T.C.
EGE UNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ

GÖMÜLÜ SİSTEMLER

DÖNEM PROJESİ RAPORU

BERKE KOCADERE

05170000484

MUSTAFA MERT ÖZTÜRK

05150000522

2021-2022 Güz Dönemi

1. Giriş

Projemiz, STM3210C-EVAL kartı ile USB Bellekten dosya/okuma yazma uygulaması oluşturmaktır. Projede, bir adet STM3210C-EVAL geliştirme kartı, 1 adet USB Flash bellek ve USB micro A-Male to A-Female kablo kullanılmıştır.

Proje bir USB Host MSC uygulaması olacaktır. Kısaca bu terimlerden bahsedelim.

2. Genel Bilgiler

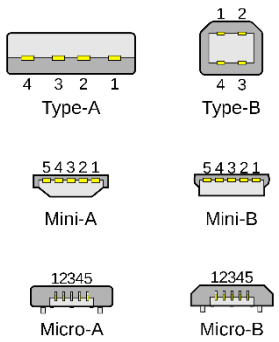
- ***USB***

  Evrensel Seri Veri Yolu (USB), bilgisayarlar, çevre birimleri ve diğer bilgisayarlar arasında bağlantı, iletişim ve güç kaynağı (arabirim) için kablolar, konektörler ve protokoller için spesifikasyonlar oluşturan bir endüstri standardıdır. En yenisi USB-C olan 14 farklı konektör türü dahil olmak üzere çok çeşitli USB donanımı mevcuttur.

  İlk olarak 1996'da piyasaya sürülen USB standartları, USB Implementers Forum (USB-IF) tarafından sürdürülür. Dört nesil USB şunlardır: USB 1.x, USB 2.0, USB 3.x ve USB4.

  Bir USB sistemi, bir veya daha fazla aşağı akış bağlantı noktasına sahip bir ana bilgisayardan ve katmanlı bir yıldız topolojisi oluşturan birden çok çevre biriminden oluşur. Beş katmana kadar izin veren ek USB hub'ları dahil edilebilir. Bir USB ana bilgisayarı, her biri bir veya daha fazla bağlantı noktasına sahip birden fazla denetleyiciye sahip olabilir. Tek bir ana bilgisayar denetleyicisine 127 adede kadar cihaz bağlanabilir.[45][24]:8–29 USB cihazları, hub'lar aracılığıyla seri olarak bağlanır. Ana bilgisayar denetleyicisinde yerleşik olan hub, kök hub olarak adlandırılır.

  Bir USB aygıtı, aygıt işlevleri olarak adlandırılan birkaç mantıksal alt aygıttan oluşabilir. Bir bileşik aygıt, örneğin yerleşik bir mikrofona (ses aygıtı işlevi) sahip bir web kamerası (video aygıtı işlevi) gibi çeşitli işlevler sağlayabilir. Buna bir alternatif, ana bilgisayarın her mantıksal aygıta ayrı bir adres atadığı ve tüm mantıksal aygıtların fiziksel USB kablosuna bağlanan yerleşik bir hub'a bağlandığı bileşik bir aygıttır.



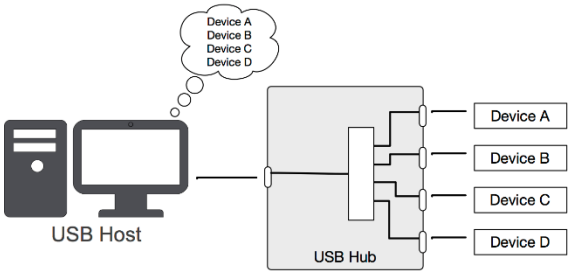
Şekil 1: USB örnek bağlantı tipleri

- **USB MASS STORAGE CLASS (MSC)**

  USB yığın depolama aygıtı sınıfı (MSC veya UMS), depolama aygıtlarına bağlantıları standart hale getirir. İlk başta manyetik ve optik sürücüler için tasarlanmıştı, flash sürücüleri destekleyecek şekilde genişletildi. Aynı zamanda, birçok sistem, dizinler içinde bilinen dosya işleme metaforu ile kontrol edilebildiğinden, çok çeşitli yeni cihazları destekleyecek şekilde genişletilmiştir. Yeni bir aygıtı tanıdık bir aygıta benzetme işlemi, uzantı olarak da bilinir. Yazma kilitli bir SD kartı bir USB adaptörüyle başlatma özelliği, önyükleme ortamının bütünlüğünü ve bozulmaz, bozulmamış durumunu korumak için özellikle avantajlıdır.

