Elektronik ve Haberleşme Mühendisliği

Gömülü Sistemlerde Yazılım ve Donanım Tasarımı Dersi

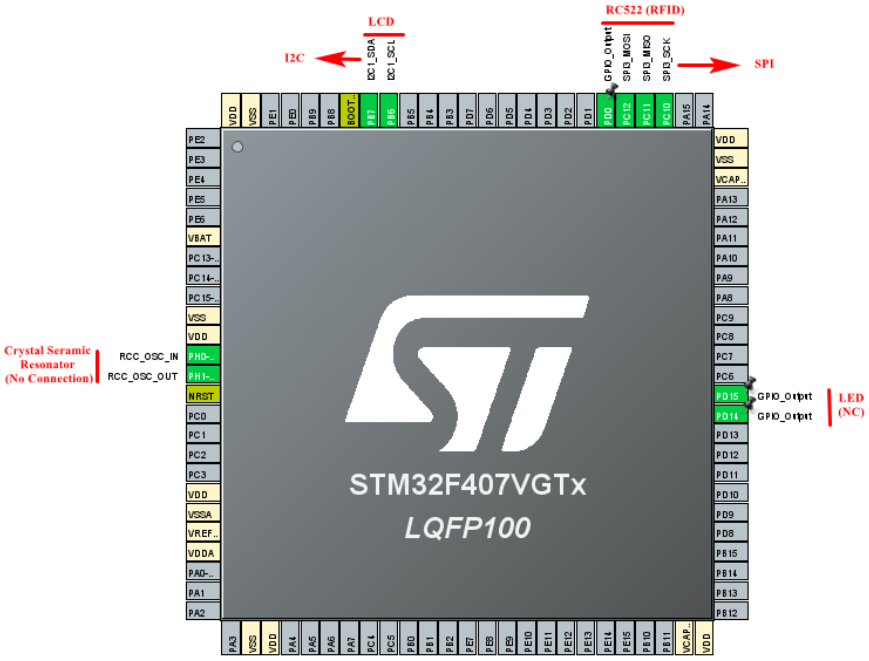
STM32 ile RFID Sistemi Yazılım Tasarımı Proje Raporu

18014125 Alirıza BİLİR

İSTANBUL, 2023

Pin Konfigürasyonu

STM32CubeIDE:



Cube IDE'de bulunan ioc tanımlama ekranında gösterilen pinler bağlantıya uygun şekilde özelleştirilmiş ve seçilmiştir. PD14 ve PD15 pinleri kart okuma esnasında bilgi vermesi için geliştirme kartında bulunan ledleri belirtmektedir. PD7 ve PD6 pinleri lcd ekran ile i2c haberleşme protokolü için kullanılacak pinlerdir. PD0 output pini seçilerek rc522'nin slave select pini olması için ayrıca kütüphanelerde ve driverda adres olarak belirtilmiştir. Diğer PC12 MOSI (Master out slave input) , PC11 MISO , PC10 SCK(Serial Clock) pinleri ise spi protoküle uygun olarak seçilmiştir.

Bağlantı Şeması:


STM32F407VG

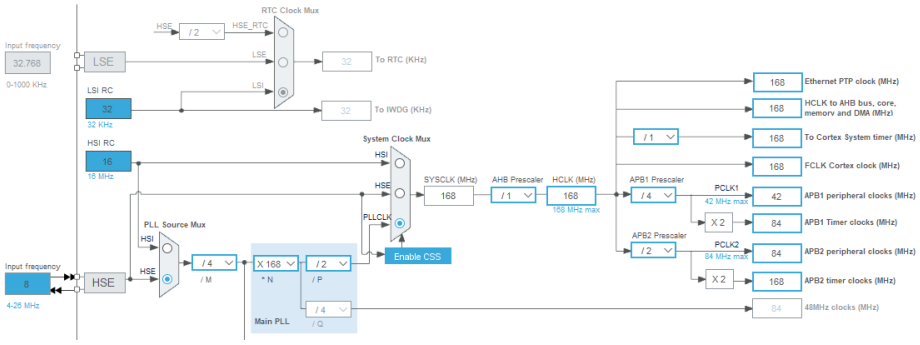
RC522

LCD I2C Arayüzü (KK245)

