NACKADEMIN

STM32 UART LED

Timmie Hasser - Timmie.Hasser@yh.nackademin.se

Innehållsförteckning

Introduktion:

Denna rapport beskriver processen för att kontrollera en LED med hjälp av UART kommunikation på ett STM32-mikroprocessor kort. Syftet med rapporten är att beskriva hur de olika delarna av programmet fungerar och förklara koden mer ingående.

Följande avsnitt kommer att beskriva STM32-plattformen, led och UART

Målet med rapporten är att läsaren ska få en förståelse av vad en LED är och hur det funkar när man använder en LED med hjälp av UART tillsammans med ett STM32 kort.

LED

LED(light emitting diode) eller lysdiod är en ljuskälla baserad på halvledarmaterial.

En LED innehåller ingen glödtråd, gas, glasbulb eller några rörliga delar till skillnad från andra ljuskällor, utan den innehåller en speciell halvledare som avger ljus när ström leds genom dioden.

Beroende på vilka grundämnen som ingår i halvledaren så lyser den med olika färger, tex röd, grön, blå, gul eller ultraviolett och det är bara några av alla färger som en led kan vara.

Det finns många fördelar med LED men 2 utav dom är att dom är väldigt energisnåla och att livslängden är upp till 50 gånger längre än andra ljuskällor och att dom är så pass energisnåla gör dom perfekta att använda på olika typer av mikrokontrollers både batteridrivna och såna med sladd.

UART

UART (Universal Asynchronous Receiver/Transmitter) är ett av dom mest använda kommunikationsprotokoll som används för seriell kommunikation mellan två enheter.

UART tar emot parallel data från tex en cpu och konverterar parallel datan till seriell data för att sedan skicka det till en annan UART.

För att skicka data mellan 2 UART så behövs endast 2 sladdar, en sladd från Tx pinen på den överförande UART till Rx pinen på mottagande UART och samma sak fast omvänt.

UART skickar data “asynchronously”, vilket betyder att det inte är någon klocksignal som synkroniserar utdata, Utan istället så används start och stopp bitar på datapaketen som visar vart datan börjar och när den slutar.

När UART känner av en start bit så börjar den läsa den inkommande datan bit för bit i en speciell takt som kallas “Baudrate”, Baudrate är hastigheten som data överförs som mäts i bits per sekund(bps), båda UART måste använda samma Baudrate för att kunna fungera och det brukar oftast vara 9600 men kan vara ända upp till 115200.

STM32

STM32 är en serie mikrokontroller enheter som tillverkas av STMicroelectronics, STM32 är baserade på ARM Cortex-M-Processorer och är populära inom inbyggda system.

Microkontrollerna i STM32 familjen erbjuder en mängd olika funktioner och prestanda alternativ för att passa olika behov.
De finns i allt från enklare 8-bits enheter till mer avancerade 32-bits enheter.

En av de fördelarna med STM32 serien är deras breda utbud av tillgängliga verktyg, mjukvara och dokumentation, STMicroelectronics erbjuder även en kostnadsfri utvecklingsmiljö(IDE) som heter STM32CubeIDE som är väldigt användarvänlig för att skapa och felsöka programvara för STM32 enheter, Det finns även många open source ramverk och bibliotek tillgängliga för att underlätta utvecklingen av program för STM32.

STM32 används i många olika applikationer såsom industriell automation, medicinteknik, fordon, hemelektronik och Internet of Things projekt.

STM32s flexibilitet, prestanda och tillgänglighet gör den till en populär plattform för utvecklare som arbetar med inbyggda system.

Program(kod)

Vårat program består ut av 6 olika filer:

- led.cpp
- led.h
- main.cpp
- stm32f4xx.h
- UART.cpp
- UART.h

Vi kan börja med led.h, UART.h och stm32f4xx.h dom filerna är så kallade header filer och dom innehåller vanligtvis deklarationer av funktioner, konstanter, datatyper och externa variabler och gör dem tillgängliga för andra programfiler eller moduler att använda.

Headerfiler används för att skapa ett internt gränssnitt mellan olika komponenter inom ett program. Genom att definiera funktioner och datatyper i en headerfil kan de inkluderas och användas i olika delar av programmet, vilket möjliggör modulär och återanvändbar kod.

I UART.h så har vi börjat med att includera dom olika biblioteken som vi behöver den första är STM32f4xx.h vilket är STM32s egna headerfil som innehåller definitioner, konstanter och funktioner som är specifika för alla STM32f4 mikrokontrollers, i headerfilen så finns all registertilldelning, strukturdefinitioner, bitmanipulering, interrupthantering och systeminitiering.
Den här headerfilen måste man använda när man programmerar STM32f4 mikrokontrollers för den definierar hur enheten fungerar så om man inte inkluderar den så kommer det uppstå flera problem såsom Svårighet att använda periferienheter, inkompatibilitet med andra bibliotek och svårighet med konfiguration och initialisering med mera.