- **USB HOST**

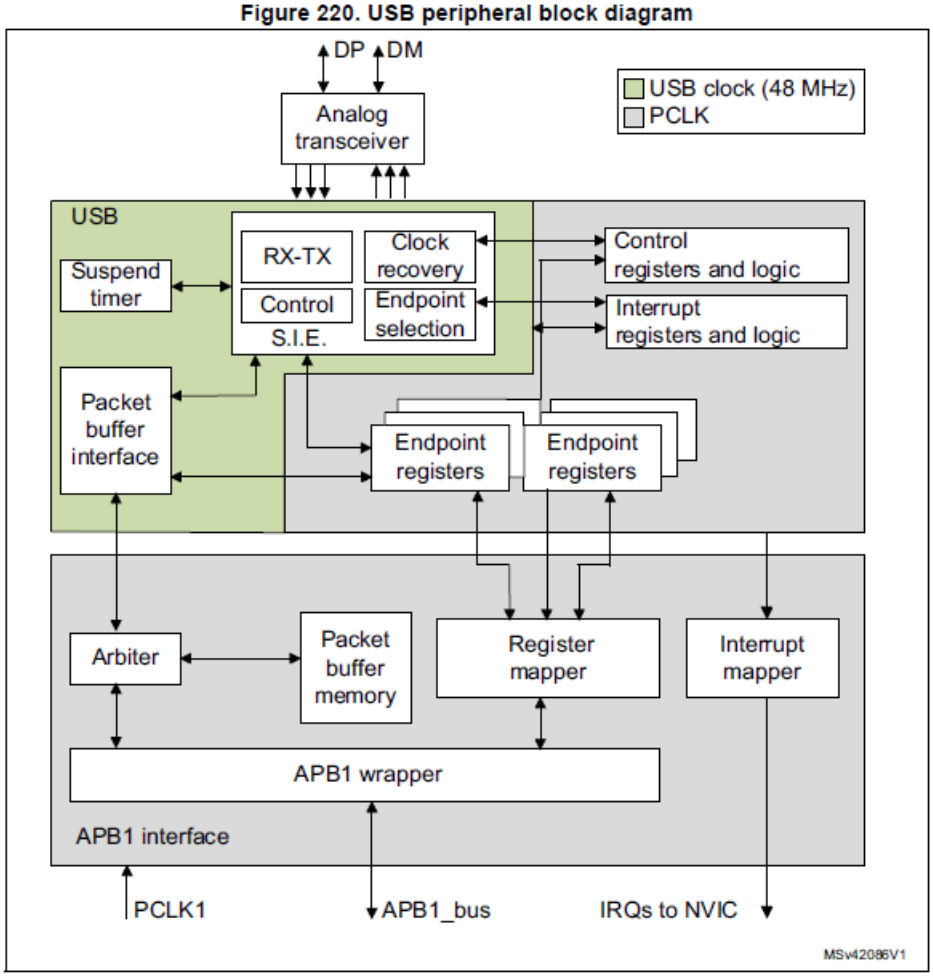
  Diğer "istemci" çevre birimlerinin bağlanabileceği bir "ana" USB aygıtı olarak hizmet edebilen bir aygıta "**USB ana bilgisayarı**" denir. Bilgisayarlar varsayılan olarak USB ana bilgisayarlarıdır. Diğer bazı cihazlar ayrıca USB ana bilgisayarları olarak da hizmet verebilir veya çevre birimleri için bir bağlantı noktası olarak hizmet etmelerini sağlayan bir "USB ana bilgisayar moduna" geçirilebilir. Bir örnek, veri aktarımı, yedekleme vb. için bir bilgisayara bağlanabilen, ancak aynı zamanda istemci çevre birimlerine bir USB ana bilgisayarı olarak da hizmet edebilen bir tablettir.



Şekil 2 : USB Host ve Device örnek şeması

Reference Manual incelenirse USB peripheral blok diyagramı aşağıdaki gibi olacaktır.

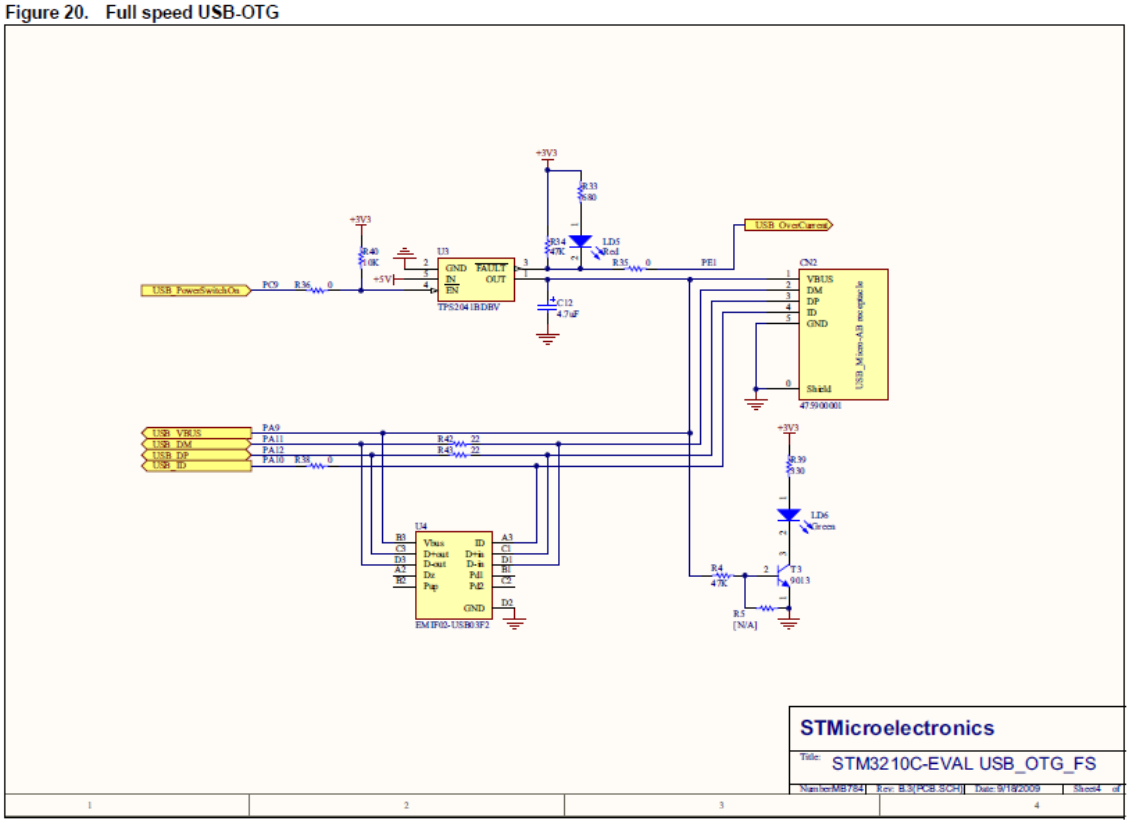
Figure 220. USB peripheral block diagram



