2023

NACKADEMIN

Hantera en LCD med UART-protokollet

MJUKVARUUTVECKLARE, INBYGGDA SYSTEM OCH IOT
MAHADI HASAN
Mahadi.Hasan@yh.nackademin.se

Innehållsförteckning

Abstrakt

Denna rapport diskuterar processen för att styra Liquid Crystal Display (LCD) med hjälp av en STM32-mikrokontroller. STM32-mikrokontrollern är ett populärt val för design av inbyggda system på grund av dess låga strömförbrukning, höga processorkraft och flexibilitet. I denna rapport, vi presenterar de grundläggande principerna för LCD-drift, STM32-mikrokontrollerarkitekturen, och processen för gränssnitt mellan LCD och STM32-mikrokontrollern. Vi presenterar också källkod i källkodsmappen som visar hur man styr LCD med STM32-mikrokontrollern.

Introduktion

LCD-teknik (Liquid Crystal Display) används ofta i elektroniska enheter som utgångsdisplay. LCD-skärmar kan hanteras med olika protokoll. En LCD-skärm (Liquid Crystal Display) med UART-protokoll (Universal Asynchronous Receiver Transmitter) är en typ av LCD-skärm som kan hanteras med UART-kommunikationsprotokollet. Detta innebär att LCD-skärmen kan ta emot data och kommandon från en mikrokontroll eller annan enhet som använder UART-kommunikation.

För att använda en LCD med UART-protokoll måste mikrokontrollern upprätta en kommunikationslänk med LCD-skärmen genom att ansluta TX- och RX-stiften på mikrokontrollern till motsvarande stift på LCD-skärmen. UART-modulen för mikrokontrollern måste också initieras genom att ställa in baudhastighet, paritet och andra parametrar.

När kommunikationslänken har upprättats och UART-modulen har initierats kan mikrokontrollern skicka kommandon och data till LCD-skärmen med UART-protokollet. Kommandona kan användas för att styra skärmen, till exempel att slå på / av, ställa in markörens position och rensa skärmen. Data kan skickas till LCD-skärmen för att visa text, bilder och annat grafiskt innehåll.

Fördelen med att använda en LCD med UART-protokoll är att det förenklar processen att hantera skärmen. I stället för att använda komplexa hårdvarugränssnitt kan mikrokontrollern kommunicera med LCD-skärmen med ett enkelt UART-gränssnitt. Detta gör det lättare att styra skärmen och minskar de hårdvaruresurser som krävs.

En LCD med UART-protokoll är ett användbart verktyg för att visa utdata i elektroniska enheter som använder mikrokontroller. Det ger ett enkelt och effektivt sätt att hantera displayen, vilket kan spara tid och resurser vid design och implementering av elektroniska system.

Bakgrund

UART (Universal Asynchronous Receiver-Transmitter) är ett kommunikationsprotokoll som används för seriell kommunikation mellan två enheter. Det är ett enkelt och allmänt använt protokoll som möjliggör dataöverföring mellan enheter med endast två ledningar (RX och TX).

UART fungerar asynkront, vilket innebär att data överförs utan användning av en extern klocksignal. I stället måste de sändande och mottagande enheterna i förväg komma överens om kommunikationshastigheten, som vanligtvis mäts i baudhastighet.

I UART-kommunikation överförs data som en serie bitar, där varje byte av data inramas av en start bit, en eller flera databitar, en valfri paritetsbit (för felkontroll) och en stoppbit. Start biten är alltid låg (0) och stoppbiten är alltid hög (1), med databitarna och paritetsbiten (om de används) placerade däremellan.

En LCD-skärm är en enhet som vanligtvis används som utgångsdisplay i elektroniska enheter. LCD-skärmen kan visa text, bilder och annat grafiskt

innehåll. För att hantera en LCD med UART-protokollet måste vi upprätta en kommunikationslänk mellan mikrokontrollern och LCD-skärmen.

