NACKADEMIN

2023-05-02
IOT 22

Inbyggda system: arkitektur och design

Slutuppgift – Hantera en LED med UART-protokollet

Marcus Nilsson – Marcus.Nilsson2@yh.nackademin.se

Innehåll

Inledning

UART utgör ett av tre huvudsakliga protokoll för kommunikation inom elektronik och IoT och utgör huvuddelen i detta projekt. UART är långsammare än exempelvis andra protokoll så som USB och Wifi men desto enklare och driver mindre minne och hårdvara.

UART består huvudsakligen av en transmitter och en receiver. Alltså en sändare av data och en mottagare av data. Där emellan finns en baud rate generator som anger hastigheten på överföringen. Överföringen vid UART är serial vilket innebär att data överförs i en serie, bit för bit över en kommunikationslinje. Mikrokontrollern kan emellertid ta emot datasekvenser som är av parallel men konverteras då senare till serial innan det når transmittern. Serien kan sedan brytas ned till asynkron som innebär överföring bit för bit och synkron som innebär överföring över block av data. Vid UART gäller asynkron men i UART-protokollet i detta projekt angavs USART vilket innebär en funktion som möjliggör att UART köra asynkront och synkront.

Följande projekt syftade till att fördjupa sig inom inbyggda system med fokus på kommunikation via UART med målsättning att arbeta närmare hårdvaran och få en djupare förståelse för kommunikation mellan enheter. Utförande involverade att med kod utveckla en lösning för STM32-plattformen där vi kan hantera LED med UART-protokollet. Utformningen av kod skedde i programmeringsspråket C.

Som hjälp för att utföra detta fanns instruktioner och dokumentation och sedan följdes arbetet i samrörelse med lektioner där läraren gick igenom dom olika stegen i utvecklingen.

Dagbok

4 april och 10 april: Föreläsning med introduktion till kommunikation och generell genomgång kring dom olika protokollen. Sedan följde vidare föreläsning kring kommunikation med fokus på UART och drivrutiner. Fokus under dom båda inledande lektionerna var att ta mycket anteckningar och bekanta sig med teori och nya begrepp.

11 april, 12 april och 13 april: Eget arbete med uppgift kring att läsa in sig på dokumentation och försöka översätta en given UART-kod. Målsättningen var att försöka bekanta sig med dokumentationen samt bättre förstå koden som tillämpas i UART.

18 april: Uppstart av projektet. Syftet med dagen var att få grepp om helheten kring projektet och sätta upp dom nödvändiga delar som krävdes för projektet. Inledningsvis laddades dokumentation och program ner. Sedan sattes även ett Github repository upp med struktur enligt anvisningar. Filstrukturen innebar en huvudmapp med tre undermappar. Fokus för min del lades på att fräscha upp minnet gällande Git och Github så mestadels av tiden spenderas på diverse Youtube-videos och att läsa dokumentation från Github.

19 april, 20 april och 21 april: Under dessa dagar följde en kombination av att först försöka sig på att förstå kodfiler för LED och sedan föreläsningar där läraren gick igenom dessa. Parallellt med föreläsningar och lärarens genomgång skrevs kod för LED och UART. Här upplevdes en del

svårigheter. Det gick exempelvis relativt smidigt att hitta relevanta delar i dokumentationen men det var däremot svårt att tolka det som stod i dokumentationen.

25 april: Genomgående har jag stämt av och hjälpts åt mycket i grupp. Efter ett kortare möte med gruppen konstaterade vi att vi har haft det svårt med projektet och tog ett möte med läraren. I mötet pratades det om förtydliganden av projektet och vad som förväntas av en vilket gjorde det lättare att komma vidare.

26 april: Tittade om dom gamla föreläsningarna för att på nytt gå igenom all kod rad för rad och på ett tydligare sätt kommentera koden. Det kändes väldigt värdefullt att kunna gå igenom allting igen i form av en repetition. Då mycket är nytt och det är visuellt svårt att föreställa sig alla delar var det givande att kolla tillbaka på gamla lektioner och anteckningar.

27 april: Rapportskrivande och struktur av filer bland mapparna.

Resultat

För kodfiler med kommentarer och förklaringar, se Github-repo:
https://github.com/MarcusNilssonn/ElevUppgift_MarcusN.git

Arbetet resulterade i totalt fem kodfiler enligt nedan, vilka redovisas i sin helhet på länkat repository från Github.

UART.h
Denna fil utgör en header-fil för UART där relevanta bibliotek inkluderas och två funktioner definieras.

UART.c
Denna fil utgör UART-protokollet. Först inkluderades relevanta bibliotek. Sedan initierades USART-funktionen som i sin tur initierade USART-protokollet. USART användes för att specificera en funktion som kan köra både synkront och asynkront. Därefter följde fyra steg.