Vi har även inkluderat stdio.h vilket är ett C bibliotek som står för standard input/output, detta bibliotek används för att hantera in och

utmatning via konsolen eller filer, stdio.h är ett av dom vanligaste C bibliotek och används i de flesta program.
Vi kallar även på en av våra funktioner som heter USART2_Init USART2_Init är en funktion som vi har skrivit i UART.cpp filen och den använder vi för att konfigurera och initiera USART protokollet och det var det sista i UART.h.

Nu börjar vi med att kolla UART.cpp och då börjar den med USART2_init och i den funktionen så börjar vi med att aktivera klockan för uart2 genom att ändra bit 17 i APB1ENR till 1 sen ändrar vi bit 0 till 1 i AHB1ENR för att aktivera port a, Sen så skickar vi 2 kommandon till gpioa moder ett kommando för att förbereda pin PA2 och PA3 och det andra för att aktivera alternativ funktionalitet på båda pinsen.
Efter det så väljer vi vilken typ av alternativ funktion vi vill ha på dom 2 pinsen genom att skriva till AFR[0] så förbereder vi och ändrar pinsen till den önskade funktionen.
Nu så ska vi konfigurera UART och då börjar vi med att sätta baudrate till 9600, efter det så ställer vi in tx och rx till att arbeta i 8 bits läge med 1 stop bit och ingen paritet, tx och rx är dom pinesen man använder för att skicka och ta emot data.
Vi nollställer CR2 och CR3 vilket gör att vi ställer in standardkonfigurationen för båda dom och sist så ändrar vi bit 13 i CR1 till 1 vilket aktiverar UART.

Och då är USART2_Init klar efter det så skapar vi en funktion som heter USART2_write och det den här funktionen ska göra är att skicka data och för att göra det börjar vi med att kolla om överföringen är tom om den är tom så skickar vi våran char till dataregistret.

Vi skapar även en funktion som heter USART2_read som ska läsa inkommande data och för att göra det så använder vi en while loop som kontrollerar SR(status registret) och väntar på en specifik status som säger att det finns data att läsa och när det finns det så läser vi av dataregistret och returnerar det.

Nu kan vi kolla hur led.h är uppbyggd och den börjar med att importera stdint.h, UART.h och stm32f4xx.h , stdint.h är ett nytt bibliotek för oss

och den innehåller olika heltalsdatatyper alltså olika typer av integer det som är bra med det är att vi kan specificera exakt hur stor man vill att våra integers ska vara och det är väldigt viktigt när man jobbar med mikrokontrollers för man vill vara så effektiv som möjligt så om man har en större int än man behöver så slösar man bara plats.
Sen så definierar vi vilken port som led ska använda och då blir det GPIO B vi definierar även klocksignalen för porten och även dom olika pinsen som dom olika led-färgerna har. Efter det så måste vi definiera mode bits för alla färger oxå.
Vi typ definierar alla färger och även de 2 olika lägen som leds kan finns i ON/OFF.
Sist så skapar vi en klass och i den klassen så definierar vi de olika attribut som led består av.

Nu går vi över till Led.cpp och i led.cpp så importerar vi led.h för att få alla beteckningarna och funktionen.
Vi börjar med att använda en konstruktor för LED och det den gör är att när en led får en färg och status så kollar den igenom listan och om färgen är tex röd så går den till röd och sedan kollar status om status är satt på on så kommer den att sätta på den röda led om status är satt på off så kommer den att stänga av den röda led istället, sen finns det samma funktionen för alla andra färger.
Sen har vi en setState funktion vilket kommer läsa av statusen på led och kolla så att rätt färg används och när den har hittat rätt färg så ändrar den pins till output läge och sedan kollar om önskad status var On isåfall definierar vi pin output till aktiv annars så definierar vi pin output till inaktiv och det var de sista.

Nu när vi har skapat alla våra .h filer och .cpp filer så för att använda dom så måste vi skapa en main.cpp fil och i den så importerar vi bara led.h eftersom den importerar dom andra header filerna vi behöver.

Vi börjar våran main med att definiera 3 leds och efter det så kallar vi på USART2_Init för att konfigurera och initiera UART efter det så definierar vi led1 som röd och ON, vi definierar led2 som blå och ON, vi definierar led3 som gul och ON, och det som har hänt nu är att en röd,blå och gul led har börjat lysa men efter det så hämtar vi status från led1 genom

getState funktionen sen ställer vi in led1 status till OFF genom setStatus funktionen vilket gör att röd led slutar lysa och efter det så använder vi delete för att ta bort led3 och då var hela programmet slut.