Displaimer Teknisk Dokumentation:
E-ink Display

Yobart Amino <yobart@kth.se>

26 maj 2017

Sammanfattning

Syftet med den electronic ink (e-ink) displayen som användes i projektet Displaimer var att visa det som skrevs på en NGINX-server som kördes på en Raspberry Pi 3. E-ink displayen kommunicerade med Raspberry Pi:en via en WiFi-modul (ESP8266). Denna kommunikation skedde via UART.

I det här dokumentet kommer det gås igenom hur displayen initieras, hur det skrivs över ett meddelande till den samt hur den stängs av. Förkomplexiteten samt enkelhetens skull går dokumentet inte igenom dessa saker mycket djupare än kodmässigt samt hur det implementeras i ett system.

Det här dokumentet är skrivet för de som har grundläggande kunskaper inom programmering.

Innehåll

1 Allmän beskrivning

1.1 Krav

För att använda e-ink displayen krävs ett MCU-kort med stöd för Serial Peripheral Interface, SPI. För Displaimer användes en mikrokontroller av typen STM32F303C6T6, men det går att använda vilken typ av MCU som helst så länge stöd för SPI finns samt en matningsspänning till displayen på 3.3V. 7 stycket GPIO pins krävs för dataöverföring samt ytterligare 3 stycket GPIO pins för SPI bussen.

1.2 Typsnitt

Varje bokstav, siffra och symbol är map:ad pixel för pixel för att kompileras till körbar C-kod som skickas till displayen.

1.3 Displayen

E-ink displayen som användes var en 2.7'' E2271BS021 Aurora Ma baserat på kretsen G2 COG. E-ink displayer är väldigt strömsnåla då de inte behöver kontinuerlig ström för att visa någonting på skärmen. Det krävs ström endast för uppstart, initiering, uppdatering samt avstängning för att skärmen ska funktionera. Därefter kan skärmen stå avstängd och fortfarande visa det meddelande man har skrivit.

2 Implementering

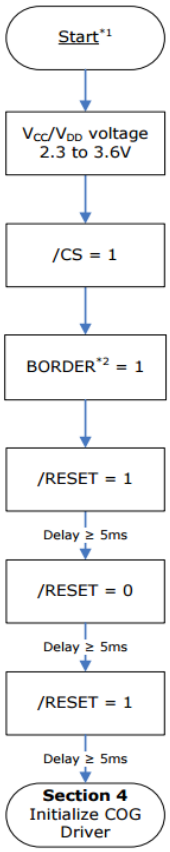
2.1 *Uppstart*

Eftersom det är en e-ink display så behöver den initieras på ett visst sätt för att fungera korrekt. Här beskrivs det hur man slår på G2 COG (den integrerade kretsens drivrutin). I referens [1] hittar man figur 1, där det beskrivs hur uppstart av skärm ska ske. För att implementera detta används GPIO-pins (General-purpose input/output) som kan ses i figur 2. Det första som görs före uppstart är att displayen flashas om och töms från kvarstående rester från förra programmet. Därefter följer, som kan ses, programmet modellen för uppstart.

Efter uppstarten så ska e-ink displayen initieras.

```
void Epaper_Init() {
    HAL_GPIO_WritePin(EPAPER_FLASH_CS_GPIO_Port, EPAPER_FLASH_CS_Pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(EPAPER_POWER_GPIO_Port, EPAPER_POWER_Pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(EPAPER_CS_GPIO_Port, EPAPER_CS_Pin, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(EPAPER_DISCHARGE_GPIO_Port, EPAPER_DISCHARGE_Pin, GPIO_PIN_RESET);

    HAL_GPIO_WritePin(EPAPER_POWER_GPIO_Port, EPAPER_POWER_Pin, GPIO_PIN_SET);

    HAL_GPIO_WritePin(EPAPER_CS_GPIO_Port, EPAPER_CS_Pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(EPAPER_BORDER_CTRL_GPIO_Port, EPAPER_BORDER_CTRL_Pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(EPAPER_RESET_GPIO_Port, EPAPER_RESET_Pin, GPIO_PIN_SET);
    HAL_Delay(10);
    HAL_GPIO_WritePin(EPAPER_RESET_GPIO_Port, EPAPER_RESET_Pin, GPIO_PIN_RESET);
    HAL_Delay(10);
    HAL_GPIO_WritePin(EPAPER_RESET_GPIO_Port, EPAPER_RESET_Pin, GPIO_PIN_SET);
    HAL_Delay(10);
```

Figur 1. Uppstartskod baserad på modellen.