Şekil 3 : USB Peripheral Blok Diyagramı

Projemizde kullanılacak olan STM3210C-EVAL kartını incelersek, CN2 konnektöründe bir USB Micro-A girişi bulunmaktadır. Diyagramı aşağıda verilmiştir.

Figure 20. Full speed USB-OTG



Şekil 4 : STM3210C-EVAL Full-Speed USB-OTG Şeması

Burada PC9 pini VBUS aktif edilmesi için önemli rol alacaktır. Bunu pin düzenini gösterirken anlatacağız.

3. Gelişme

Öncelikle projemizi oluşturmak için bizim için oldukça yararlı olacak olan STM32CubeIDE belgelerinde Repository kısmındaki STM3210C-EVAL USB HOST MSC Standalone örneğini inceledik.

Fakat proje dosyası elimizde mevcut olan kartta çalışmıyordu. Hatayı ayıklamak amaçlı DEBUG modunda işlemleri incelediğimizde, programın sürekli `USBH_Process` fonksiyonu içinde döndüğünde `USBH_FAIL` durumuna ulaştığını gördük. Deneyimimiz sonucu, çıkarımımızın örnek programın mevcut kartımızla uyumlu çalışmadığı yönündeydi.

Başka örneklerden faydalanmak için farklı kaynaklardan araştırmalarda bulunduk. Daha sonra araştırmalarımız sonucu, STM32-Discovery kartı için denenmiş bir örneği kendi kartımıza uyarlamaya çalıştık. Bu denememizde, amacımız ilk olarak USB Flash belleğin takıldığında LED1 ledinin yanmasıydı. Programı karta aktardığımızda, bahsedilen led USB MSC aygıtını karta bağladığımızda yanıyordu. Daha sonra, USB

MSC aygıtımıza okuma yazma işlemi için UsbTest_Write ve UsbTest_Read fonksiyonlarını yazdık. Daha sonra bunları, USB durumunu kontrol eden Application_State_Check() fonksiyonuna gerekli durumlara göre gerçekleşecek şekilde ekledik. Ve ana döngüde kullandık. Programı çalıştırdığımızda, LED yanıyordu fakat okuma yazma işlemi yapılamıyordu. DEBUG modunda hata ayıklama yaptığımızda, Application_State_Check() fonksiyonundaki `Appli_state` değişkeni hiçbir zaman APPLICATION_READY durumuna geçmiyordu. Programın çalışabilmesi için buraya geçebilmesi gerekliydi.

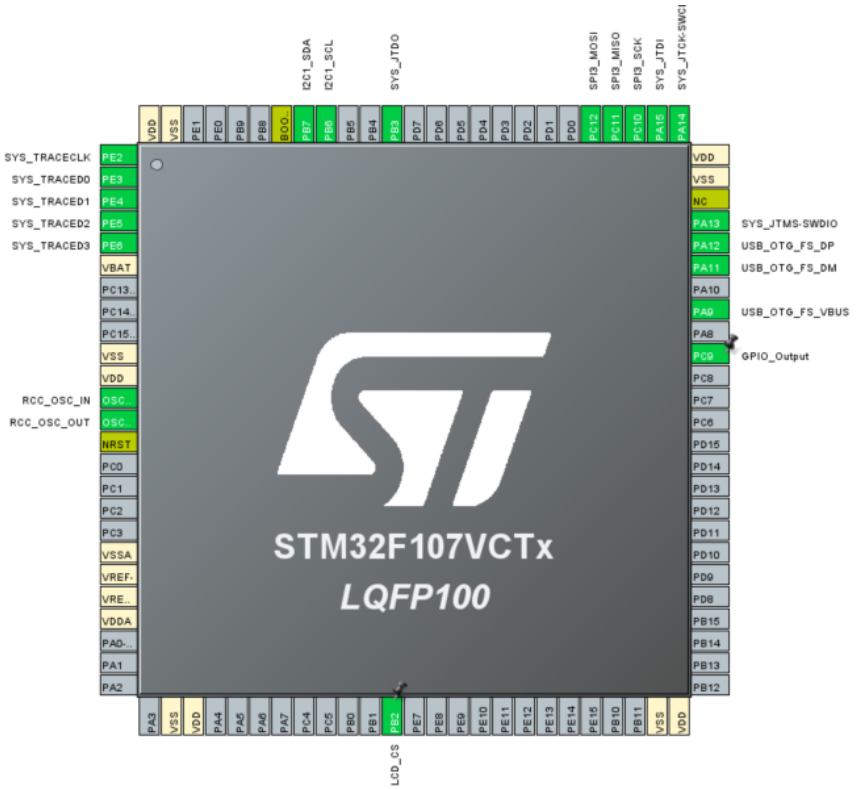
Bu denememizde de başarısız olduktan sonra, tekrar araştırmalırımız sonucunda, STM3210C-EVAL kartı için oluşturulmuş başka bir USB Host MSC örneği bulduk. Bu örnek de kartımızda çalışmadı. En son olarak bu örneği de DEBUG ettiğimizde, programın sonsuz loop'a girdiğini gözlemledik.

