NACKADEMIN

Rapport

Linda Kjellberg IoT22

Linda.kjellberg@yh.nackademin.se

Github repo: https://github.com/LindaKjellberg/STM32_UART_LED.git

NACKADEMIN

Innehållsförteckning

NACKADEMIN

Inledning

Förtydligande:

I den här rapporten används begreppen main och huvudfunktion omväxlande. Även UART-protokollet och USART2.

SYFTE

Syftet med denna rapport är att ge läsaren insikt i den plattforms-lösning som rapporten beskriver, den förklarar de funktioner som används i lösningen, hur dessa fungerar och vilket samspel som finns mellan de olika källkod-filerna.

Den här lösningen består av drivrutiner till STM32F411-platformen skapade med ett UART kommunikationsprotokoll, med syftet att kontrollera periferienheter bestående av LED lampor.

Tanken med projektet är att få LED-lampor av fyra valda möjliga färger (röd, grön, gul och blå), att blinka vid kommando. Denna funktionalitet styrs av de två funktionerna ”USART2_read” som ansvarar för att ta emot data, och ”USART2_write” som ansvarar för skickad data. Kommunikationen mellan plattformen och periferienheterna möjliggörs med hjälp av USART2, vilket är en half-duplex, Syncrom variant av UART- kommunikationsprotokollet som är snabbare än UART.

Syftet bakom detta projekt var att uppnå en fördjupad förståelse av vanligt förekommande kommunikationsprotokoll inom inbyggda system, hur dessa fungerar och hur vi kan tillämpa dem för kommunikation mellan enheter.

NACKADEMIN

Dagbok

Inför uppstarten av detta projekt har vi bekantat oss med några vanligt förekommande kommunikationsprotokoll som används vid utveckling och drift av inbyggda system, och hur man kan implementera dessa. Däribland UART (Universal Asynchronous Receiver/Transmitter) vilket är det kommunikationsprotokoll vi kommer använde oss av i det här projektet.

Inledningsvis fick vi bekanta oss med den hårdvara och den utvecklingsmiljö som var tänkt att vi skulle använda i projektet. Detta gjordes genom att titta på en YouTube-tutorial som läraren hade hänvisat till. I videon sattes det upp ett liknande projekt till det vi har utfört, och en genomgång av hur man använder STM32 CubeIDE – den programvaran (kodeditor) som var tänkt att vi skulle använda i det här projektet.

Nästa steg var att bli bekant med relevant dokumentation tillhörande hårdvaran. Detta innefattade datablad, referensguide samt användarmanual för STM32F4-plattformen för att få en bättre förståelse för hur plattformen är uppbyggd och hur man arbetar med den. Det blev en stor utmaning för mig att läsa igenom all dokumentation för att försöka sätta mig in i och förstå helheten av plattformen. Jag tyckte att det både var svårt att förstå hur man läser av dokumentationen och även veta vilka delar som vi skulle arbeta med i projektet. Trotts att vi innan uppstart grundligt gick igenom vilka delar som var viktiga att kunna läsa av så upplevde jag en hel del förvirring kring hur jag skulle gå till väga för att hitta den informationen jag sökte.

Därefter gick jag vidare med att bekanta mig med hur man utvecklar drivrutiner med hjälp av UART-protokollet, detta gjordes med en uppgift där vi fick ett stycke kod som vi skulle skriva kommentarer till och förklara koden med hjälp av den dokumentation vi tidigare fått. Även det var en stor utmaning för mig, jag fann det svår att hitta informationen om vissa komponenter i dokumentationen.

Fortsatt bekantade vi oss med att utveckla drivrutiner för periferi-enheterna, som kontrollerar LED lamporna på samma sätt som med UART-övningen – vi fick färdiga kodstycken som vi med hjälp av dokumentationen skulle kommentera för att själva få en förståelse för vad koden gör och hur man konstruerar drivrutiner till den här typen av projekt. Jag stötte på liknande motgångar som i UART-övningen, även om jag kände mig lite mer bekväm så var den här koden mer omfattande.

Tack vare uppgifterna fick jag en helhetsbild av vad inlämningen ska bestå av och kunde därefter starta upp projektet.