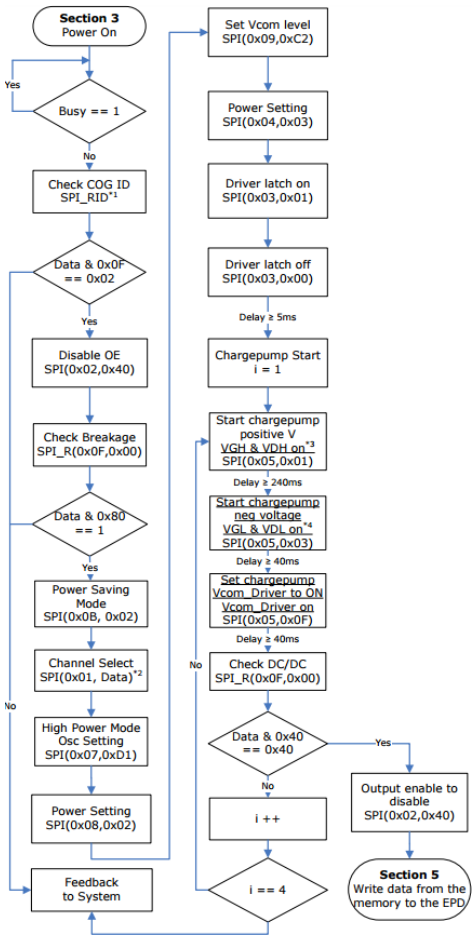


Figur 2. Modell för uppstart av display.

2.2 *Initiering*

Samma sak som förra kapitlet om uppstart gäller för initiering. I referens [1] kan man även se hur man ska gå tillväga för att initiera displayen på ett korrekt sätt, detta illustreras i figur 3 samt implementeras i figur 4. Notera att G2 COG e-ink displayer finns i två olika storlekar. För Displaimer användes 2.7'' varianten. Detta spelar roll för Channel Select, där det för 2.7'' ska skickas 0x0000, 0x007F, 0xFFFE, 0x0000.

Efter att initieringen av G2 COG drivrutinerna skett kan dataöverföring börja för att skriva över meddelanden till displayen.



Figur 3. Modell för tillvägagångsätt.

```
// Check COG id
if ((Epaper_Read_ID() & 0x0f) != 0x02) {
    return;
}

// Disable OE
Epaper_Send_Byte(0x02, 0x40);

// Check breakage
if ((Epaper_Read(0x0F, 0x00) & 0x80) != 0x80) {
    Error_Handler();
}

// Power Saving Mode
Epaper_Send_Byte(0x0B, 0x02);

// Channel Select
Epaper_Send(0x01, ((uint8_t[]){
    0x00, 0x00,
    0x00, 0x7F,
    0xFF, 0xFE,
    0x00, 0x00
}), 8);

// Oscillator Select
Epaper_Send_Byte(0x07, 0xD1);
// Power Setting
Epaper_Send_Byte(0x08, 0x02);
//Vcom Level
Epaper_Send_Byte(0x09, 0xC2);
// Power Setting
Epaper_Send_Byte(0x04, 0x03);
// Driver Latch On
Epaper_Send_Byte(0x03, 0x01);
// Driver Latch Off
Epaper_Send_Byte(0x03, 0x00);
HAL_Delay(10);

// Charge Pump
for (int i = 0; i < 4; ++i) {
    // Positive Voltage
    Epaper_Send_Byte(0x05, 0x01);
    HAL_Delay(150);
    // Negative Voltage
    Epaper_Send_Byte(0x05, 0x03);
    HAL_Delay(100);
    // Vcom on
    Epaper_Send_Byte(0x05, 0x0F);
    HAL_Delay(50);

    if ((Epaper_Read(0x0F, 0x00) & 0x40) == 0x40) {
        // Output enable to disable
        Epaper_Send_Byte(0x02, 0x06);
        epaperOn = 1;
        return;
    }
}
```

Figur 4. Initieringskod för G2 COG.