Başarısız denemelirimiz sonucu, bütün denemelerimizin başarısız olmasından dolayı, kartımızın USB girişinde bir problem olabileceği kanaatine vardık. Danışmanımızla bu konuyu görüştükten sonra, aynı uygulamaların farklı bir kartta çalıştığını tespit ettik, ve sonuç olarak programları denediğimiz kartın USB girişinde arıza olduğu sonucuna vardık.

Kartımızı yeni kartımızla değiştirdik ve tekrar örnekleri incelemeye başladık. Bütün örnekler sorunsuz bir şekilde çalışıyordu. İncelediğimiz örnekler ve oluşturduğumuz kendi programımız sonucunda, projeyi uygulamaya geçirdik.

- Proje düzeni

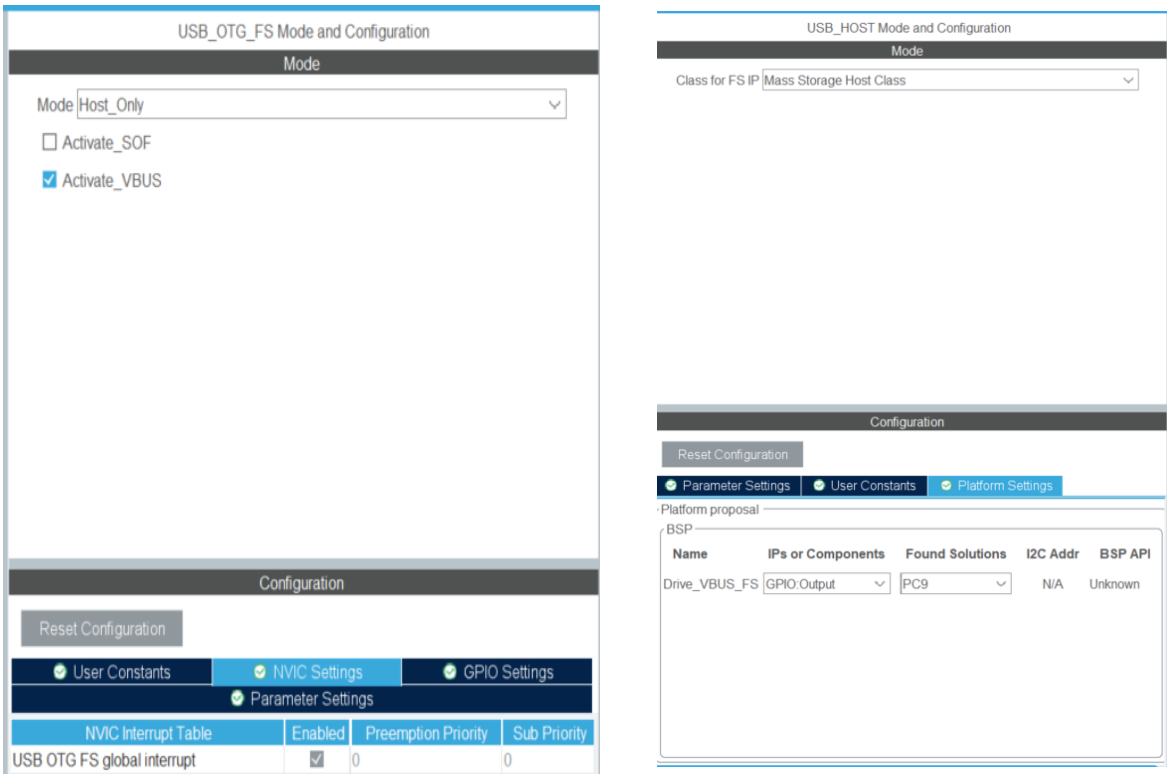
  Projemizin PIN Configuration ayarları aşağıdaki gibidir.



Şekil 5 : PIN Konfigürasyonu

Burada, PC9 pini, GPIO_Output olarak seçilmiştir. Bunun nedeni, USB VBUS aktive edilmesi için gereklidir (*Bkz: Şekil 4*).

Connectivity sekmesinden USB_OTG_FS, Middleware sekmesinden USB Host modu ayarlanmıştır.

