
```

Es wird das “DEV_Module” initialisiert … keine Ahnung was das ist, vielleicht später genauer schauen.

Dann wird das “EPD_7IN5” initialisiert, das ist unser E-Paper. Und es wird gelöscht und dann eine halbe Sekunde pausiert … ob für uns oder das E-Paper … weiß ich nicht.

Der nächste Block hat den Titel … zumindest sagt der Kommentar das

Create a new image cache

“Cache” ist ein Zwischenspeicher, wenn man nicht direkt ins Ziel schreiben kann oder will. Das heisst hier wird der Speicherbereich angelegt oder festgelegt oder initialisiert oder …

```
  //Create a new image cache
  UBYTE *BlackImage;
```

```
  /* you have to edit the startup_stm32fxxx.s file and set a big
enough heap size */
  UWORD Imagesize = ((EPD_7IN5_WIDTH % 8 == 0) ? (EPD_7IN5_WIDTH / 8
) : (EPD_7IN5_WIDTH / 8 + 1)) * EPD_7IN5_HEIGHT;
  if ((BlackImage = (UBYTE *)malloc(Imagesize)) == NULL) {
    printf("Failed to apply for black memory...\r\n");
    while (1);
  }
```

Das ist ziemlich heftig, beziehungsweise plain old C. Die Details interessieren uns frühestens später.

Wenn es nicht funktioniert gibt das Programm “"`Failed to apply for black memory…`” aus und hängt dann für immer in “while (1);”.

Wenn es funktioniert hat, ist BlackImage so groß wie es für unsere Anzeige nötig ist. Dann kommt

Paint_NewImage

```
  printf("Paint_NewImage\r\n");
  Paint_NewImage(BlackImage, EPD_7IN5_WIDTH, EPD_7IN5_HEIGHT, 0,
WHITE);
```

Das klingt jetzt komisch, aber ich glaube hier wird BlackImage weiß angemalt. Der Name sollte vielleicht besser “Image” oder “Zwischenbild” sein.

Dann gibt es zwei Blöcke die zwischen omische Zeilen gesetzt sind

```
#if 1   // show image for array
    …
#endif
#if 1   // Drawing on the image
   …
#endif
```

Die Hashtag Zeilen sind nicht für den C-Compiler sondern für den Präprozessor, das ist nicht C-Code, aber ähnlich “`#if 1`” ist immer wahr, das heisst der nachfolgende Bereich bis “`#endif`” ist im Programm.

Wenn ich nicht ins Bild malen will ändere ich das zweite “#if”
zu “`#if 0   // Drawing on the image`” oder lösche die ganzen Zeilen in dem Bereich.

Um ein Bitmap auszugeben, schauen wir uns den ersten Bereich an.

show image for array

```
#if 1   // show image for array
  printf("show image for array\r\n");
  Paint_SelectImage(BlackImage);
  Paint_Clear(WHITE);
  Paint_DrawBitMap(gImage_7in5);
  EPD_7IN5_Display(BlackImage);
  DEV_Delay_ms(500);
#endif
```

`Paint_SelectImage` … select = wählen , wir wählen unser BlackImage als das Bild mit dem wir arbeiten.

`Paint_Clear(WHITE)` … wir malen noch einmal weiß drüber.

(NOTE: ich habe das Display nicht, ich lese nur den Code, das kann also auch falsch/anders sein)

`Paint_DrawBitMap(gImage_7in5)` … hier wird die Bitmap gImage_7in5 auf das gewählte Bild ge…zeichnet.

`EPD_7IN5_Display(BlackImage)` … EPD_7IN5 … so heißt unser Demoprogramm … E-Paper 7,5 inch, das ist unser E-Paper. “Display” heißt “Anzeige” ist aber auch ein Zeitwort (sollte es dann nicht klein geschrieben werden). Das heisst hier wird unser BlackImage zum E-Paper geschickt.

Dann eine halbe Sekunde warten … wir könnten statt 500, 5000 schreiben dann sollte das Bild länger angezeigt werden.

Woher kommt

gImage_7in5

In der arduino IDE sind noch zwei andere Dateien geöffnet worden

ImageData.c und ImageData.h

In ImageData.c steht nach dem langen Copyright-Kommentar