STM32 Arkitektur för mikrokontroller

STM32-mikrokontrollerarkitekturen är baserad på ARM Cortex-M-processorarkitekturen, som är en populär arkitektur för inbyggda system på grund av dess låga strömförbrukning, höga prestanda och användarvänlighet. STM32-mikrokontrollerfamiljen stöder olika ARM Cortex-M-processorkärnor, inklusive Cortex-M0, Cortex-M0+, Cortex-M3, Cortex-M4, Cortex-M7 och Cortex-M23 / M33.

STM32-mikrokontrollerna har vanligtvis följande nyckelkomponenter:

1. Kärna: Detta är centralenheten (CPU) som utför instruktioner och utför beräkningar. Cortex-M-kärnorna som används i STM32-mikrokontroller är 32-bitars RISC-processorer (Reduced Instruction Set Computer) som erbjuder hög prestanda och låg strömförbrukning.

2. Minne: STM32-mikrokontrollerna har vanligtvis minne på chipet, inklusive flashminne för lagring av programkoden och RAM (Random Access Memory) för lagring av data och stack.

3. Kringutrustning: STM32-mikrokontrollerna har ett brett utbud av kringutrustning, inklusive GPIO, timers, ADC, DAC, USART, SPI, I2Cs, USB, CAN, Ethernet och mer. Dessa kringutrustning ger funktionalitet för att samverka med den externa världen och kommunicera med andra enheter.

4. Klock- och strömhantering: STM32-mikrokontrollerna har ett sofistikerat klock- och strömhanteringssystem som möjliggör effektiv drift och strömförbrukning. Klocksystemet genererar olika klocksignaler som krävs för drift av mikrokontroller och kringutrustning. Strömhanteringssystemet gör att mikrokontrollern kan gå in i lågeffektlägen när den inte används, vilket minskar strömförbrukningen.

5. Felsökning och programmering: STM32-mikrokontrollerna stöder vanligtvis olika felsöknings- och programmeringsgränssnitt, inklusive JTAG, SWD (Serial Wire Debug) och BOOT (Bootloader) lägen.

Sammantaget ger STM32-mikrokontrollerarkitekturen en kraftfull och flexibel plattform för att utveckla inbyggda system med ett brett utbud av funktioner och funktioner. Arkitekturen stöder en rik uppsättning kringutrustning, låg strömförbrukning, effektiv drift och användarvänlighet, vilket gör den till ett idealiskt val för ett brett spektrum av applikationer.

Metodik:

För att hantera en LCD med UART-protokollet måste vi följa följande steg:

Först, jag skapade UART.c fil. Gör en funktion som kallas UART_Config genom att aktivera USART2- och GPIOA-klockorna med RCC APB1ENR- respektive AHB1ENR-registren. Genom att aktivera dessa klockor kan mikrokontrollern komma åt UART-modulen och GPIO-stiften. GPIO-stiften används för att konfigurera USART-stiften för alternativa funktioner.

Nästa steg är att konfigurera USART-stiften för alternativa funktioner. Detta steg innebär att ställa in lägesregistret (MODER) för att välja den alternativa funktionen för stiften PA2 och PA3, som används för sändnings- respektive mottagningssignalerna. Dessutom är utgångstypregistret (OTYPER) inställt på att konfigurera stiften som utgångar med öppen dränering, vilket gör att stiften kan

användas i kommunikation med flera droppar. Utgångshastighetsregistret (OSPEEDR) är inställt för att konfigurera stiften för höghastighetsutgång och pull-up/pull-down-registret (PUPDR) är inställt för att konfigurera stiften för pull-up. Slutligen är det alternativa funktionsregistret (AFR) inställt på att välja USART-alternativfunktionen för stiften PA2 och PA3.