2.3 Skicka meddelanden

För att skicka ett meddelande användes koden som visas i figur 5.

```
void Epaper_MessageCard_Display(uint8_t *msg, int16_t len) {
  epaperMessageCardBufLen = len;
  for (int i = 0; i < len; ++i) {
    epaperMessageCardBuf[i] = msg[i];
  }
  epaperMessageCardOn = 1;
  Epaper_MessageCard_Update();
}

void Epaper_MessageCard_Update() {
  if (epaperMessageCardUpdating || !epaperMessageCardOn) {
    return;
  }

  epaperMessageCardUpdating = 1;
  Epaper_Clear();
  Epaper_Write_StrLine(0, "Anders Sj\xF6gren");
  Epaper_Draw_HorizLine(1, 1);
  Epaper_Write_StrnLine(2, epaperMessageCardBuf, epaperMessageCardBufLen);
  Epaper_Draw_HorizLine(9, 6);
  Epaper_Write_StrLine(10, dateTimeStr());
  Epaper_Flush();
  epaperMessageCardUpdating = 0;
}
```

Figur 5. Kod för sändning av meddelande

I main.c finns en buffert som fylls genom UART. Denna buffert fylls av ESP8266 som skickar data trådlöst via WiFi. För varje nytt meddelande i bufferten anropas Epaper_MessageCard_Display(buffer, length); där parametrarna är hela bufferten respektive längden på bufferten. Därefter anropas Epaper_MessageCard_Update(); för att uppdatera layouten på skärmen. Layouten ser ut som sådan att ”Anders Sjögren” visas längst upp följt av en horisontell linje, meddelandet, horisontell linje och slutligen tiden på dygnet. Epaper_Write_StrnLine(buffer, length) skriver ut varje karaktär till displayen. För att veta om ett meddelande skulle skrivas ut eller om tiden skulle uppdateras användes en switch på bufferten, där första platsen (plats 0) switchades. Case ’M’ skickar iväg ett meddelande genom anrop av tidigare nämnda funktioner. Case ’T’ anropar istället setDateTime(x,y); där x,y är samma argument som för sändning av meddelande.

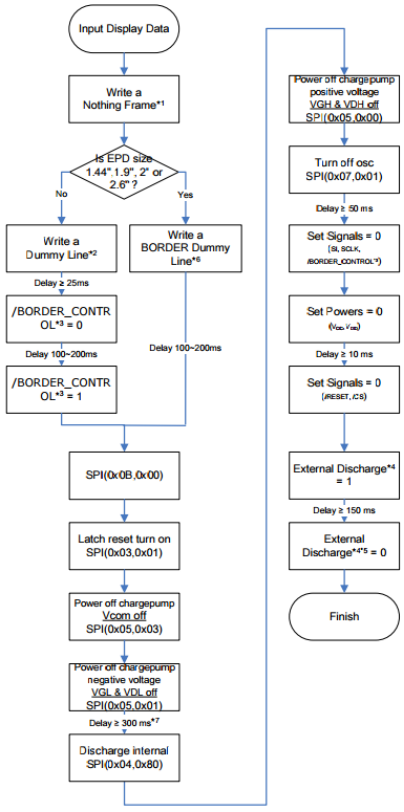
2.4 Avstängning

Eftersom det är en e-ink display som används så krävs inte kontinuerlig ström för att displayen ska vara på eller visa någonting på skärmen. Därför är avstängning en del av processen för implementation av displayen. Precis som påslagning av G2 COG samt initiering av drivrutiner så måste avstängning ske i visas steg.

