Serijska komunikacija z UART v STM32F7

Na tokratni vaji bomo spoznali serijsko komunikacijo po protokolu UART. UART je kratica za Universal Asynchronous Receiver/Transmitter. Kratico UART uporabljamo tudi za ime naprave mikrokrmilnika, ki omogoča asinhrono serijsko komunikacijo. Obstajajo tudi naprave, ki poleg asinhrone ponujajo tudi sinhrono komunikacijo. Le-te označujemo s kratico USART. Na tokratni vaji se bomo posvetili predvsem asinhroni različici.

Mikrokrmilniške USART naprave običajno delujejo s signali z napetostmi od 0V do 3,3V ali 5V. Obstajajo pa tudi dodatni moduli, ki omogočajo da USART signale pošiljamo v skladu s standardom RS-232. Slednji je telekomunikacijski standard, ki določa, da je logična enica predstavljena z napetostjo od -25V do -3V, logična ničla pa z napetostjo od 3V do 25V. Klasični računalniški serijski porti, ki jih danes sicer poredko srečamo v osebnih računalnikih in prenosnikih, delujejo na napetostnih nivojih od -13V do 13V. Včasih so bili serijski porti standardni del vsakega računalnika, saj so služili za komunikacijo z dial-up modemi, čitalci za črtne kode, CNC stroji, igralnimi palicami (angl. joystick) ter množico ostalih naprav.

Danes se USART standard samostojno ali v navezavi s standardi RS-232 uporablja za komunikacijo med mikrokrmilniki, z GPS senzorji, moduli za Bluetooth, WiFi, Ethernet, Zigbee in podobno. Pogosto jih srečujemo tudi v industrijskih aplikacijah. Skorajda ni mikrokrmilniškega sistema, ki ne bi uporabljal UART protokola.

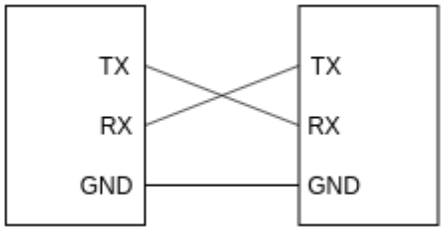
Povezava UART naprav

Najenostavnejši in hkrati najpogostejši priklop dveh UART naprav je prikazan na sliki 1. Na napravah uporabimo zgolj pina TX (Transmit) in RX (Receive) ter GND. Pin TX je namenjen pošiljanju, RX pa sprejemanju, zato jih povežemo križno. Povežemo tudi GND pin, da zagotovimo usklajenost logičnih nivojev na obeh straneh.

UART HFC (Hardware Flow Control)

Dodatno lahko na napravah uporabimo še pina CTS (Clear to Send) in RTS (Request to Send). Omenjena pina sta namenjena temu, da se UART napravi uskladita kdaj sta pripravljeni za sprejem podatkov. Če je naprava pripravljena na sprejem podatkov, pin RTS postavi na nizko logično vrednost. Na

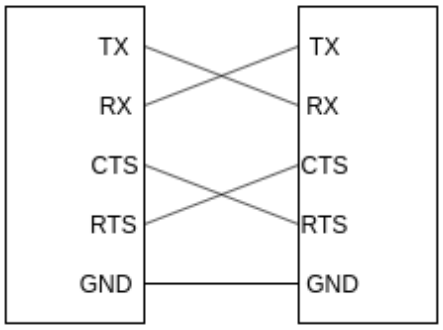
Slika 1: Enostavna vezava UART naprav.



pinu CTS pa naprava preveri, če je naprava na drugi strani pripravljena za sprejem. Ko oba pina na povezanih napravah povežemo križno, kot je prikazano na sliki 2, dobimo povezani napravi, ki pošiljata zgolj, ko je naprava na drugi strani pripravljena. Preverjanje in nastavljanje RTS in CTS pinov se v UART napravi izvaja strojno, brez programskega posredovanja.

HFC se uporablja predvsem pri visokih prenosnih hitrostih, ko prenašamo velike količine podatkov ali ko je naprava, ki sprejema podatke počasna.

Slika 2: Enostavna vezava UART naprav z uporabo Hardware Flow Control signalov.

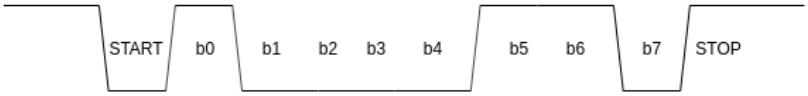


Asinhron prenos

V mirujočem stanju mora biti prenosna linija v visokem logičnem stanju (logična enica), kar se običajno zagotovi s pull-up uporom. Pojavitev ničle, ko je linija v mirujočem stanju, označuje začetek prenosa. Omenjena ničla

se zato imenuje start bit. Start bitu sledi 8 ali 9 podatkovnih bitov. Podatkovnim bitom nato lahko sledi paritetni bit temu pa stop bit. V primeru, da paritetnega bita ne potrebujemo stop bit sledi takoj za podatkovnimi biti. Primer prenosa števila 0x61 (0110 0001) brez paritetnega bita je prikazana na sliki 3.

Slika 3: Primer prenosa števila 0x61 (0110 0001).



