ESP32forth and Arduino C++ - notes about ESP32 and e-Paper display.

ESP32forth is written in Arduino C and some knowledge of C is very helpful. As amateur programmer, with basic knowledge of Forth only, I resolved to learn also basics of Arduino C to be able to better understand and use ESP32forth. I created some notes about my examples to help others in the same situation.

For my next construction I need low power display, so e-Paper display is good solution. ESP32forth does not have direct support for it, but ESP32 chip is able to do SPI serial communication, which is typically used for some e-Paper displays. For testing I have bought 7.5 inch 800x480 b/w display with appropriate SPI-e-Paper connection board. Display is labeled GDEY075T7 and uses 3.3V, so perfect for ESP32 chip. My project is inspired with this project on github https://github.com/G6EJD/ESP32-e-Paper-Weather-Display , where is possible to find a lot of details. Of course project is written in Arduino C.
For forth support I at first started with display data sheet, but direct use of 50 display commands needs a lot of testing and mainly a lot of time. So better for me to use something more easy - C libraries prepared for it. With this is not so difficult to take control about this e-Paper display board. There is also some

graphics standard with prepared ready basic commands for shapes and text creation - Adafruit GFX Graphics Library. This is equiped with DOCUMENTATION, which pleased me in my forth world. Well, for ESP32forth I have created *userwords.h* file using necessary c libs. Full files from my article are located on github, in article are shortened parts: https://github.com/Vaclav-Poselt/ESP32_forth_code . Here is as example first part of u*serwords.h* file:

```
9
10  #include <GxEPD2_BW.h> // Include the B/W e-paper library
11  #include <Adafruit_GFX.h> // Include Adafruit GFX for graphics commands
12  #include <Fonts/FreeSans18pt7b.h>
13  #include <Fonts/FreeSansBold18pt7b.h>
14  #define DISPLAY_TYPE_GDEY075T7
15
16  #if __has_include("gdey/GxEPD2_750_GDEY075T7.h")
17  | #include "gdey/GxEPD2_750_GDEY075T7.h"
18  | #endif
19  // pins for ePaper display connection
20  | #define MOSI   23
21  | #define CLK    18
22  | #define SS     5
23  | #define DC     17
24  | #define RST    16
25  | #define BUSY   4
26  | // #define POWER 2 // power of display from out pin does not work?
27  |
28  typedef GxEPD2_BW<GxEPD2_750_GDEY075T7, GxEPD2_750_GDEY075T7::HEIGHT> DisplayType;
29  DisplayType display(GxEPD2_750_GDEY075T7(SS, DC, RST, BUSY));
30
31  #define USER_WORDS \
32  X("D.init", dinit, display.init(); ) \
33  X("D.setRotation",setrot, display.setRotation(n0); DROP; ) \
34  X("D.display", disp, display.display(); ) \
35  X("D.fillScreen", filscr, display.fillScreen(n0); DROP;) \
36  X("D.setCursor", setcur, display.setCursor(n1,n0); DROPn(2);) \
37  X("D.drawPixel", drpix, display.drawPixel(n2,n1,n0); DROPn(3);) \
38  X("D.drawLine", drlin, display.drawLine(n4,n3,n2,n1,n0); DROPn(5);) \
39  X("D.drawFastVLine", drfavlin, display.drawFastVLine(n3,n2,n1,n0); DROPn(4);) \
40  ...
```