Dessa steg implementerades enligt figur 6 från avstängningsproceduren i figur 7[1].

```
void Epaper_Shutdown() {
  Epaper_Write_Nothing_Frame();
  Epaper_Write_Dummy_Line();
  HAL_Delay(30);
  HAL_GPIO_WritePin(EPAPER_BORDER_CTRL_GPIO_Port, EPAPER_BORDER_CTRL_Pin, GPIO_PIN_RESET);
  HAL_Delay(150);
  HAL_GPIO_WritePin(EPAPER_BORDER_CTRL_GPIO_Port, EPAPER_BORDER_CTRL_Pin, GPIO_PIN_SET);

  Epaper_Send_Byte(0x0B, 0x00);
  // Latch Reset
  Epaper_Send_Byte(0x03, 0x01);
  // Chargepump Off
  Epaper_Send_Byte(0x05, 0x03);
  // Chargepump Negative Off
  Epaper_Send_Byte(0x05, 0x01);
  HAL_Delay(400);

  // Internal Discharge
  Epaper_Send_Byte(0x04, 0x80);
  // Chargepump Positive Off
  Epaper_Send_Byte(0x05, 0x00);
  // Oscillator off
  Epaper_Send_Byte(0x07, 0x01);
  HAL_Delay(50);

  HAL_GPIO_WritePin(EPAPER_POWER_GPIO_Port, EPAPER_POWER_Pin, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(EPAPER_POWER_GPIO_Port, EPAPER_POWER_Pin, GPIO_PIN_RESET);
  HAL_Delay(20);
  HAL_GPIO_WritePin(EPAPER_RESET_GPIO_Port, EPAPER_RESET_Pin, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(EPAPER_CS_GPIO_Port, EPAPER_CS_Pin, GPIO_PIN_RESET);

  HAL_GPIO_WritePin(EPAPER_DISCHARGE_GPIO_Port, EPAPER_DISCHARGE_Pin, GPIO_PIN_SET);
  HAL_Delay(200);
  HAL_GPIO_WritePin(EPAPER_DISCHARGE_GPIO_Port, EPAPER_DISCHARGE_Pin, GPIO_PIN_RESET);

  epaperOn = 0;
}
```

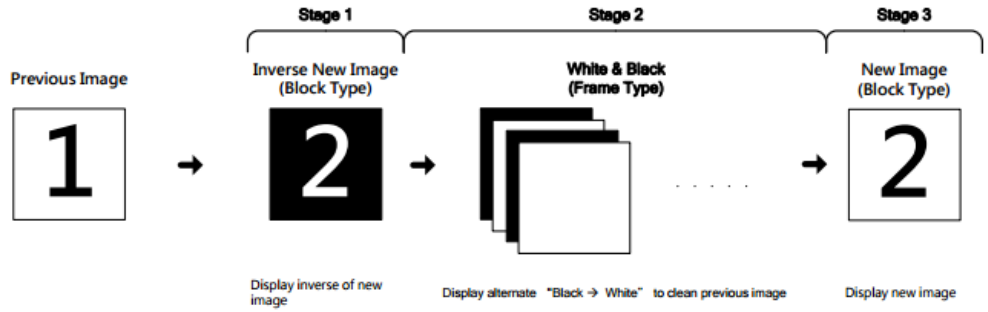
Figur 6. Kod för avstängning av display



Figur 7. Avstängningsprocedur

3 Kända Brister

Ett vanligt förekommande problem med e-ink displayer är så kallad ’ghosting’ eller ’artifacts’. Ghosting sker då en del av förra skärmmönstret visas svagt i det nya. Artifacts ser ut som små prickar på skärmen och stör meddelandets synlighet på skärmen. För att använda e-ink displayen utan att stöta på dessa problem samt att få bästa skärpa och prestanda så ska displayen uppdateras i steg, innan det nya meddelandet kan visas. Hur dessa steg ska ske samt i vilken ordning exemplifieras figur 8 hämtad från referens [1].



Figur 8. Uppdateringssteg för e-ink displayen

Som man kan se i figuren ovan, ska displayen först visa en inverterad bild av meddelandet, sedan ska den växla mellan svart och vit bild ett antal gånger för att till sist visa ett läsbart meddelande på skärmen.

4 Referenser

[1] PERSAVIE DISPLAYS, “E-paper Display COG Driver Interface Timing for Wide Temperature of 1.44”,2” and 2.7” EPD with G2 COG and Aurora Ma Film,” 14-Aug-2015. [Online]. Available: http://www.pervasivedisplays.com/_literature_198794/COG_Driver_Interface_Timing_for_small_size_G2_V230.