Efter att ha konfigurerat USART-stiften för alternativa funktioner konfigureras USART-modulen genom att ställa in baudhastighetsregistret (BRR), kontrollregistret 1 (CR1), kontrollregistret 2 (CR2) och kontrollregistret 3 (CR3). BRR-registret är inställt på önskad överföringshastighet på 9600 bps. CR1 registret är inställt på att aktivera sändaren och mottagaren, och CR2 och CR3 registren avaktiveras. Slutligen aktiveras USART genom att ställa in UE-biten i CR1-registret.

Funktionen UART_Write används för att skriva data till UART-modulen. Denna funktion kontrollerar TDR-statusregistret (SR) för att säkerställa att det är tomt innan data skrivs till registret. Den här funktionen är användbar för felsökning och dataloggning.

UART_Read-funktionen används för att läsa data från UART-modulen. Denna funktion implementerar två steg: Om bara en byte behöver läsas skriver funktionen slavadressen och väntar på att adressbiten (bit 5 i SR) ska ställas in. Sedan väntar den på att RXNE-biten (mottagningsbuffert inte tom) ska ställa in och läser data från DR. Om flera byte behöver läsas skriver funktionen slavadressen och väntar på att adressbiten (bit 5 i SR1) ska ställas in. Sedan väntar den på att RXNE-biten (mottagningsbuffert inte tom) ska ställa in och läser data från DR.

Den UART_Address funktionen ansvarar för att skicka slavadressen via UART. Funktionen tar en Address-parameter av typen uint8_t som är slavadressen som ska skickas. Funktionen skickar slavadressen till dataregistret (DR) för USART2-kringutrustningen. Dataregistret innehåller de uppgifter som ska överföras/tas emot via UART. När adressen har skickats väntar funktionen på att ADDR-biten ska ställas in i statusregistret (SR) för USART2-kringutrustningen. ADDR-biten

anger slutet på adressöverföringen. När ADDR-biten har angetts läser funktionen statusregistret (SR) för att rensa ADDR-biten. Detta görs för att säkerställa att UART-kommunikationsprotokollet fungerar korrekt.

Sedan skapade jag en UART.h-fil som anropar UART_config, UART_Write, UART_address och UART_Read funktioner. Sedan inkluderade jag alla rubrikfiler som krävs som "stdint.h" för C: s standardbibliotek och "stm32f4xx.h" STM32-plattformens bibliotek.

Därefter skapade jag filen "UART-LCD.c". Koden i den här filen implementerar en uppsättning funktioner för att styra en LCD-skärm via ett UART-kommunikationsgränssnitt. Den använder en header-fil "UART.h" för att inkludera UART-funktionerna och "Delay.h" header-filen för tidsfördröjningar och "UART-LCD.h" header-fil för LCD funktioner. Koden implementerar flera funktioner för att initiera, rensa, ställa in markörposition, skicka data och skicka kommandon till LCD-skärmen.

Lcd_Write funktion ansvarar för att skriva data till LCD-skärmen. Det tar LCD-slavadressen, databytematrisen och storleken på data som ingångar. Den ställer in slavadressen med funktionen UART_Address (Adress) och skickar sedan databytearrayen till LCD-skärmen via UART med funktionen UART_Write (* Data ++). Lcd_send (0, cmd) funktion skickar ett kommando till LCD-skärmen. Ställer sedan in databyte för att aktivera och inaktivera TS- och RS-stiften. Den använder LCD_Write funktion för att skriva kommandobyte till LCD-skärmen. lcd_send(1, data) funktion ansvarar för att skicka data till LCD-skärmen. ställer sedan in databyte för att aktivera och inaktivera TS- och RS-stiften. Den använder LCD_Write funktion för att skriva databyte till LCD-skärmen. lcd_clear funktion rensar LCD-skärmen genom att skicka clear kommando till skärmen. lcd_put_cur funktion ställer markörpositionen på LCD-skärmen. Det tar rad- och kolumnnumren som indata och ställer sedan in DDRAM-adressen baserat på dessa nummer. lcd_init funktion initierar LCD-skärmen. Den implementerar 4-bitars initialiseringssekvensen genom att skicka kommandon till displayen för att konfigurera sina interna register. lcd_send_string funktion skickar en teckensträng till LCD-skärmen genom att anropa lcd_send (1, data) funktionen för varje tecken i strängen.