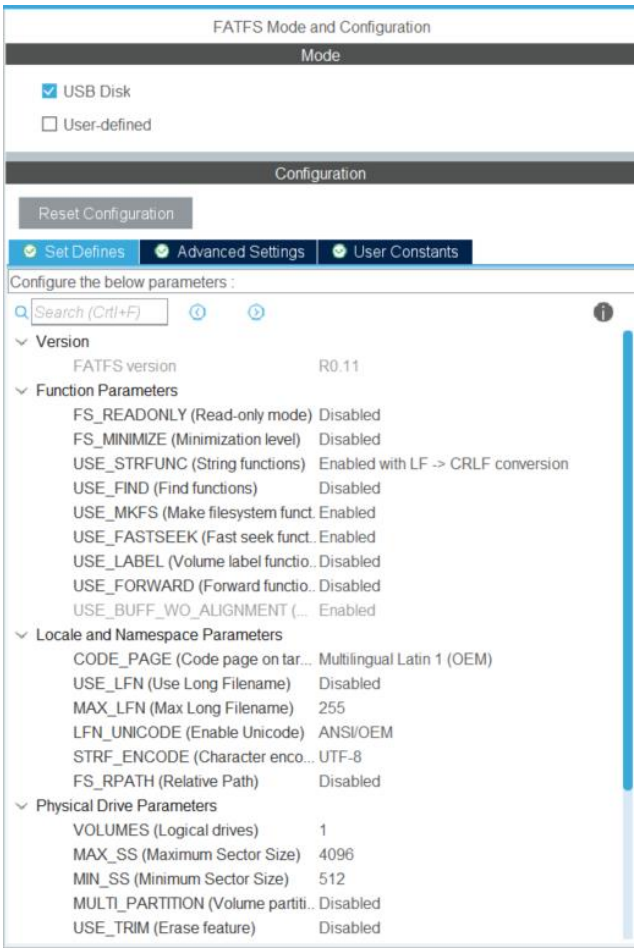


Şekil 6 : USB_OTG_FS ve USB_HOST ayarlamaları

Okuma yazma için, FATFS kütüphanesi kullanılacaktır. Middleware sekmesinden ayarlanmıştır.

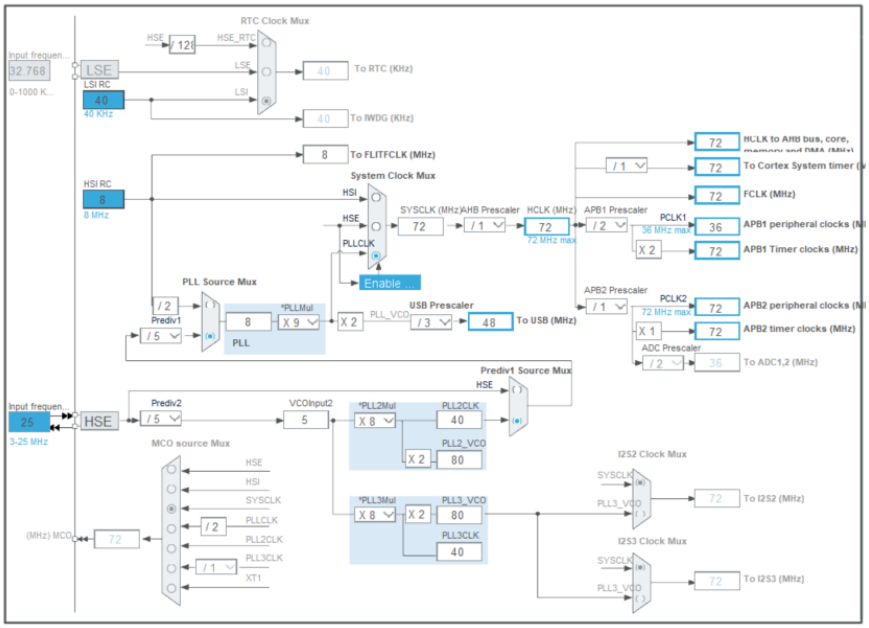
- FATFS

FatFs, küçük gömülü sistemler için genel bir FAT/exFAT dosya sistemi modülüdür. FatFs modülü, ANSI C (C89) ile uyumlu olarak yazılmıştır ve disk G/Ç katmanından tamamen ayrılmıştır. Bu nedenle platformdan bağımsızdır. 8051, PIC, AVR, ARM, Z80, RX vb. gibi sınırlı kaynağa sahip küçük mikro denetleyicilere dahil edilebilir. Ayrıca küçük mikro denetleyiciler için Petit FatFs modülü burada mevcuttur.



Şekil 7 : FATFS ayarlamaları

Clock Konfigürasyonu aşağıdaki gibi ayarlanmıştır.



Şekil 8 : Clock konfigürasyonu

- Fonksiyonlar ve tanımlar

Programımızın **main.c** dosyasında, aşağıdaki tanımları yaptık.

```
/* USER CODE BEGIN PV */
//uint16_t x, y;
JOYState_TypeDef JoyStick;
uint8_t status = 0;
uint8_t count_idle = 0;
uint8_t i = 1;
char i_str[100];
extern ApplicationTypeDef Appli_state;
//FATFS varibale
FATFS myUsbFatFS;
//USB Logical path
extern char USBHPath[4]; /* USBH logical drive path */

//File IO Variables
FIL myFile;
FRESULT res;
UINT byteswritten, bytesread;
char rwtext[100];  //Read/Write buf
FILE *fp;
TCHAR file_name[100];
/* USER CODE END PV */
```

Daha sonra bu tanımları gerekli fonksiyonlarda kullanacağız. Programımızın main() fonksiyonu aşağıdaki gibidir.

```
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_I2C1_Init();
  MX_SPI3_Init();
  MX_FATFS_Init();
  MX_USB_HOST_Init();
```

```
  /* USER CODE BEGIN 2 */
  USB_InitApplication();

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
      while (1) {
    /* USER CODE END WHILE */
    MX_USB_HOST_Process();

    /* USER CODE BEGIN 3 */
    Application_State_Check();
      }
  /* USER CODE END 3 */
}
```

Buradaki fonksiyonları açıklayalım.