```
const unsigned char gImage_7in5[] = { /* 0X00,0X01,0X80,0X02,0X80,0X01, */
0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,
```

Hier wird ein Array (“[]”) mit dem Namen gImage_7in5 angelegt.

Die Größe rechnet sich der Compiler aus, das kann er weil der Inhalt danach zugewiesen wird

```
 = { =  /* 0X00,0X01,0X80,0X02,0X80,0X01, */
0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0xFF
```

Danach kommen ganz viele 0xFF und manchmal 0x3F und 0xF0.

F hexadezimal ist 1111, 3 ist 0011.

Binär betrachtet ist :

```
0xFF 0XF0 0X3F 0XFF

11111111111100000011111111111111
F    F    F    0    3    F    F    F
```

Wenn 0 ein schwarzer Punkt und 1 ein weisser ist ist das eine unterbrochene Linie.

VERSUCHE

1. Ändern des Musters in gImage_7in5.
2. Das Muster vom Programm aus setzen.
   Wir können gImage_7in5 nicht vom Programm aus ändern, weil es
   `const unsigned char gImage_7in5[] = {`
   ist, “const” meint konstant, nicht veränderbar.

   Aber das lässt sich ändern, wir löschen “const”:
   `unsigned char gImage_7in5[] = {`

   dann können wir vor der Zeile

   ```
   Paint_DrawBitMap(gImage_7in5);
   ```

   ein Muster in gImage_7in5 schreiben

   ```
   for (int i=0; i<sizeof(gImage_7in5); i++) {
     if (i % 2) {
       gImage_7in5[i] = 0xFF;
     }
   ```

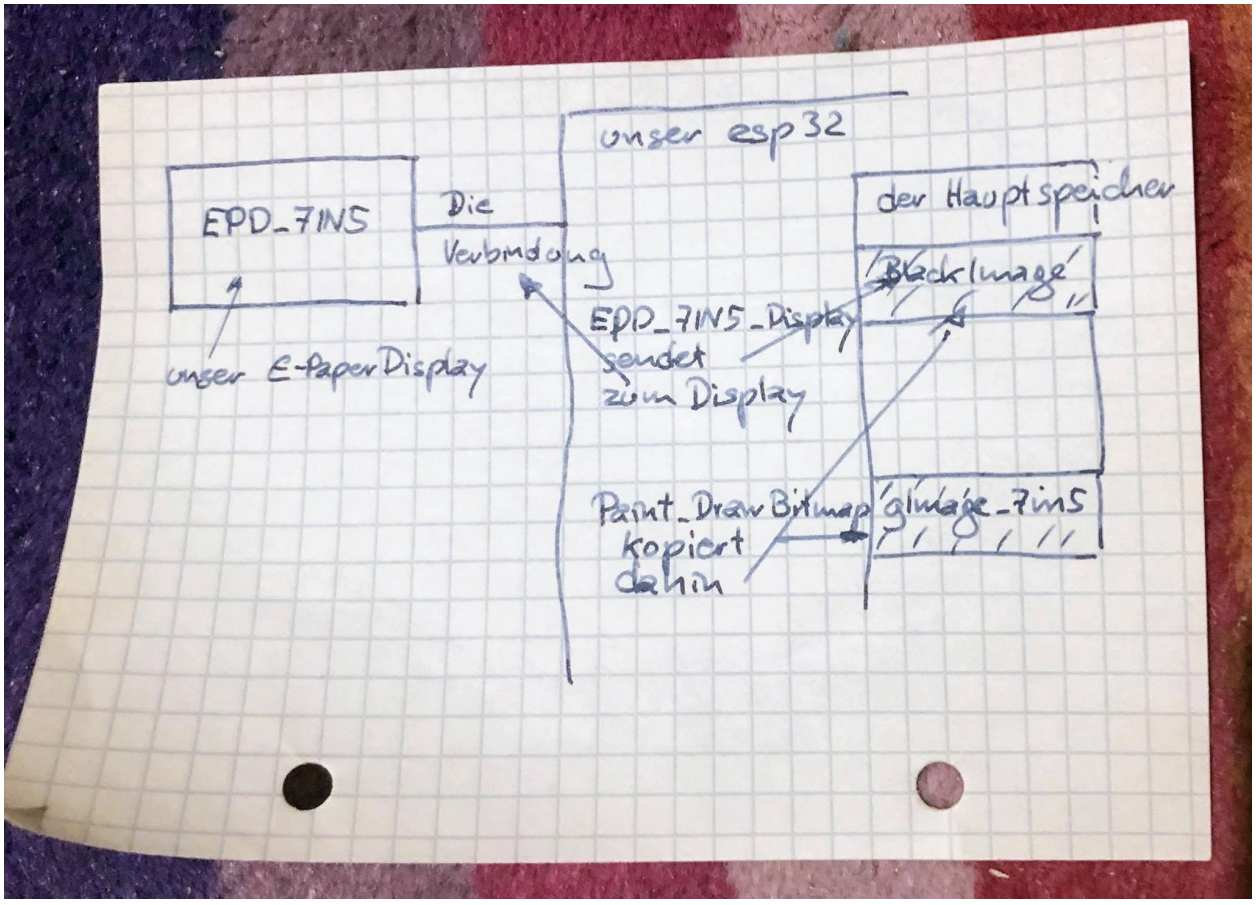
```
    else {
      gImage_7in5[i] = 0;
    }
  }
```

Dann sollte ein Streifenmuster angezeigt werden.

3. Wir könnten das direkt in BlackImage schreiben .
   Problem : sizeof(BlackImage) ist nur die Größe des Zeigers nicht des Speicherbereiches.
4. Ein Bild von einem webserver holen und in gImage_7in5 oder BlackImage schreiben.

RESULTS

Are there any ?



Wo ist `EPD_7IN5_Display` ?

In `examples/esp32-waveshare-epd/examples/epd7in5-demo` sind nur drei Dateien:

