

FC41D Modülü Kurulumu

FC41D'nin stm32 içinde çalışması için gerekli kurulumların yapılması.

İçerikler

- [Cube IDE ve Priority Ayarı](#)
- [DMA Kullanımı ve Çalışması](#)
- [BLE Peripheral Mod Kod Kurulumu ve Akışı](#)
- [WiFi via BLE](#)
- [MQTT Ayarı](#)
- [Kesme İçinde İşleme](#)
- [Hotspot Ayarları](#)
- [QFLASH ile Yazılım Güncelleme](#)

Cube IDE ve Priority Ayarı

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	<input checked="" type="checkbox"/>	0	0
Hard fault interrupt	<input checked="" type="checkbox"/>	0	0
Memory management fault	<input checked="" type="checkbox"/>	0	0
Pre-fetch fault, memory access fault	<input checked="" type="checkbox"/>	0	0
Undefined instruction or illegal state	<input checked="" type="checkbox"/>	0	0
System service call via SWI instruction	<input checked="" type="checkbox"/>	0	0
Debug monitor	<input checked="" type="checkbox"/>	0	0
Pendable request for system service	<input checked="" type="checkbox"/>	0	0
Time base: System tick timer	<input checked="" type="checkbox"/>	0	0
PVD interrupt through EXTI line 16	<input type="checkbox"/>	0	0
Flash global interrupt	<input type="checkbox"/>	0	0
RCC global interrupt	<input type="checkbox"/>	0	0
USART1 global interrupt	<input checked="" type="checkbox"/>	1	0
DMA2 stream2 global interrupt	<input checked="" type="checkbox"/>	1	0
DMA2 stream7 global interrupt	<input checked="" type="checkbox"/>	1	0
FPU global interrupt	<input type="checkbox"/>	0	0

Dma ve usartın önceliğiyle systick aynı kalmamalı çünkü herhangi bir delay fonksiyonunda kesme ve delay çıkarsa sınırsız döngüye neden oluyor yukarıdaki resimdeki gibi çözülebilir.FC41D modulun at komutunu işleme hızı hakkında bilgiye sahip değiliz sadece OK yada Error dönme süresini biliyoruz.Tahmini her komut gönderilip kesmeye girildikten sonra verileri dma nın tam olarak yazması için 2000ms bekleme ile sorun çözülebilir .

DMA Kullanımı ve Çalışması

**

Aşağıda DMA ile USART'ı yapılandıran fonksiyon yer almaktadır:

Bu fonksiyon, **USART'**ı Idle moddayken interrupt'a girmesini sağlar. Bu durumda, belli bir süre veri alımı olmadığında USART işlemi durdurulur ve DMA kullanarak veriler bellekteki **pRxBuffPtr** tamponuna (*buffer*) aktarılır ve kesmeye girer.

```
void SetupDMAUSART() {
    HAL_UARTEx_ReceiveToIdle_DMA(&huart1, RxBuffer, RxBufferSize) ;
    __HAL_DMA_DISABLE_IT(&hdma_usart1_rx, DMA_IT_HT);
}
```

bu kodlardan üstteki kod Usart idle moddayken (Belli bir süre veri alımı olmadığında usartın işlem alımını bitirmesi) interrupta girmesini sağlar ve DMA la memorye pRxBuffPtr bufferine veriyi atar,bu interrupt girmesi dma çalışma grafiğindeki TC(transfer Complete durumuna bağlıdır).

Aşağıdaki ayarda DMA nın Usartın üzerinde olması gerekiyor.

HAL_Disable_IT_HT kodu ise Dma memorye yazma kısmının yarısında ise kesmeye girmesini kapatmaya yarıyor burada öyle bir amacımız olmadığı için kapattım.

Aşağıdaki ayarda DMA nın Usartın üzerinde olması gerekiyor.

Generated Function Calls

Generate Code	Rank	Function Name	Peripheral Instance Nam..	<input type="checkbox"/> Do Not Generate Function Call	<input type="checkbox"/> Visibility (Static)
<input checked="" type="checkbox"/>	1	SystemClock_Config	RCC	<input type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	2	MX_GPIO_Init	GPIO	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	3	MX_DMA_Init	DMA	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	4	MX_USART1_UART_Init	USART1	<input type="checkbox"/>	<input checked="" type="checkbox"/>