SDA(PB7)		SDA
SCL(PB6)		
SS (PD0)	SDA	SCL
SCK (PC10)	SCK	
MOSI(PC12)	MOSI	
MISO(PC11)	MISO	
3.3V	3.3V , RST	VCC
GND	GND	GND
	IRQ (NC)	

LCD ekranla daha kolay haberleşebilmesi için KK245 i2c arayüzü modülü ayrıca dahil edilmiş olarak 16 pin sayısı 4'e düşürülmüştür. Yukarıdaki görselde pin giriş ve çıkışları belirtilmiştir.

Clock Konfigürasyonu



1. Kristal Frekansı (8 MHz):
    - 8 MHz, mikrodenetleyiciler için yaygın bir kristal frekansıdır. Standart ve geniş bir kullanıma sahip olduğu için tercih edilir.
    - Bu frekans, birçok uygulama için yeterli performansı sağlar.
    - Tasarım gereksinimleri ve donanım uyumluluğu göz önüne alındığında optimal bir değerdir.
2. HCLK Frekansı (168 MHz):
    - HCLK frekansı, mikrodenetleyicinin işlem gücünü belirler. Yüksek HCLK frekansları, mikrodenetleyicinin daha hızlı işlemler gerçekleştirmesine olanak tanır.
    - 168 MHz, yüksek performanslı uygulamalarda kullanılmak üzere seçilmiş olabilir. Bu, grafik kullanıcı arayüzleri, sıkıştırma algoritmaları veya diğer işlem gücü yoğun uygulamalar için önemlidir.
    - Ancak, yüksek frekanslar enerji tüketimini ve ısıyı artırabilir, bu nedenle enerji verimliliği ve ısı yönetimi göz önünde bulundurulmalıdır.

Kodlar:

Kütüphaneler:

fonts.h

EmbeddedProject_RFID.ioc | main.c | fonts.h | i2c-lcd.h | rc522.h | ssd1306.h

```
#include "stm32f4xx_hal.h"

#ifndef Fonts
#define Fonts


typedef struct {
    const uint8_t FontWidth;    /*!< Font width in pixels */
    uint8_t FontHeight;         /*!< Font height in pixels */
    const uint16_t *data;       /*!< Pointer to data font data array */
} FontDef;



extern FontDef Font_7x10;
extern FontDef Font_11x18;
extern FontDef Font_16x26;

#endif
```

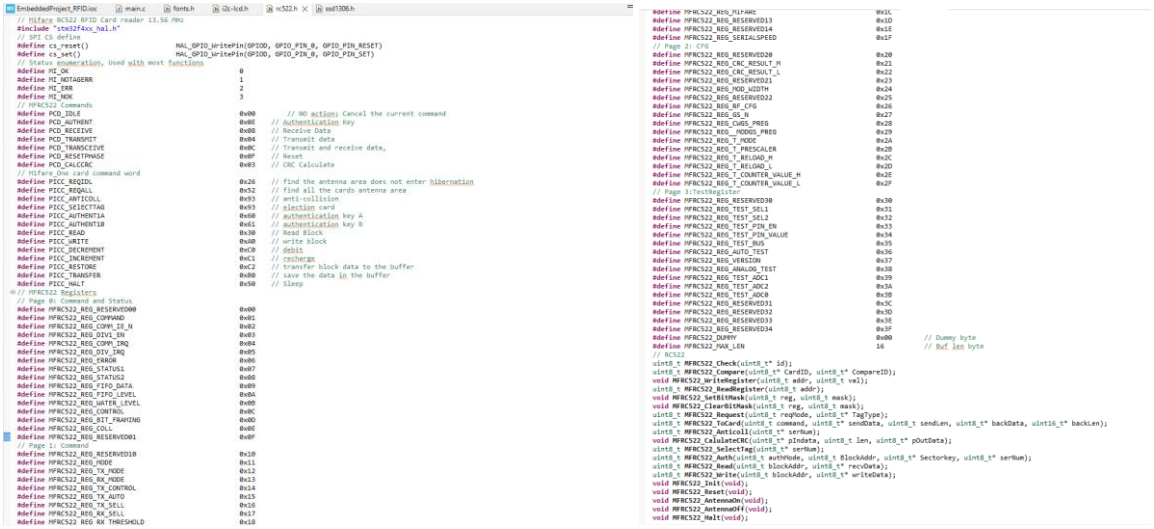
Bu kütüphanede belirtilen 3 farklı font tipi ve yapısı oluşturulmuştur.

