NACKADEMIN

Inbyggda system: arkitektur och design

Student: Olivia Werner
Email: olivia.werner@yh.nackademin.se

INNEHÅLLSFÖRTECKNING

INLEDNING

Vi har jobbat med ett drivrutins projekt med hjälp av plattformen STM32F411X. Detta projekt skall kunna slå på och av LED-lampor, samt välja vilken färg som skall lysas upp. För att ge oss en mer djupgående förståelse hur dessa projekt kan utformas i arbetslivet, har vi under projektets gång bekantat oss med hårdvaran, projektets manual samt datablad, UART-protokoll och drivrutiner.

ARBETSBESKRIVNING

Vi börjar med att kolla igenom Reference manual STM32F411xC[1] och User manual STM32 Nucleo-64[2]. För att förstå hur deras referens och user manualer är uppbyggda, samt hur vi skall kunna använda dessa hjälpmedel under projektets gång.

Därefter kollade vi på en Youtube[3] video samt följde instruktionerna för hur IDE STM32 programmet skall användas. Detta var ett bra första projekt samt ett test på hur mjukvaran fungerar. Det var väldigt givande då det många gånger kan vara lite svårare att endast läsa sig till hur olika program/applikationer skall fungera. Vid dessa fall är en video mycket hjälpsam att handha.

Därefter började vi att gå igenom och bearbeta koden vi fått tillhandahålla, för att förstå hur den är uppbyggt och vilka delar som gör vad. Utmanande i detta projekt var att förstå hur allt är uppbyggt och hur koden skall jobba tillsammans med hårdvaran. Att till exempel förstå sig på dem olika hexadecimaler fungerar och vad som gör vad beroende på vilka siffror som hexadecimalen innehåller.

Länkt till koden på GitHub: https://github.com/97olivia/UART_LED_DRIVER

---

[1] Reference manual STM32F411xC
https://www.st.com/resource/en/reference_manual/dm00119316-stm32f411xc-e-advanced-arm-based-32-bit-mcus-stmicroelectronics.pdf

[2] User manual STM32 Nucleo-64
https://www.st.com/resource/en/user_manual/um1724-stm32-nucleo64-boards-mb1136-stmicroelectronics.pdf

[3] YouTube video
https://www.youtube.com/watch?v=dnfuNT1dPiM&ab_channel=RobertFeranec

FÖRKLARING TILL KODEN

Led.cpp

Denna fil är kopplad till `LED.H` biblioteket för att kunna köras korrekt. Översiktligt är denna kod är till för att definiera LED-lampornas färg samt status med hjälp av 3 olika switch satser. Med en `setState` funktion för att uppdatera LED-lampans status och en `getState` funktioner för att hämta vad LED-lampans status är. I switch satsen konfigureras LED-pins baserat på deras färg samt status. Varje case fungerar på samma sätt men har olika LED-lampor/LED-färg kopplade till sig. I varje case ställs pin läget för LEDen med hjälp av rätt bit till MODER registret. Därefter kontrolleras LEDens tillstånd, om LEDen är på skickas en signal till ODR registret annars rensas pinen och stänger av LED-lampan.

Led.h

I denna fil definieras olika klasser samt typer för de olika LEDen. `#ifndef LED_h` och `#define LED_H` används för att förhindra flera inkluderingar för samma heder fil och säkerställer att innehållet endast används en gång för att förhindra krock i systemet. Därefter definieras en GPIOB port, samt en klocka till för att hjälp för de olika portarna. Under det definieras de olika led färgerna samt vilken port de tillhör och vilket mode i bit dem de skall använda. `typedef eum` bestämmer om LEDen skall vara av eller på 0/1. I klassen led finns både en private och en public. Private innehåller LEDens färg samt status(på eller av). I public finner man constructor som initierar LED färgen samt dess tillstånd. Den deklarerar även `setState` och `getState` som ställer in och hämtar LEDens status. `#endif` avslutar alla conditions.

main.cpp

Det är main funktionen som startar programmet. Den är kopplad till led.h biblioteket, och har en while loop som i detta fall blir en infinity loop.

stm32f4xx.h

`stm32f4xx.h` är en samlings fil som ofta används under projekt med STM32 microcontroller. Den innehåller exempelvis definitioner, makros och funktionsprototyper för att underlätta utvecklingsprocessen med mikrokontroller.

uart.cpp

Funktionen `USART2_Init(void)` deklarerar och initierar från UART protokollet och dess innehåll. Med att aktivera åtkomsten till klockan för UASRT2 med att placera bit 17 i APB1ENR registret till 1. Därefter gör den liknande med GPIOA genom att placera bit 0 register till 1. Efter rensas bit 4-7 i MODER för att förbereda pin PA2 och PA3. Bits 5-7 läggs om till 1 detta för att aktivera en alternativ funktion. Koden rensar bitarna 8-15 i AFR[0] för att förbereda pins och ställer sedan in bits 8-11 och 12-15 till det binära värdet 0111. Under UART2 ställs bandhastigheten in till 9600 bps genom att skriva värdet 0x0683 till BRR-registret. Den ställer in sändning och mottagning till 8-bits data med 1 stop bit, genom att skriva värdet 0x000C till CR1. Som man kan se i denna kod på rad 27 är att baud rate satt till 9600bps med hjälp av hexadecimalen 0x0683. Bude (Bps) menas med bit per sekund som kan överföras mellan dem olika applikationer. I just denna port är max bps satt till 9600 vilket är en standard fart då många applikationer oftast inte behöver ett snabbare bps än detta. Slutligen rensas CR2 och CR3. Därefter sätter koden bit 13 i CR1 till 1, vilket möjliggör UART.

Funktionen `USART2_write(int ch)` är ansvarig för att överföra data till terminalen. Den väntar på att statusen skall vara tom och redo på att ta emot nästa byte. Med hjälp av en while loop kontrolleras detta. När det är kontrollerat tomt skriver det ut den nya och returnerar i terminalen.

Funktionen `USART2_read(void)` läser av och tar in information. Den läser av data från registret SR med hjälp av en while loop. Sedan skriver den över data till DR registret.

uart.h

Är en heder fil som inkluderar och deklarerar filerna som behövs för att kunna kommunicera med resterande bibliotek.