BLE Peripheral Mod Kod Kurulumu ve Akışı

```
void BleSetupFunction() {
    char *BleSettings[] = {
        "AT+QRST\r\n", // modul reseti
        "AT+QBLEINIT=2\r\n", // peripheral modda başlatır ble yi
        "AT+QBLEADDR?\r\n", // adres sorgulanır
        "AT+QBLENAME=BERKE\r\n", // isim verilir ble cihazına.
        "AT+QBLEGATTSSRV=FFF1\r\n", // Servis açılır bu servisin altında aşağıda
        // karakteristlikler açılıyor. Bu servis ve karakteristlikler hexadecimal olarak
        // ayarlanabilir.
        "AT+QBLEGATTSCCHAR=FFF2\r\n", // transfer karakteristliği veri bu
        // karakteristik üzerinden gönderilir
        "AT+QBLEGATTSCCHAR=FFF3\r\n", // burda 2. karakteristliği acabiliriz
        // yukardakinden farkı yok
        "AT+QBLEADVPARAM=150,150\r\n", // bu maximum reklam yayınlama degeridir bu
        // deger cihazın master tarafından bulunmasını kolaylaştırır.
        "AT+QBLEADVDATA=0201040BFF42454E204245524B4500\r\n", // hexadecimal olarak
        // yazılan bu kod soldan sağ a doğru (02 veri uzunlugunu), (01=veri türünü),
        // (04=reklam ayarı), (0BBF=MANUFACTURER DATA (Bu data ble protokolunde belirli
        // başlıklara işaret etmektedir isteğe göre degistirilebilir ble tarafından
        // tanımlanmıştır.)) kalan kısımda veri ("Ben Berke")
        "AT+QBLEADVSTART\r\n", // bütün ayarlar yapıldıktan sonra bu artık
        // bluetoothu başlatır.
    };

    for (int i = 0; i < 10; i++) {
        memset(TxBuffer, 0, sizeof(TxBuffer));
        strcpy(TxBuffer, Settings[i]);

        HAL_UART_Transmit_DMA(&huart1, (uint8_t*)TxBuffer, strlen(TxBuffer));

        HAL_Delay(500);
    }

    HAL_Delay(500);
}
```

Yukarıdaki kod çalıştırılırken Delay in 500 ms olmasına dikkat edilip kesmedeki delayinde 2000 saniye olması gerekmektedir.

```
"AT+QBLEGATTSNTFY=FFF2,merhaba\r\n"
```

//istenilen karakteristiklerde bu şekilde peripheral modda gönderim yapılabilir

```
void BLEHIGH() {
```

```
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7, GPIO_PIN_SET); //Modul uyku modundaysa bu  
    pin set edildiğinde uyanır (wakeUP)
```

```
}
```

WiFi via BLE

```
void TransmitBleFunction() {  
  
    char *Settings[] = {  
        "AT+QCRST\r\n",  
        "AT+QBLEINIT=3\r\n", //Ble yi wifi modunda baslatiyor BleSetupFunction()  
        fonksiyonunda yaptığımız ayarları burda otomatik kuruyor  
        "AT+QBLENAME=Rise_X_Gate_Way\r\n"  
    };  
  
    for ( *p=0; *p < 3; (*p)++) {  
        memset(TxBuffer, 0, sizeof(*p));  
        strcpy(TxBuffer, Settings[*p]);  
  
        HAL_UART_Transmit_DMA(&huart1, (uint8_t*)TxBuffer, strlen(TxBuffer));  
  
        HAL_Delay(700);  
    }  
}
```

*Settings[] dizisini pointerla tanımlamak güvenlik açısından avantajlı diziyi Rom a yazıyor ve degistirilmesine izin vermiyor.

MQTT Ayarı

```
void TransmitWifitoMQTTfunction()
{
    const char* SettingsWifi[] = {
        "AT+QMTCFG=\"version\",1,4\r\n", // bağlantı kimliği = 1 , versiyon =4 ile belirtilir
        "AT+QMTOPEN=1,\"104.248.21.33\",1883\r\n", // bağlantı kimliği = 1 , IP , PORT
        "AT+QMTCONN=1,\"Client8320\"\r\n", // bağlantı kimliği , Client8320 bu sunucu tarafından bağlantının tanımlanmasını sağlar
        "AT+QMTSUB=1,1,\"risex/1\",1\r\n", // bağlantı kimliği, mesaj kimliği, Mqtt topic, qos değeri 1 yapınca mesajın en az bir kere alınması garanti edilir ,
        "AT+QMPUB=1,1,2,0,\"risex/1\",5,\"HELLO\"\r\n" // bağlantı k , mesaj k , Qos, retain flag(0 olduğunda sunucuda saklanmaz 1 olduğunda saklanır, topic, mesajın bayt değeri, Mesaj)
    };

    for ( *q=0; *q < 5; (*q)++) {
        memset(TxBuffer, 0, sizeof(*q));
        strcpy(TxBuffer, SettingsWifi[*q]);

        HAL_UART_Transmit_DMA(&huart1, (uint8_t*)TxBuffer, strlen(TxBuffer));

        HAL_Delay(600);
    }
}
```

Qos 0 ken mesaj kimliği kullanılmaz qos 1 yapıldığında kullanılır çünkü mesajı takip etmek için mesaj kimliğine ihtiyacı vardır. Bağlantı kimliği hepsinde aynı olmalıdır.