i2c-lcd.h

```c
#include "stm32f4xx_hal.h"

void lcd_init (void);   // initialize lcd

void lcd_send_cmd (char cmd);  // send command to the lcd

void lcd_send_data (char data);  // send data to the lcd

void lcd_send_string (char *str);  // send string to the lcd
```

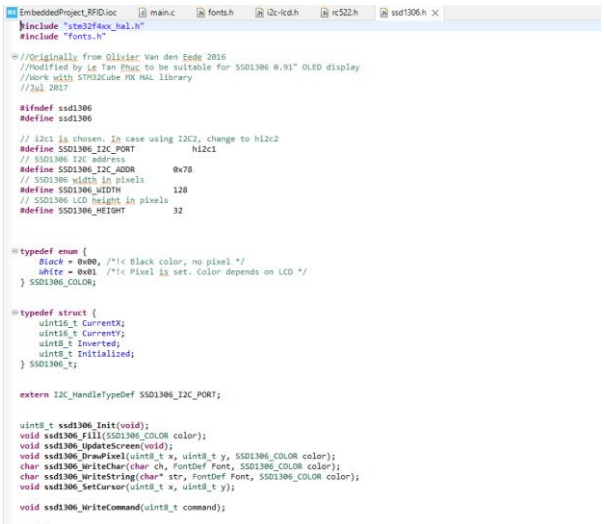
Driver dosyasında bulunan fonksiyonlar, bu kütüphanede deklare edilmiştir.

rc522.h



Slave select yapılması için bulunan SDA pinine bağlanacak olan PD0 pini burada belirtilmiş olup adreslemeler de ayrıca burada bulunmaktadır.

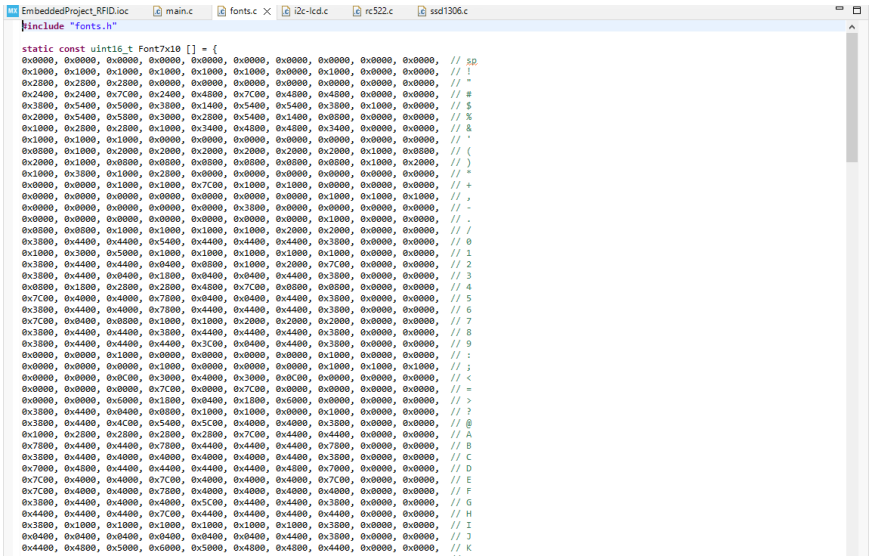
ssd1306.h



i2c portu ve lcd'ye ait tanımalar bulunmaktadır.