- Başlangıç fonksiyonları

  Aşağıdaki fonksiyonlar pin konfigürasyonu ve gerekli ayarlamaları yaptığımızda program başlatıldığında otomatik olarak kurulumu sağlayan fonksiyonlardır. Bu fonksiyonlarda herhangi bir değişiklikte bulunulmamıştır.

  ```
  MX_GPIO_Init();
  MX_I2C1_Init();
  MX_SPI3_Init();
  MX_FATFS_Init();
  MX_USB_HOST_Init();
  ```

  Buradan sonraki **USB_InitApplication()** fonksiyonu, LCD ekran ve LED ayarlamalarını yapmaktadır.

  While(1) döngüsüne girdiğimizde, iki fonksiyon çalışmaktadır. **MX_USB_HOST_Process** fonksiyonu, USB arkaplan işlemlerini yürütmektedir. Burada da herhangi bir değişiklik yapılmamıştır. Fakat bu fonksiyonun içinden ilerleyerek usbh_core.c dosyasına ulaştığımızda, burada USBH_HandleEnum() fonksiyonu bulunmaktadır. Bu fonksiyonda Enumeration işlemi yapılmaktadır.

- Enumeration : USB Enumeration, bir USB aygıtı için sürücüleri algılama, tanımlama ve yükleme işlemidir. Bu, bir şeyin var olduğunu algılamak için donanım tekniklerinin ve neyin bağlı olduğunu belirlemek için yazılımın bir karışımını içerir. Bu belgenin amacı, sürecin mekaniğine genel bir bakış sağlamaktır.

  Burada, USB aygıtının tanımlamaları yapıldığından, gerekli tanımlar ekranda gösterilmesi için LCD Log kütüphanesinden yararlanılarak, USB MSC aygıtının bilgilerinin ekrana bastırılması sağlanmıştır. Örneğin; 900. Satır incelendiğinde aşağıdaki fonksiyon, USB MSC aygıtının üreticisinin isminin ekrana bastırılmasını sağlamaktadır.

```
LCD_UsrLog("Manufacturer : %s\n", (char *)(void *)phost->device.Data);bool
```

Application_State_Check() fonksiyonu, programın genel çalışmasını kapsamaktadır.

```
case APPLICATION_READY:
```

`MSC_MenuProcess()` durumu olduğunda, MSC_MenuProcess() fonksiyonuna gitmektedir. İşlemler için ayriyeten bir **menu.c** dosyası oluşturulmuştur. Fonksiyon incelendiğinde, buradan **Key_press()** fonksiyonuna gittiğini görürüz. Bu fonksiyon ise, programımızın çalışma mantığını oluşturmaktadır.

- Programın çalışma mantığı

Programımızın çalışma mantığını kurduğumuzda, ilk olarak joystick ile idare edilebilecek şekilde kurmak istedik. Amacımız, hem usb bellek içindeki dosyası sistem içinde joystick ile gezebilmeyi sağlamak hem de içine örnek bir txt dosyası yazdırabilmekti. Hatta, bu dosya sisteminde txt ve bmp olan istediğimiz dosyayı seçtiğimizde açabilme yeteneğini sağlamaktı.
Joystick ile kontrol etmeyi denediğimizde, programımızın kitlendiğini gözledik. Bu yüzden joystick fonksiyonunu kullanamadık. Resim bastırma ve dosya sisteminde gezinme uygulamalarını da, proje süresi kısıtlı olduğundan dolayı, ayrıntılı bir şekilde yapamadık.

Daha sonra, biz de incelediğimiz örneklerden birine benzer uygulama yapmaya çalıştık. Kısaca, USB bellek takıldığında, ekran bize takıldı bilgisini ve USB belleğin bilgilerini gösterecek. Projede Key tuşunu kullanarak sırayla,

- **Bir txt dosyası yazma**
- **Disk content'ini görüntüleme**
- **Son oluşturulan txt dosyasını okuma**

İşlemleri yapılabilmektedir. Daha sonra işlemler başa dönmekte, her seferinde yeni bir txt dosyası oluşturulmaktadır.

Projede örnek uygulamalardaki birkaç fonksiyondan yararlandık ve geri kalan kısımda kendi fonksiyon ve algoritmalarımızı kullandık.