```
epd7in5-demo.ino  ImageData.c  ImageData.h
```

In `epd7in5-demo.ino` werden einige Dateien inkludiert, unter anderem :

```
#include "DEV_Config.h"
#include "EPD.h"
#include "GUI_Paint.h"
#include "imagedata.h"
#include <stdlib.h>
```

- `imagedata.h` , kennen wir
- `GUI_Paint.h` , da wird Paint_DrawBitmap sein
- bleibt `EPD.h` für `EPD_7IN5_Display` .

Die Dateien sind nicht im epd_7in5-demo-Verzeichnis.

Nicht im darüberliegenden. Das wäre ein guter Platz, da alle die demo-Programme wahrscheinlich **`EPD.h`** und **`GUI_Paint.h`** verwenden. Aber da sind wir noch im examples-Verzeichnis des **`esp32-waveshare-epd`**.

Der Code zur Ansteuerung gehört aber nicht zu den examples sondern zum Paket, der Bibliothekskomponente. Im Verzeichnis **`esp32-waveshare-epd`** gibt es auch ein src-Verzeichnis, in dem sind **`EPD.h`** und **`GUI_Paint.h`** und **`GUI_Paint.c`**.

EPD.h enthält nur includes, unter anderem :

```
#include "utility/EPD_7in5.h"
```

Dort liegt auch `EPD_7in5.cpp` und da finden wir den Code von EPD_7IN5_Display:

```
/******************************************************************************
function :   Sends the image buffer in RAM to e-Paper and displays
Parameter:
******************************************************************************/
void EPD_7IN5_Display(UBYTE *Image)
{
  UBYTE Data_Black, Data;
  UWORD Width, Height;
  Width = (EPD_7IN5_WIDTH % 8 == 0)? (EPD_7IN5_WIDTH / 8 ): (EPD_7IN5_WIDTH / 8 + 1);
  Height = EPD_7IN5_HEIGHT;