Sürücüler ve İçerikleri:

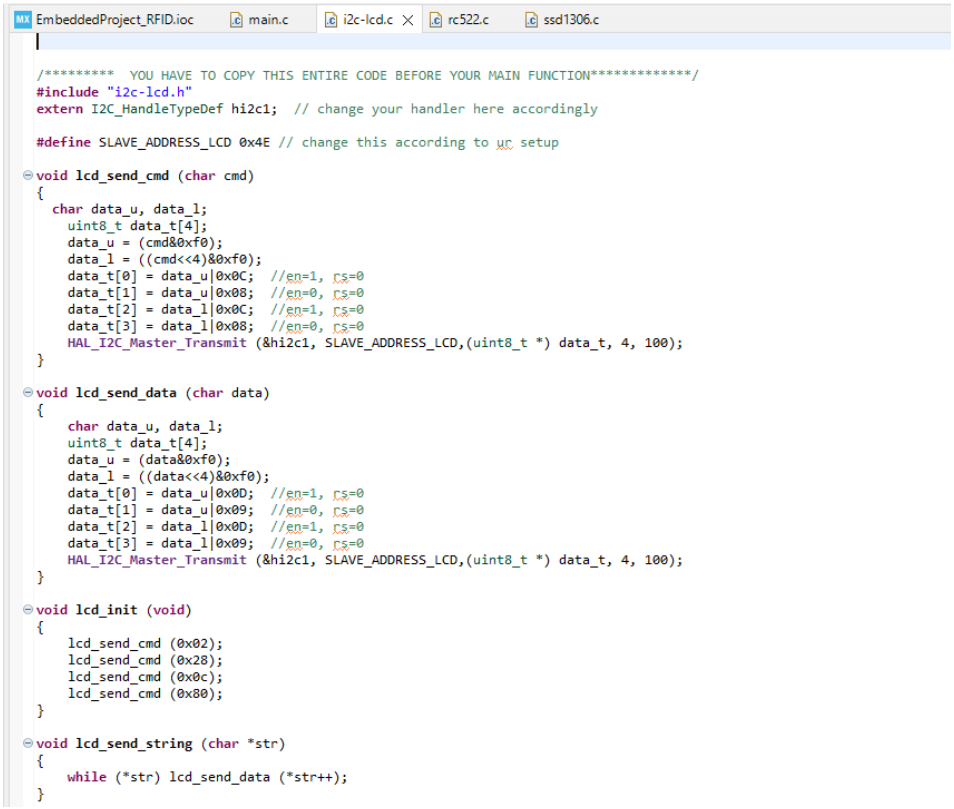
fonts.c



```
#include "fonts.h"

static const uint16_t Font7x10 [] = {
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,  // sp
0x1000, 0x1000, 0x1000, 0x1000, 0x1000, 0x1000, 0x0000, 0x1000, 0x0000, 0x0000,  // !
0x2800, 0x2800, 0x2800, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,  // "
0x2400, 0x2400, 0x7C00, 0x2400, 0x4800, 0x7C00, 0x4800, 0x4800, 0x0000, 0x0000,  // #
0x3800, 0x5400, 0x5000, 0x3800, 0x1400, 0x5400, 0x5400, 0x3800, 0x1000, 0x0000,  // $
0x2000, 0x5400, 0x5800, 0x3000, 0x2800, 0x5400, 0x1400, 0x0800, 0x0000, 0x0000,  // %
0x1000, 0x2800, 0x2800, 0x1000, 0x3400, 0x4800, 0x4800, 0x3400, 0x0000, 0x0000,  // &
0x1000, 0x1000, 0x1000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,  // '
0x0800, 0x1000, 0x2000, 0x2000, 0x2000, 0x2000, 0x2000, 0x2000, 0x1000, 0x0800,  // (
0x2000, 0x1000, 0x0800, 0x0800, 0x0800, 0x0800, 0x0800, 0x0800, 0x1000, 0x2000,  // )
0x1000, 0x3800, 0x1000, 0x2800, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,  // *
0x0000, 0x0000, 0x1000, 0x1000, 0x7C00, 0x1000, 0x1000, 0x0000, 0x0000, 0x0000,  // +
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x1000, 0x1000, 0x1000,  // ,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x3800, 0x0000, 0x0000, 0x0000, 0x0000,  // -
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x1000, 0x0000, 0x0000,  // .
0x0800, 0x0800, 0x1000, 0x1000, 0x1000, 0x1000, 0x2000, 0x2000, 0x0000, 0x0000,  // /
0x3800, 0x4400, 0x4400, 0x5400, 0x4400, 0x4400, 0x4400, 0x3800, 0x0000, 0x0000,  // 0
0x1000, 0x3000, 0x5000, 0x1000, 0x1000, 0x1000, 0x1000, 0x1000, 0x0000, 0x0000,  // 1
0x3800, 0x4400, 0x4400, 0x0400, 0x0800, 0x1000, 0x2000, 0x7C00, 0x0000, 0x0000,  // 2
0x3800, 0x4400, 0x0400, 0x1800, 0x0400, 0x0400, 0x4400, 0x3800, 0x0000, 0x0000,  // 3
0x0800, 0x1800, 0x2800, 0x2800, 0x4800, 0x7C00, 0x0800, 0x0800, 0x0000, 0x0000,  // 4