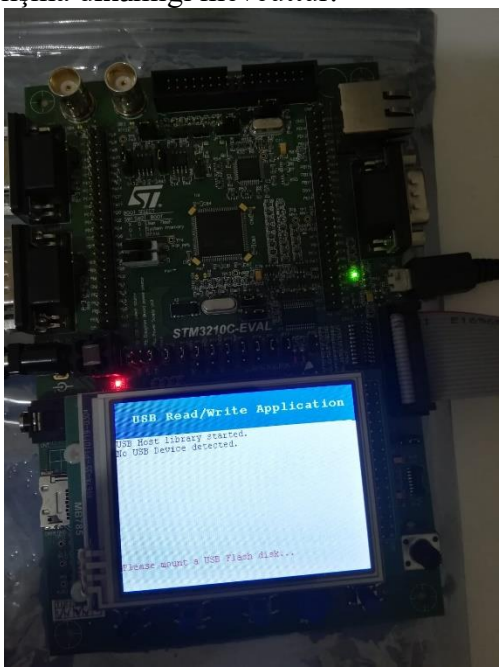
Txt dosyası yazma, Disk content görüntüleme ve dosya okuma uygulamarı için aşağıdaki UsbTest_Write(), UsbTest_Read() ve Explore_Disk() fonksiyonlarını kullandık.

```
UsbTest_Write(void)
{
        //Open or Create file for writing
        strcpy(file_name,"TEST");
        sprintf(i_str, "%d", i);
        strcat(file_name,i_str);
        strcat(file_name,".TXT");
        if(f_open(&myFile,(TCHAR*)file_name,FA_WRITE | FA_CREATE_ALWAYS) != FR_OK)
        {
                return 0;
        }
        //Copy test Text to my temporary read/write buffer
        sprintf(rwtext, "TEST%d.TXT -> Bu bir test dosyasidir.",i);
        //Write to text file
        res = f_write(&myFile, (const void *)rwtext, strlen(rwtext), &byteswritten);
        if((res != FR_OK) || (byteswritten == 0))
        {
```

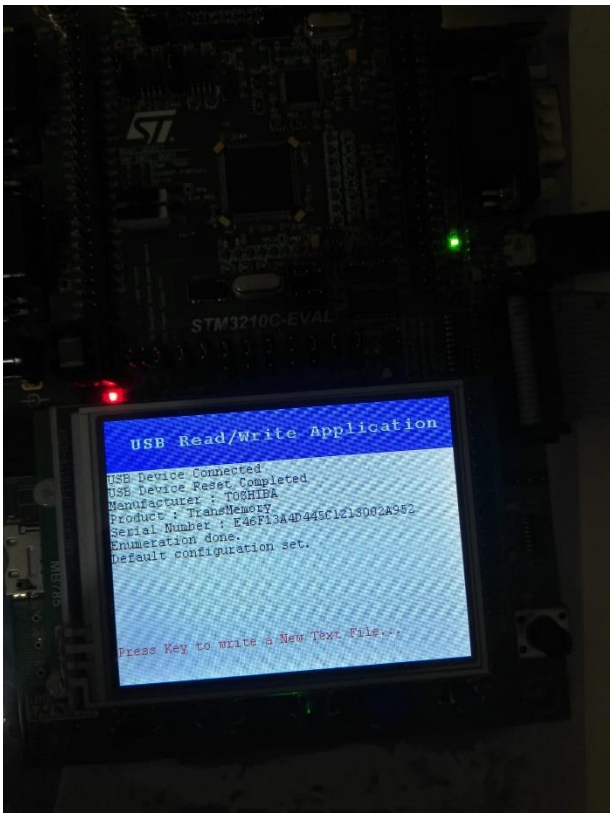
```
                return 0;
        }
        f_close(&myFile);
        return 1; //Success
}

//2. USB test Read function
bool UsbTest_Read(void)
{
                strcpy(file_name,"TEST");
                sprintf(i_str, "%d", i);
                strcat(file_name,i_str);
                strcat(file_name,".TXT");
                //Open file for reading
                if(f_open(&myFile, (TCHAR*)file_name, FA_READ) != FR_OK)
                {
                        return 0;
                }

                //Read text from files until NULL
                for(uint8_t i=0; i<100; i++)
                {
                        res = f_read(&myFile, (uint8_t*)&rwtext[i], 1, &bytesread);
                        if(rwtext[i] == 0x00) // NULL string
                        {
                                bytesread = i;
                                break;
                        }
                }
                LCD_UsrLog((const void *)rwtext);
                LCD_UsrLog("\n");
                i++;
                //Reading error handling
                if(bytesread==0) return 0;

                //Close file
                f_close(&myFile);
        return 1;  // success

        }

static uint8_t Explore_Disk(char *path, uint8_t recu_level)
{

  FRESULT res;
  FILINFO fno;
  DIR dir;
  char *fn;
  char tmp[14];

  res = f_opendir(&dir, path);
  if (res == FR_OK)
  {
    while (Appli_state == APPLICATION_READY)
    {
      res = f_readdir(&dir, &fno);
      if (res != FR_OK || fno.fname[0] == 0)
      {
        break;
      }
      if (fno.fname[0] == '.')
      {
        continue;
      }

      fn = fno.fname;
      strcpy(tmp, fn);
```

```
      line_idx++;
      if (line_idx > 9)
      {
        line_idx = 0;
      }

      if (recu_level == 1)
      {
        LCD_UsrLog("   |__");
      }
      else if (recu_level == 2)
      {
        LCD_UsrLog("   |   |__");
      }
      if ((fno.fattrib & AM_MASK) == AM_DIR)
      {
        strcat(tmp, "\n");
        LCD_UsrLog((void *)tmp);
      }
      else
      {
        strcat(tmp, "\n");
        LCD_UsrLog((void *)tmp);
      }

      if (((fno.fattrib & AM_MASK) == AM_DIR) && (recu_level == 1))
      {
        Explore_Disk(fn, 2);
      }
    }
  }
  return res;
```

Programımızı derledikten sonra kartımıza yükledik. Projemizde, aşağıdaki ekran görüntülerindeki gibi çalışma dinamiği mevcuttur.