Kesme İçinde İşleme

Idle kesmesine giren veriler, Feedback için işlenir aşağıdaki kodda işlemler yer alıyor.

```
void HAL_UARTEx_RxEventCallback(UART_HandleTypeDef *huart, uint16_t Size) {
    HAL_Delay(2000);
    if (huart->Instance == USART1) {
        memcpy(RxBuffer, huart->pRxBuffPtr, Size);

        if (*p < 3) {
            if ((strncmp((const char*) RxBuffer, "OK", 2) == 0) || //BLE için
```

KONTROL

```

        (strcmp((const char*) RxBuffer, "\r\nready\r\n", 5) == 0)) {

        memset(RxBuffer, 0, sizeof(RxBuffer));

        if (*p == 2) {
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_SET);
        }

    } else {
        memset(RxBuffer, 0, sizeof(RxBuffer));
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_3, GPIO_PIN_SET);
        Error_Handler();
    }
}

else if (strcmp((const char*) RxBuffer, "AT+QSTAAPINFO", 12) == 0) {
//BLE ÜZERİNDEN WIFI BAGLANTISI
    memcpy(TxBuffer, RxBuffer, sizeof(RxBuffer));
    memset(RxBuffer, 0, sizeof(RxBuffer));
    HAL_UART_Transmit_DMA(&huart1, (uint8_t*) TxBuffer, strlen(TxBuffer));
    // HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3, GPIO_PIN_SET); // yapildi
}

else if (strcmp((const char*) RxBuffer, "ERROR", 5) == 0) {
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_SET); // SADECE KIRMIZI
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_3, GPIO_PIN_RESET);
    memset(RxBuffer, 0, sizeof(RxBuffer));
}

else if (strcmp((const char*) RxBuffer, "+QSTATAT:WLAN_CONNECTED", 24)
== 0) { //WIFI BAGLANTISI SAGLANDI
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_3, GPIO_PIN_SET);
    memset(RxBuffer, 0, sizeof(RxBuffer));
}

else if (strcmp((const char*) RxBuffer, "+QSTATAT:AP_DISCONNECT", 23) ==
0) { //AP DISCONNECT OLDU
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_3, GPIO_PIN_RESET);
    memset(RxBuffer, 0, sizeof(RxBuffer));
}

// +QIURC: "recv",10
else if (strcmp((const char*) RxBuffer, "\r\n+QIURC: \"recv\"", 16) == 0)
{ // VERİ GELDİ
    char atCommand[100];
    char *ptr;
    int length;

    ptr = strstr((char*) RxBuffer, "+QIURC: \"recv\",");
    sscanf(ptr, "+QIURC: \"recv\",%d", &length);
    snprintf(atCommand, sizeof(atCommand), "AT+QIRD=%d,256\r\n", length);

```

```

        HAL_UART_Transmit_DMA(&huart1, (uint8_t*) atCommand,
strlen(atCommand));
        memset(RxBuffer, 0, sizeof(RxBuffer));
    }

    else if (strncmp((const char*) RxBuffer, "+QIRD:", 6) == 0) {
//VERİYİ OKUDU
        char atCommand[100] = "AT+QSTAAPINFO=";
        char *ptr;
        int length;

        ptr = strstr((char*) RxBuffer, "D: ");
        sscanf(ptr, "D: %d\r\n", &length);
        ptr = strstr(ptr, "\r\n") + 2;

        char Wifi[length + 1];
        strncpy(Wifi, ptr, length);
        Wifi[length] = '\0';
        strcat(atCommand, Wifi);

        HAL_UART_Transmit_DMA(&huart1, (uint8_t*) atCommand,
strlen(atCommand));
    }

    else {
        memset(RxBuffer, 0, sizeof(RxBuffer));
    }
}

HAL_UARTEx_ReceiveToIdle_DMA(&huart1, RxBuffer, RxBufferSize);
__HAL_DMA_DISABLE_IT(&hdma_usart1_rx, DMA_IT_HT);
}

```

Kesme içinde işlem yaparken, yukarıdaki gibi verileri bir **buffere** atarak geri bildirim (feedback) alabilir veya kontrol mekanizması yazabilirsiniz. Ancak, **kesme içinde delay kullanmak genellikle doğru bir tercih değildir**. Bu kodda, sadece tek kanaldan veri alıp kesmeye girdiğimiz için bu durum bir sorun yaratmıyor. Bu **delay**, modülün ayarlanması için gereken ortalama süreyi sağlıyor.