1. Enabla klocktillgång för UART2. Genom att titta i blockdiagrammet i dokumentationen så såg att USART2 hade APB1 som buss. I referensmanualen, under klockan, så förtydligades att det var RCC som skulle användas och klockan ska sättas till 1 för att vara på. UART2 aktiverades sedan genommatt sätta bit 17 till 1. Med hjälp av hexadecimaler.
2. Enabla klocktillgång för port A. Ungefär på samma tillvägagångssätt enligt punkt 1 aktiverades GPIO genom att sätta bit 0 till 1.
3. Enabla pins relaterade till porten genom att i referensmanualen kolla på mode register. Detta utfördes genom att först rensa bitarna 4-7 innan vi aktiverade dom. Rensning utfördes för att säkerställa utifall dom var i andra lägen.
4. Val av alternativ funktion för de valda pinsen. Fanns ”low” och ”high” som variation för hur bitarna ska definieras men i aktuellt fall sattes en nolla, AFR[0], för att visa att alternativ funktion skulle användas och börja från början.

Sedan gjordes en konfiguration av UART där vi kontrollerade och satte en baud rate på 9600 bps, gav kontrollregister 1 aktiverad sändare och mottagare att arbeta med 8-bitarsdata samt att register 2 och 3 är nollställda. Sedan kunde UART aktiveras igen.

Ovanstående har resulterat i att UART kan användas.

Sedan skulle write- och read-funktioner göras vilket inleddes med att skapa write-regler. Till denna deklarerades write-funktion som kontrollerade att statusen på dataöverföringsregistret (bit 7) var tom och kan ta emot nästa byte och då sätter överföringen av byten till registret. Därefter gjordes på liknande sätt en read-funktion som med hjälp av en while-loop istället kontrollerar ifall det finns mer data att hämta och då läser ut datan.

Därefter strukturerades överföringsströmmarna med hjälp av en struct.

Sedan gjordes två funktioner. Den första funktionen läste från konsolen och den andra skrev ut till konsolen och slutligen gjordes en funktion för att testa läs- och skrivfunktionerna.

LED.h
Denna fil utgör en header-fil till LED. Här inkluderades relevanta bibliotek samt headern från UART. Vidare sattes ett antal definitioner vilka i huvuddrag var:

- Vilken GPIO som ska vara ansvarig för LED-funktionen.
- Klocksignalen för porten.
- Pins och mode bits för respektive LED-färg.
- Två enums för färg och status.
- En struct bestående av variablerna för färg och status.
- Funktioner för LED-konstruktorn och kontroll av status.

LED.c
Denna fil utgör C-filen där funktionerna, inkluderade från LED-headerfilen skrivs.

I den första funktionen, konstruktorn för LED-lamporna, skapades först två konstanter för färg och status som pekar på adressen till headern. Sedan enablas klockan. Därefter skrevs konfiguration av LED-pinsen beroende på färg med hjälp av ett switch-statement. Reglerna utfördes så att om färgen exempelvis var röd och om statusen är satt till ”på” så sattes LEDen på, annars till av. Detta case gjorde för alla färger.

Nästa funktion, som ska kontrollera statusen, var uppbyggd med liknande konstruktion som den första funktionen och tillämpades på alla färger. Det vill säga, med hjälp av ett switch-statement kontrollerades att färgen exempelvis var röd och om statusen är så definierades pinsens output till aktiv, annars inaktiv.

Avslutningsvis i denna fil skapades en funktion för att kontrollera färgen på en LED och printa vilken färg LEDen är satt till.

Main.c

Här testkörs all kod genom att inkludera våra filer och sedan skapades två LED-lampor. Sedan sattes färgen och statusen på dessa.

Github

I Github skapades ett repository med struktur enligt anvisningarna. Det vill säga en huvudmapp innehållandes undermappar för:

- Hårdvarumapp – Dokument avseende hårdvaran som projektet konstruerats för.
- Mapp för källkod – All kod.
- Mapp med rapport – Denna rapport.

Git och Github användes i den utsträckning det var möjligt för att bilda sig en bekvämlighet med användningen. Den initiala planen var även att nyttja tillfället i akt och arbeta med Git och använda sig av SSH-nyckel då detta tycks vara en mer företagsmässiga användningen. På grund av tidsbrist så fullföljdes aldrig detta utan vanlig token nyttjades.

Källhänvisning

** Muntlig information och föreläsningsmaterial från Ludwig Simonsson*

** UART Communication Protocol - How it works? - Codrey Electronics*

** Rapidtables.com/convert/number/*

** RM0383 Reference Manual– STMicroelectronics*

** STM32F411xC STM32F411xE – STMicroelectronics*

** UM1724 User manual - STMicroelectronics*