0x7C00, 0x4000, 0x4000, 0x7800, 0x0400, 0x0400, 0x4400, 0x3800, 0x0000, 0x0000,  // 5
0x3800, 0x4400, 0x4000, 0x7800, 0x4400, 0x4400, 0x4400, 0x3800, 0x0000, 0x0000,  // 6
0x7C00, 0x0400, 0x0800, 0x1000, 0x1000, 0x2000, 0x2000, 0x2000, 0x0000, 0x0000,  // 7
0x3800, 0x4400, 0x4400, 0x3800, 0x4400, 0x4400, 0x4400, 0x3800, 0x0000, 0x0000,  // 8
0x3800, 0x4400, 0x4400, 0x4400, 0x3C00, 0x0400, 0x4400, 0x3800, 0x0000, 0x0000,  // 9
0x0000, 0x0000, 0x1000, 0x0000, 0x0000, 0x0000, 0x0000, 0x1000, 0x0000, 0x0000,  // :
0x0000, 0x0000, 0x0000, 0x1000, 0x0000, 0x0000, 0x0000, 0x1000, 0x1000, 0x1000,  // ;
0x0000, 0x0000, 0x0C00, 0x3000, 0x4000, 0x3000, 0x0C00, 0x0000, 0x0000, 0x0000,  // <
0x0000, 0x0000, 0x0000, 0x7C00, 0x0000, 0x7C00, 0x0000, 0x0000, 0x0000, 0x0000,  // =
0x0000, 0x0000, 0x6000, 0x1800, 0x0400, 0x1800, 0x6000, 0x0000, 0x0000, 0x0000,  // >
0x3800, 0x4400, 0x0400, 0x0800, 0x1000, 0x1000, 0x0000, 0x1000, 0x0000, 0x0000,  // ?
0x3800, 0x4400, 0x4C00, 0x5400, 0x5C00, 0x4000, 0x4000, 0x3800, 0x0000, 0x0000,  // @
0x1000, 0x2800, 0x2800, 0x2800, 0x2800, 0x7C00, 0x4400, 0x4400, 0x0000, 0x0000,  // A
0x7800, 0x4400, 0x4400, 0x7800, 0x4400, 0x4400, 0x4400, 0x7800, 0x0000, 0x0000,  // B
0x3800, 0x4400, 0x4000, 0x4000, 0x4000, 0x4000, 0x4400, 0x3800, 0x0000, 0x0000,  // C
0x7000, 0x4800, 0x4400, 0x4400, 0x4400, 0x4400, 0x4800, 0x7000, 0x0000, 0x0000,  // D
0x7C00, 0x4000, 0x4000, 0x7C00, 0x4000, 0x4000, 0x4000, 0x7C00, 0x0000, 0x0000,  // E
0x7C00, 0x4000, 0x4000, 0x7800, 0x4000, 0x4000, 0x4000, 0x4000, 0x0000, 0x0000,  // F
0x3800, 0x4400, 0x4000, 0x4000, 0x5C00, 0x4400, 0x4400, 0x3800, 0x0000, 0x0000,  // G
0x4400, 0x4400, 0x4400, 0x7C00, 0x4400, 0x4400, 0x4400, 0x4400, 0x0000, 0x0000,  // H
0x3800, 0x1000, 0x1000, 0x1000, 0x1000, 0x1000, 0x1000, 0x3800, 0x0000, 0x0000,  // I
0x0400, 0x0400, 0x0400, 0x0400, 0x0400, 0x0400, 0x4400, 0x3800, 0x0000, 0x0000,  // J
0x4400, 0x4800, 0x5000, 0x6000, 0x5000, 0x4800, 0x4800, 0x4400, 0x0000, 0x0000,  // K
```