(a)


STM3210C-EVAL
USB Read/Write Application
USB Device Connected
USB Device Reset Completed
Manufacturer : TOSHIBA
Product : TransMemory
Serial Number : E46F13A4D445C1213002A952
Enumeration done.
Default configuration set.
Press Key to write a New Text File...

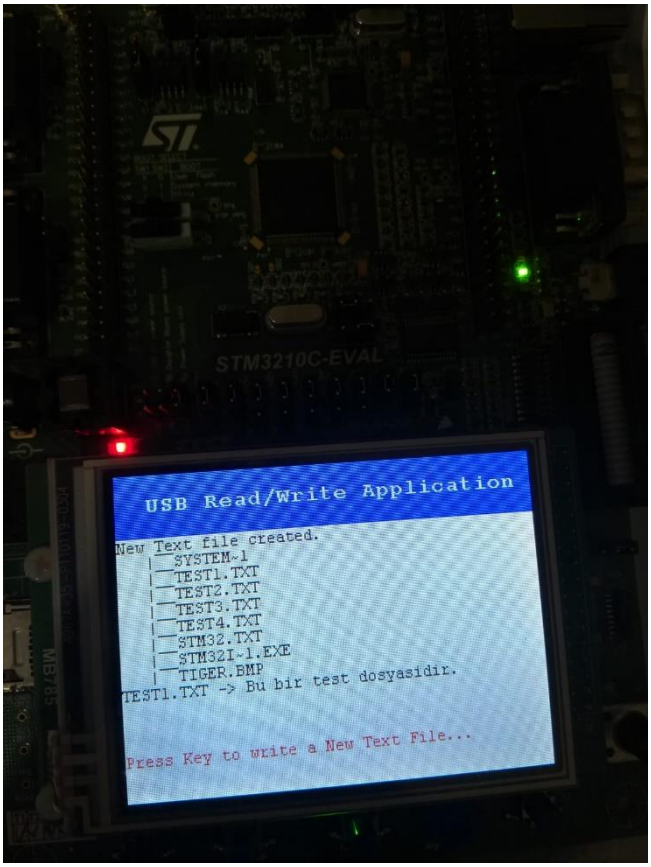
(b)


STM3210C-EVAL
USB Read/Write Application
USB Device Connected
USB Device Reset Completed
Manufacturer : TOSHIBA
Product : TransMemory
Serial Number : E46F13A4D445C1213002A952
Enumeration done.
Default configuration set.
New Text file created.
Press Key to Explore Disk Content...

(c)


STM3210C-EVAL
USB Read/Write Application
Default configuration set.
New Text file created.
|__SYSTEM~1
|__TEST1.TXT
|__TEST2.TXT
|__TEST3.TXT
|__TEST4.TXT
|__STM32.TXT
|__STM32I~1.EXE
|__TIGER.BMP
Press Key to read last Text File....

(d)


STM3210C-EVAL
USB Read/Write Application
New Text file created.
|__SYSTEM~1
|__TEST1.TXT
|__TEST2.TXT
|__TEST3.TXT
|__TEST4.TXT
|__STM32.TXT
|__STM32I~1.EXE
|__TIGER.BMP
TEST1.TXT -> Bu bir test dosyasidir.
Press Key to write a New Text File...

(e)

(f)

Şekil 9 : Program çalışma adımları
(a) USB bellek takılı değil durumu (b) USB bellek ilk takıldığındaki durum (c) Yazma işlemi (d) Disk içeriğini görüntüleme (e) Okuma işlemi (f) USB bellek çıkarıldı durumu

4. Sonuç

Projede karşılaştığımız genel sorun, ilk elimizde bulunan STM3210C-EVAL kartının USB girişinin bozuk olmasıydı. Bozuk olduğundan ötürü, incelediğimiz örneklerde sorun olduğunu düşünerek, fazla vakit kaybettik. Eğer girişin bozuk olduğu daha erken farkedilseydi, daha gelişmiş bir uygulama ortaya çıkarabilirdik.

Genel olarak incelediğimizde, proje bize önemli ölçüde bilgi ve deneyim kattı diyebiliriz. USB çalışma mantığını, USB Host ve Device farkı ve uygulamaları, MSC, HID vs. kavramların anlamları ve uygulamarını inceledik. Projemiz basit kalsa da, gerekli vakit ve tecürebede daha iyi bir uygulama çıkartabileceğimizi düşünüyoruz.

5. Referanslar

[1] RM0008 Reference Manual

[2] UM0600 User Manual

[3] https://www.st.com/resource/en/user_manual/dm00105256-stm32cube-usb-host-library-stmicroelectronics.pdf

[4] https://controllerstech.com/stm32-usb-msc/

[5] https://www.electronics-notes.com/articles/connectivity/usb-universal-serial-bus/basics-tutorial.php