NACKADEMIN

Resultat

Github repo: https://github.com/LindaKjellberg/STM32_UART_LED.git

UART.cpp:

Filen innehåller de tre UART-funktioner som används i projektet, alltså det är här vi skapar upp och möjliggör kommunikationen mellan STM34-kortet och periferienheterna (LED-lamporna).

USART2_Init-funktionen innehåller de konfigurationer som krävs för att hämta information från plattformens källkod (headerfil), detta görs genom att med en pekare (->) referera till betäckningen för de olika register vi vill använda och sen sätta den biten till (=) det hexadecimala talet som motsvarar aktivering av pinsen.

Det här är inställningarna för att chippet ska kunna använda UART-protokollet.

USART2_write är funktionen som hanterar utskicket av data med en kontroll-loop som kontrollerar att det inte finns någon data som ligger kvar i bufferten innan data skickas. Då skickar den ut data en bit i taget, bit efter bit.

Och USART2_read är den funktion som ansvarar för inläsning av data, den tar emot och läser in en bit data i taget.

UART.h:

Headerfilen som sammanbinder kommunikationsprotokollet vi har skapat, filen inkluderar källkoden för STM34-plattformen samt Standard I/O fil för cpp, och kallar på USART_Init funktionen, samt en testfunktion som inte används.

stm32f4xx.h:

Detta är en headerfil som automatiskt lades till i projektet vid uppstart i CubeIDE. Den innehåller nödvändiga funktionalitet för STM32-platformen.

Led.cpp:

NACKADEMIN

Det är LED.cpp filen som all funktionalitet för LED-lamporna finns, genom att inkludera header filen pekar med konstruktorn (struct) till LED.h (header filen) samt sätta funktionerna som hämtar- och sätter statusen.

Vi använde en konstruktor som inehåller (tar argumenten) (LedColor_Type som betonas av _color, och LedState_Type som betäcknas av _state) och med hjälp av en switch sats används för att växla mellan de olika färgerna och tända/släcka LED-lamporna.

Led.h:

I LED.h har vi skapat en led när den har en färg och ett tillstånd (på/av)

För att göra underlätta kodens läsbarhet är makron definierade för porten som vi anslöt klocksignalen till, porten som ansvarar för kommunikationen med LED-funktionen och aktiverade de pinsen och för varje pin som är satt till samtliga LED färger och tillstånd. Sedan används enum för att skapa upp och initialisera LED färgerna och LED tillstånden. Detta sammankopplas sen i en klass som möjliggör att vi kan kalla på de olika LED-lamporna och sätta på/stänga av dem i huvudfunktionen.

Main.cpp

Slutligen satte vi ihop projektet i main.cpp-filen med huvudfunktionen main som kör projektet. I filen skapas inledningsvis tre objekt upp som instanser av LED lampor globalt, dessa kommer kunna utnyttja den funktionalitet vi har satt upp åt lamporna med hjälp av objektens konstruktorer. Det vill säga – de kommer kunna anta en av de för-konstruerade färgerna (röd, blå, grön eller gul), samt anta ett utav två tillstånd (på/av). Därefter initialiseras den första lampan led1 samt ställs in till färgen röd och tänds utanför huvudfunktionen.

Fortsatt startar vi upp huvudfunktionen som först initialiserar USART2 protokollet. Och inuti main initialiserar vi även led2 till färgen blå och tänder lampan.

Sen skapar vi ett nytt Led objekt med färgen gul som vi också tänder. Och den här lampan allokerar vi minne dynamiskt för, samt med en pekare tilldelas det nya objektet variabelnamnet led3. Senare tar vi bort objektet så det allokerade minnet frigörs.

Huvudfunktionen avslutas med en oändlig loop som upprepar funktionen så länge programmet körs.

NACKADEMIN

Som avslut i projektet var det tänkt att vi skulle testköra projektet i simuleringsprogrammet Proteus, men det gick inte då vi stötte på problem med kodhanteringen efter en ny uppdatering av mjukvarans demoversion som vi skulle använda.

NACKADEMIN

Referenser

Lektionsmaterial och muntlig information från Ludwig Simonsson

Dokumentation tillhörande STM32F411-platform:

- Datablad
- Referensmanual
- Användarmanual