Sedan skapade jag en UART-LCD.h-fil som anropar lcd_init, Lcd_Send, lcd_send_string, lcd_put_cur och lcd_clear funktioner. Sedan inkluderade jag alla rubrikfiler som krävs, till exempel "stdint.h" för C: s standardbibliotek och "stm32f4xx.h" STM32-plattformens bibliotek.

Efter detta skapade jag "Delay.c" -fil som innehåller två header-filer, "Delay.h" och "UART.h". Header-filen "Delay.h" innehåller deklarationen av Delay_us- och Delay_ms-funktioner, och header-filen “UART.h" innehåller deklarationer för UART-funktioner.

Funktionen TIM5Config konfigurerar timer5 så att den genererar fördröjningar i mikrosekunder. I steg 1 aktiveras timer5-klockan genom att ställa in den 3: e biten i APB1ENR-registret. I steg 2 är praskalarvärdet satt till 90–1, vilket innebär att timern körs med 1 MHz-frekvens. ARR-registret är inställt på det maximala värdet 0xFFFF. I steg 3 aktiveras räknaren genom att ställa in den 0:e biten i CR1-registret och while-loopen väntar på att uppdateringsflaggan (UIF) ska ställas in. Den här flaggan anger att timerräknaren har nått det maximala värdet och återställts till noll. Denna process för att återställa timerräknaren är nödvändig för att ge den exakta fördröjningen.

Funktionen Delay_us genererar fördröjningar i mikrosekunder. I steg 1 nollställs timerräknaren. I steg 2 väntar while-loopen tills timerräknaren når önskat värde, vilket skickas som ett argument till funktionen. Eftersom varje räkning av timern tar en mikrosekund kommer den totala väntetiden att vara den nödvändiga mikrosekundfördröjningen.

Sedan skapade jag en Delay.h-fil som anropar TIM5Config, Delay_us och Delay_ms funktioner. Sedan inkluderade jag alla rubrikfiler som krävs som "stdint.h" för C: s standardbibliotek.

Sedan tog jag "stm32f4xx.h" -fil från Ludwigs github-repo som är en enhetsspecifik header-fil för STM32F4xx-mikrokontroller. Den innehåller definitioner för alla register, bitar och kringutrustning som finns tillgängliga i STM32F4xx-familjen av mikrokontroller, vilket gör det lättare att programmera mikrokontrollern utan att behöva memorera minnesadresser eller bitpositioner. Denna header fil ingår i alla STM32F4xx projekt för att tillåta åtkomst till

hårdvaran i mikrokontrollern. Den definierar olika strukturer och makron som representerar register och bitar på mikrokontrollern. Huvudfilen ger också tillgång till viktiga funktioner och bibliotek som är specifika för STM32F4xx-familjen av mikrokontroller. Genom att inkludera "stm32f4xx.h" i ett projekt kan programmeraren enkelt interagera med mikrokontrollerns hårdvara och utveckla kod för att styra kringutrustning som timers, ADC, UART och mer. Denna header fil är ett viktigt verktyg för STM32F4xx mikrokontroller programmering.

Slutsats:

Sammanfattningsvis är hantering av en LCD med UART-protokollet ett enkelt och effektivt sätt att visa utdata på en LCD-skärm. UART-protokollet används ofta i mikrokontroller, och LCD-skärmen används ofta som utgångsdisplay i elektroniska enheter. Genom att följa stegen som beskrivs i denna rapport kan vi framgångsrikt hantera en LCD-skärm med UART-protokollet.