- Eğer çok kanallı işlem yapacaksanız, **RTOS** kullanarak kesmeleri yönetmek daha uygun bir çözüm olacaktır.
- Bu modülde optimizasyon için **QUEUE** kullanmak mantıklı olur çünkü verilerin hepsi bir buffere girmiyor ya da işlem sırasında veri kaybı yaşanabiliyor. (2 saniyelik **delay** tek kanal için optimal bir çözümdür.)

Hotspot Ayarları

Bu fonksiyon, cihazın Wi-Fi erişim noktası (hotspot) ve TCP bağlantılarını yönetmek için AT komutlarını kullanarak gerekli ayarları yapar. Aşağıda adım adım açıklaması verilmiştir:

```

void TransmitapFunction() {

```



```

// AT komutları dizisi tanımlanıyor. Bu komutlar, cihazın Wi-Fi erişim noktası
// ve TCP bağlantısı gibi ayarlarını yapılandırmak için kullanılır.
char *Settings[] = {
    "AT+QSOFTAP=\"Rise_X_Gate_AzkaBBy\"\r\n",           // Wi-Fi erişim
noktası adını belirlemek için kullanılır.
    "AT+QIOPEN=1,\"TCP LISTENER\",\"\",\"\",20,0\r\n",   // TCP Listener
açma komutu. 1. kanalda dinlemeyi başlatır.
    "AT+QIOPEN=2,\"TCP\", \"192.168.19.1\",20.10.0\r\n", // 2. kanalda
192.168.19.1 IP'si üzerinden TCP bağlantısı açma.
    "AT+QIRD=10,256\r\n"                             // TCP
bağlantısında veri okuma komutu. Maksimum 256 bayt veri okur.
};

// Döngü ile 3 komutu sırasıyla işleme alıyoruz.
for (*p = 0; *p < 3; (*p)++) {

    // Her döngüde TxBuffer temizleniyor. Burada p'nin gösterdiği değere bağlı
olan bellek alanı temizleniyor.
    memset(TxBuffer, 0, sizeof(*p));

    // AT komutları TxBuffer'a kopyalanıyor. Böylece gönderilmek üzere UART
için hazır hale getiriliyor.
    strcpy(TxBuffer, Settings[*p]);

    // Hazırlanan AT komutu DMA (Direct Memory Access) üzerinden UART1 ile
gönderiliyor.
    HAL_UART_Transmit_DMA(&huart1, (uint8_t*) TxBuffer, strlen(TxBuffer));


    // Her komut arasında 700 milisaniyelik bir gecikme veriliyor. Bu gecikme,
modülün komutları
    // işleyebilmesi ve komutlar arasında bir tampon görevi görmesi için
kullanılır.
    HAL_Delay(700);
}
}

```

->Hotspot ayarları yaparken TCP LISTENER'ı başlatmak zorundasınız.

QFLASH ile Yazılım Güncelleme


Eğer AT komutlarını doğru gönderdiğinizden eminseniz ve hala çalışmıyorsa, son çare olarak modülü **flashlamanız** gerekebilir. Aşağıda adım adım nasıl yapılacağı açıklanmıştır:



移远官网: www.quectel.com.cn
全国热线: 400 960 7678
邮箱: info@quectel.com

Build a Smarter World

Config

 Load FW Files

COM Port:
Baudrate: 921600

Name	Location
<input checked="" type="checkbox"/> File_Path	C:\Users\berke\Desktop\FC41DAAR03A08
<input checked="" type="checkbox"/> FC41DAAR03A08.bin	C:\Users\berke\Desktop\FC41DAAR03A08\FC41DAAR03A08.bin

Do not remove USB or terminate the downloading process before it completes.

Connect Internet Line Start...

Start

1. **ADIM**: Firmware sürümü seçilir.
2. **ADIM**: COM portu seçilir.
3. **ADIM**: Baudrate 921600 olarak ayarlanır.
4. **ADIM**: Start'a basılır.
5. **ADIM**:Altta "Erasing Flash" ifadesini gördüğünüzde **CEN**, 2 saniyeliğine **GND**'ye çekilir.
6. **ADIM**:Firmware yüklendikten sonra modüle **reset** atılır ve versiyon sorgulanır.