Bu dosya 3 farklı yazı büyüklüğüne için ayrı ayrı tüm ascii kodlarını bulundurmaktadır.

i2c-lcd.c



```
/*********  YOU HAVE TO COPY THIS ENTIRE CODE BEFORE YOUR MAIN FUNCTION**********/
#include "i2c-lcd.h"
extern I2C_HandleTypeDef hi2c1;  // change your handler here accordingly

#define SLAVE_ADDRESS_LCD 0x4E // change this according to ur setup

void lcd_send_cmd (char cmd)
{
  char data_u, data_l;
    uint8_t data_t[4];
    data_u = (cmd&0xf0);
    data_l = ((cmd<<4)&0xf0);
    data_t[0] = data_u|0x0C;  //en=1, rs=0
    data_t[1] = data_u|0x08;  //en=0, rs=0
    data_t[2] = data_l|0x0C;  //en=1, rs=0
    data_t[3] = data_l|0x08;  //en=0, rs=0
    HAL_I2C_Master_Transmit (&hi2c1, SLAVE_ADDRESS_LCD,(uint8_t *) data_t, 4, 100);
}

void lcd_send_data (char data)
{
    char data_u, data_l;
    uint8_t data_t[4];
    data_u = (data&0xf0);
    data_l = ((data<<4)&0xf0);
    data_t[0] = data_u|0x0D;  //en=1, rs=0
    data_t[1] = data_u|0x09;  //en=0, rs=0
    data_t[2] = data_l|0x0D;  //en=1, rs=0
    data_t[3] = data_l|0x09;  //en=0, rs=0
    HAL_I2C_Master_Transmit (&hi2c1, SLAVE_ADDRESS_LCD,(uint8_t *) data_t, 4, 100);
}

void lcd_init (void)
{
    lcd_send_cmd (0x02);
    lcd_send_cmd (0x28);
    lcd_send_cmd (0x0c);
    lcd_send_cmd (0x80);
}

void lcd_send_string (char *str)
{
    while (*str) lcd_send_data (*str++);
}
```

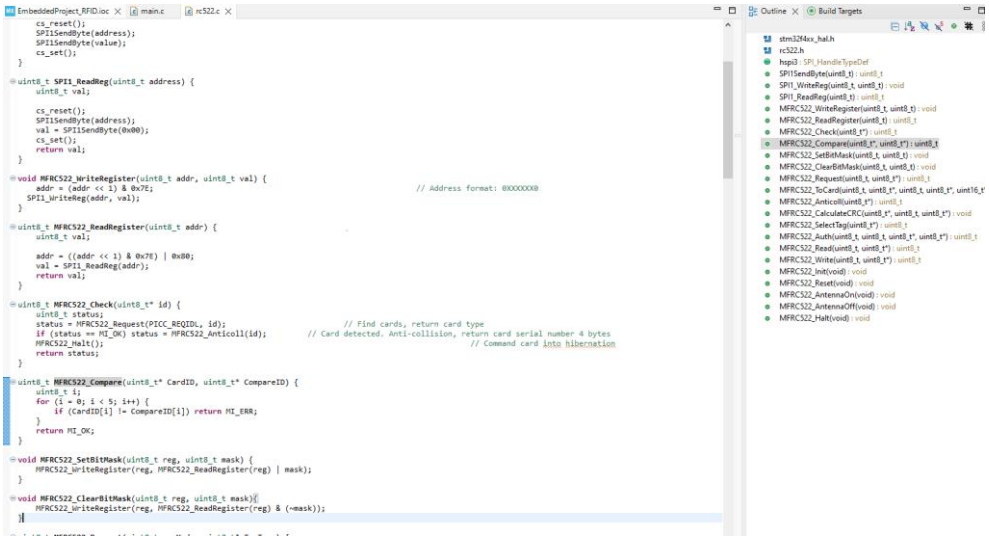
Yukarıdaki kod dizininde kullanılacak olan i2c portu ve adresi belirtilmektedir. Bu işlem slave select işlemi için gereklidir. Ayrıca gönderilecek olan data paketleri de kodun devamında bulunmaktadır.