Basic part is *Adafruit_GFX.h* file with graphics commands. I don't create stack diagrams as commands are explained in documentation for this library:
https://learn.adafruit.com/adafruit-gfx-graphics-library?view=all
http://adafruit.github.io/Adafruit-GFX-Library/html/class_adafruit___g_f_x.html
Next there are libs for used display type, which can be easily changed if other model is used and example of free fonts usage. Library also takes care of SPI communication, not necessary to install it separately. e-Paper display is slow, but for weather forecast there is no hurry, main advantage is that picture remains even when disconnected from power.
Practical usage runs in cycle initialize - draw content to buffer- refresh display to show it. There are two types of graphical commands. First type draw pictures or text according adjusted x, y starting points, length, diameter aso. Second type draw bitmap images defined in memory. For creation of this bitmaps there is in adafruit guide mentioned simple in browser program image2cpp, which calculates from universal format as png, bmp desired bitmap data. Example of ESP32forth code used looks like this:

```
\ graphics for ePaper display prepared for
\ D.drawbitmap ( x y data sizex sizey--- )

: EPbitmap  ( sizex sizey name--- | --- addr sizex sizey )
    \ word for creation of graphics data
    create swap c, c,           \ save x and y sizes, max 255*255
    does> dup dup c@ swap 1+ c@ rot 2 + -rot \ retrive x and y
    ;

\ '01d', 50x50px
50 50 EPbitmap 01d
$00 c, $00 c, $00 c, $00 c, $00 c, $00 c, $00 c, $00 c, $00 c, $00 c, $00 c, $00
c, $00 c, $00 c, $00 c, $00 c,
$01 c, $e0 c, $00 c, $00 c, $00 c, $00 c, $00 c, $01 c, $e0 c, $00 c, $00 c, $00
c, $00 c, $00 c, $01 c, $e0 c,
$00 c, $00 c, $00 c, $00 c, $00 c, $01 c, $e0 c, $00 c, $00 c, $00 c, $00 c, $00
c, $01 c, $e0 c, $00 c, $00 c,
$00 c, $00 c, $1e c, $01 c, $e0 c, $1e c, $00 c, $00 c, $00 c, $1f c, $01 c, $c0
c, $3e c, $00 c, $00 c, $00 c,
$1f c, $80 c, $00 c, $7e c, $00 c, $00 c, $00 c, $0f c, $80 c, $00 c, $7c c, $00
c, $00 c, $00 c, $07 c, $87 c,
$f8 c, $78 c, $00 c, $00 c, $00 c, $03 c, $9f c, $fe c, $70 c, $00 c, $00 c, $00
c, $00 c, $3f c, $ff c, $00 c,
......
```

There is definition word *EPbitmap* which creates bitmap in memory including sizes and when used it gives necessary data *addr xsize ysize* on stack. for D.drawbitmap command.
Simple practical usage is in next short program, which creates picture used in photo of e-Paper display for this article. It is finished with *D.hibernate* word to switch display to low power mode.

```
0 constant BLACK
1 constant WHITE

: EPinit    ( --- )     \ show white clear screen
    D.init 0 D.setrotation WHITE D.fillscreen
    D.display
;
: rectTest  ( --- )
    50 0 do
        i 10 + dup 200 i 2 * - 100 i 2 * - 5 BLACK D.drawroundrect
    2 +loop  D.display
;
: textTest  ( --- )
    D.freesans18 BLACK D.settextcolor 2 D.settextsize
    210 100 D.setcursor
    z" Hello ESP32forth" D.println
    D.freesansbold18 1 D.settextsize
    z"             Hello ESP32forth" D.println D.display
;
: graphTest ( --- )
    10 300 01d BLACK D.drawbitmap
    60 300 02d BLACK d.drawbitmap
    D.display
;

epinit recttest texttest graphtest D.hibernate

```

If it helps somebody I will be pleased, if there are errors in my explanation I will be also pleased to be corrected.
Files from my article are located on github:
https://github.com/Vaclav-Poselt/ESP32_forth_code
Final note: tested on ESP32forth 7.0.7.20, Arduino IDE2.3.2, ESP32 lib 2.0.14, ESP32 Dev Module board.. In Arduino is necessary to install Adafruit-GFX-Library and GxEPD2 library. I use in Arduino-Tools-Partition scheme adjustment for more memory for apps as No OTA 2M/2M SPIFFS as ready applications can be bigger.