  EPD_7IN5_SendCommand(0x10);
```

```
  for (UWORD j = 0; j < Height; j++) {
    for (UWORD i = 0; i < Width; i++) {
      Data_Black = ~Image[i + j * Width];
      for(UBYTE k = 0; k < 8; k++) {
        if(Data_Black & 0x80)
          Data = 0x00;
        else
          Data = 0x03;
        Data <<= 4;
        Data_Black <<= 1;
        k++;
        if(Data_Black & 0x80)
          Data |= 0x00;
        else
          Data |= 0x03;
        Data_Black <<= 1;
        EPD_7IN5_SendData(Data);
      }
    }
  }
  EPD_7IN5_TurnOnDisplay();
}
```

Das ist ziemlich tief und es ist schon spät und so genau muss ich es nicht wissen. Aber bei Bedarf weiß ich wo es steht.

`EPD_7IN5_SendData` ist viel feiner/kürzer

```
/******************************************************************************
function :    send data
parameter:
  Data : Write data
******************************************************************************/
static void EPD_7IN5_SendData(UBYTE Data)
{
  DEV_Digital_Write(EPD_DC_PIN, 1);
  DEV_Digital_Write(EPD_CS_PIN, 0);
  DEV_SPI_WriteByte(Data);
  DEV_Digital_Write(EPD_CS_PIN, 1);
}
```

Ha, und da steht dann auch SPI.

EPD_7IN5_Display Nachwort

Es ist nicht kompliziert sondern beschreibt den Vorgang, das Verständnis, recht gut:

Die an die Funktion übergebenen Daten werden Byteweise durchgeschaut

- jede Zeile von 0 bis Height (-1 weil wir mit 0 angefangen haben),
- in jeder Zeile vom ersten bis zum letzten Byte,
- in jedem Byte vom höchstwertigen bis zum nullten Bit.

In C-Sprache:

```
void EPD_7IN5_Display(UBYTE *Image)
```

Die Funktion `EPD_7IN5_Display` hat einen Parameter, benannt "`Image`", weil da ein Stern ist "*" ist "`Image`" die Adresse eines "UBYTE", in Programmiersprache ein Zeiger, Pointer auf ein UBYTE.

```
{
  UBYTE Data_Black, Data;
```

`Data_Black` und `Data` sind jeweils ein `UBYTE`.

```
  UWORD Width, Height;
  Width = (EPD_7IN5_WIDTH % 8 == 0)? (EPD_7IN5_WIDTH / 8 ): (EPD_7IN5_WIDTH / 8 + 1);
  Height = EPD_7IN5_HEIGHT;
```

Weil in einem Byte 8 Bit enthalten sind, können dort 8 Displaypunkte stehen und wir brauchen für eine Bildweite von 640 Punkten nur 80 Byte.

```
  EPD_7IN5_SendCommand(0x10);
  for (UWORD j = 0; j < Height; j++) {    // Zeile 0 bis Height-1
    for (UWORD i = 0; i < Width; i++) {   // Jedes Byte in der Zeile
      Data_Black = ~Image[i + j * Width]; // Das Byte an der Position, invertiert

      for(UBYTE k = 0; k < 8; k++) {      // Jedes Bit im Byte
        if(Data_Black & 0x80)             // nur Bit 7 anschauen, Maske 0x80=10000000
          Data = 0x00;                    //     "&" ist bitweises AND
        else
          Data = 0x03;
        Data <<= 4;
        Data_Black <<= 1;                 // um eins nach links schieben
                                          // das 6.Bit wird zum 7.Bit
        k++;
        if(Data_Black & 0x80)             // nur Bit 7 anschauen
          Data |= 0x00;                   // kann man sich sparen, tut nichts
        else
          Data |= 0x03;                   // "|= 0x03" verodert 00000011, setzt 11.
        Data_Black <<= 1;
```

```
          EPD_7IN5_SendData(Data);
        }
      }
    }
    EPD_7IN5_TurnOnDisplay();
}
```

CONCLUSION

If there are Results, there are possibilities, are there ?