ssd1306.c

```
char ssd1306_WriteChar(char ch, FontDef Font, SSD1306_COLOR color)
{
    uint32_t i, b, j;

    if (SSD1306_WIDTH <= (SSD1306.CurrentX + Font.FontWidth) ||
        SSD1306_HEIGHT <= (SSD1306.CurrentY + Font.FontHeight))
    {
        return 0;
    }

    for (i = 0; i < Font.FontHeight; i++)
    {
        b = Font.data[(ch - 32) * Font.FontHeight + i];
        for (j = 0; j < Font.FontWidth; j++)
        {
            if ((b << j) & 0x8000)
            {
                ssd1306_DrawPixel(SSD1306.CurrentX + j, (SSD1306.CurrentY + i), (SSD1306_COLOR) color);
            }
            else
            {
                ssd1306_DrawPixel(SSD1306.CurrentX + j, (SSD1306.CurrentY + i), (SSD1306_COLOR)!color);
            }
        }
    }

    SSD1306.CurrentX += Font.FontWidth;

    return ch;
}

/*
 Writing string function
 Params:
 - str: pointer to string
 - Font: font selection Font_7x10/Font_11x18/Font_16x26
 - color: Black/White
 */
char ssd1306_WriteString(char* str, FontDef Font, SSD1306_COLOR color)
{
    while (*str)
    {
        if (ssd1306_WriteChar(*str, Font, color) != *str)
        {
            return *str;
        }

        str++;
    }

    return *str;
}
```

LCD ekrana yazdırma fonksiyonları bulunan bu sürücü dosyasındaki birkaç fonksiyonun işlevi şunlardır: yazdırmak için başlangıç bitini seçmek, ekranı yenilemek, yazı yazdırmak.

rc522.c



Bu kod dosyasında rc522 modülünün tüm işlevleri bulunmaktadır. Bunlar kartların id'lerini okumak, hafızasındaki numaralarla kıyaslamak, karta farklı bilgiler yüklemek ve o bilgileri silebilmek gibi birçok fonksiyon içermektedir. Ayrıca driver'ın başlangıcında kullanılan spi kanalı ve adresi belirtilmektedir. Slave select işlevi ile gönderilecek data paketleri de belirtilmektedir.

main.c

```
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "rc522.h"
#include "i2c-lcd.h"
#include "ssd1306.h"
/* USER CODE END Includes */
```

```
/* Private variables ---------------------------------------------------------*/
CRC_HandleTypeDef hcrc;

I2C_HandleTypeDef hi2c1;

SPI_HandleTypeDef hspi3;

/* USER CODE BEGIN PV */
char CardID[4];
char MyCardID[4] = {199,246,213,115,151};
char lcd_buf[15];
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_SPI3_Init(void);
static void MX_CRC_Init(void);
static void MX_I2C1_Init(void);
/* USER CODE BEGIN PFP */
```

```
int main(void)
{

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


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


  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_SPI3_Init();
  MX_CRC_Init();
  MX_I2C1_Init();
  /* USER CODE BEGIN 2 */
  HAL_Delay(20);
  MFRC522_Init();
  ssd1306_Init();
  /* USER CODE END 2 */

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

        ssd1306_SetCursor(0,0);
        ssd1306_WriteString(lcd_buf, Font_7x10 , white);
        RFID_Id_Check();
        RFID_Id_Check();
        ssd1306_UpdateScreen();

      HAL_Delay(250);
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

```
void RFID_Id_Check()
{
if (MFRC522_Check(CardID) == MI_OK)
{
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_SET);
    HAL_Delay(25);
    if(MFRC522_Compare(CardID, MyCardID) == MI_OK)
    {
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, GPIO_PIN_SET);
        sprintf()(lcd_buf,"Gecerli Giris");
        ssd1306_SetCursor(0,1);
        sprintf()(lcd_buf,"Card ID: %c", CardID);
        HAL_Delay(250);
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, GPIO_PIN_RESET);
    }
    else
    {
        sprintf()(lcd_buf,"Gecersiz Giris");
        ssd1306_SetCursor(0,1);
        sprintf()(lcd_buf,"Card ID %c", CardID);
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, GPIO_PIN_RESET);
    }
}
else
{
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_RESET);
    HAL_Delay(25);
}

}
```

1. **Kütüphane ve Modül İnclude'ları:**
    - **rc522.h**: RFID okuma modülü ile iletişim kurmak için kullanılan kütüphanedir.
    - **i2c-lcd.h**: I2C ile haberleşen bir LCD ekran modülü ile iletişim kurmak için kullanılan kütüphanedir.
    - **ssd1306.h**: OLED ekran modülü ile iletişim kurmak için kullanılan kütüphanedir.
2. **Global Değişkenler:**
    - **CardID**: RFID karttan alınan kimlik bilgisini tutan bir karakter dizisidir.
    - **MyCardID**: Sistemde tanımlanmış olan geçerli kartın kimlik bilgisini tutan bir karakter dizisidir.
    - **lcd_buf**: LCD ekranına yazılacak metinleri tutan bir karakter dizisidir.
3. **Fonksiyonlar:**
    - **main**: Ana program döngüsünü başlatan fonksiyondur. RFID kart kontrol fonksiyonu çağrılarak, LCD ekranı güncellenir ve sürekli olarak kontrol yapılır.
    - **SystemClock_Config**: Sistem saatini yapılandırmak için kullanılan fonksiyondur.

- **MX_GPIO_Init**, **MX_SPI3_Init**, **MX_CRC_Init**, **MX_I2C1_Init**: Çeşitli donanım birimlerini başlatmak için kullanılan fonksiyonlardır.
- **RFID_Id_Check**: RFID kart kontrolü yapan ve LCD ekranına bilgi yazan ana kontrol fonksiyonudur.

4. **Ana Kontrol Döngüsü:**
   - **RFID_Id_Check** fonksiyonu çağrılarak RFID kart kontrolü yapılır. Bu fonksiyon şu adımları içerir:

     RFID_Id_Check fonksiyonu başlar.

     MFRC522_Check(CardID) == MI_OK ifadesi ile RFID kartın varlığı kontrol edilir.

     Eğer RFID kart tespit edilirse, HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_SET) ile LED1 (GPIOD_PIN_14) yanar.

     Ardından kısa bir bekleme süresi eklenir (HAL_Delay(25)).

     Daha sonra, MFRC522_Compare(CardID, MyCardID) == MI_OK ifadesi ile RFID kartın tanımlı geçerli kart ile eşleşip eşleşmediği kontrol edilir.

     Eğer eşleşiyorsa, HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, GPIO_PIN_SET) ile LED2 (GPIOD_PIN_15) yanar.

     LCD ekranına "Geçerli Giriş" ve kartın ID'si yazdırılır.

     Kısa bir bekleme süresi eklenir (HAL_Delay(250)).

     LED2 (GPIOD_PIN_15) sıfırlanır (HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, GPIO_PIN_RESET)).

     Eğer RFID kart eşleşmiyorsa, "Geçersiz Giriş" ve kart ID'si LCD ekranına yazdırılır.

     LED2 (GPIOD_PIN_15) sıfırlanır (HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, GPIO_PIN_RESET)).

     Eğer RFID kart tespit edilmediyse, LED1 (GPIOD_PIN_14) sıfırlanır (HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_RESET)).

     Fonksiyon sona erer.

5. **Hata İşleme:**
   - **Error_Handler**: Hata durumunda programın durdurulduğu bir hata işleme fonksiyonudur.