Hitrost prenosa je pri UART napravah podana v baudrate-u, kar označuje število simbolov na sekundo. Tipične vrednosti za baudrate so 9600, 19200, 38400, 57600, 115200, 230400, 460800 in 921600. UART napravi, ki nastopata v prenosu, morata imeti usklajen baudrate. V nasprotnem primeru prenos ne bo uspešen. Prav tako morata napravi biti usklajeni v pariteti ter dolžini stop bit periode. Ta je lahko enaka času prenosa enega ali dveh bitov.

UART v STM32F769

Krmilnik STM32F469, ki ga uporabljamo na vajah, ima 8 UART naprav: USART1, USART2, USART3, UART4, UART5, USART6, UART7 in UART8. Naprave 1,2,3 in 6 so torej sposobne asinhronega in sinhronega prenosa, napravi 4,5,7 in 8 pa zgolj asinhronega.

Inicializacija UART naprave

Kot običajno moramo na začetku najprej prižgati uro U(S)ART naprave. To storimo z ukazom `__HAL_RCC_USARTx_CLK_ENABLE()`, kjer `USARTx` predstalvlja oznako določene USART naprave. Bodite pozorni na razlike v imenih naprav (USART ali UART).

Nato je potrebno napravi določiti vse nastavitve prenosa. Te določimo s strukturo `UART_HandleTypeDef`. Prvi element te strukture je, podobno kot pri inicializaciji SPI naprave, element `Instance` s katerim določite napravo, ki jo želite uporabljati.

Drugi element strukture je struktura `Init`, ki hrani vse nastavitve UART prenosa. Posamezni elementi te strukture so opisani v nadaljevanju. Ko določimo vse nastavitve prenosa inicializiramo napravo s klicem funkcije `HAL_UART_Init(UART_HandleTypeDef*)`.

Mode

Nastavitev `Mode` določa namen za katerega bomo uporabili napravo. UART napravo namreč lahko uporabimo samo za sprejemanje (`UART_MODE_RX`), samo za pošiljanje (`UART_MODE_TX`) ali za oboje hkrati (`UART_MODE_TX_RX`).

BaudRate

Nastavitev `BaudRate` določa hitrost prenosa. Kot že rečeno, se največkrat uporablja vrednosti 9600, 19200, 38400, 57600, 115200, 230400, 460800 in 921600.

WordLength

Z nastavitvijo `WordLength` določate koliko podatkovnih bitov se prenese v enemu UART prenosu. Možnosti sta 8 (`UART_WORDLENGTH_8B`) ali 9 bitov (`UART_WORDLENGTH_9B`).

Parity

`Parity` določa želeno pariteto, ki je lahko liha (`UART_PARITY_ODD`), soda (`UART_PARITY_EVEN`) ali pa je ne uporabljamo ((`UART_PARITY_NONE`).

StopBits

Lahko imamo en stop bit (`UART_STOPBITS_1`) ali dva (`UART_STOPBITS_2`).

HwFlowCtl

`HwFlowCtl` nastavlja HFC, imamo več opcij:

- `UART_HWCONTROL_NONE` – HFC je izklopljen.
- `UART_HWCONTROL_RTS` – uporabimo zgolj RTS.

- `UART_HWCONTROL_CTS` – uporabimo zgolj CTS.
- `UART_HWCONTROL_RTS_CTS` – HFC uporabimo v celoti.

Primer inicalizacije

Primer inicalizacije UART naprave je prikazan spodaj, nastavljamo napravo USART1.

```
UART_HandleTypeDef uart;
uart.Instance = USART1;
uart.Init.BaudRate = 115200;
uart.Init.WordLength = UART_WORDLENGTH_8B;
uart.Init.StopBits = UART_STOPBITS_2;
uart.Init.Parity = UART_PARITY_ODD;
uart.Init.Mode = UART_MODE_TX;
uart.Init.HwFlowCtl = UART_HWCONTROL_RTS_CTS;
HAL_UART_Init(&uart);
```

Inicializacija GPIO pinov

Poleg inicializacije UART naprave moramo inicializirati tudi pine, ki jih naprave uporabljajo. Nastavimo jih kot alternativne funkcije (ne izhod in ne vhod) v push-pull (PP) načinu brez pull-up ali pull-down uporov. Poleg tega moramo določiti tudi katero alternativno funkcijo želimo. Primer inicializacije je prikazan spodaj.

```
__HAL_RCC_GPIOA_CLK_ENABLE();
GPIO_InitTypeDef init_structure;
init_structure.Pin = GPIO_PIN_9 | GPIO_PIN_10;
init_structure.Pull = GPIO_NOPULL;
init_structure.Speed = GPIO_SPEED_FREQ_LOW;
init_structure.Mode = GPIO_MODE_AF_PP;
init_structure.Alternate = GPIO_AF7_USART1;
HAL_GPIO_Init(GPIOA, &init_structure);
```

Pošiljanje in sprejemanje

Funkcije za pošiljanje in sprejemanje so podobne tistim, ki smo jih uporabili pri SPI napravah. Za pošiljanje uporabimo funkcijo `HAL_UART_Transmit()`.

Prvi parameter je kazalec na inicializacijsko strukturo UART naprave. Sledi kazalec na spremenljivko za pošiljanje. Z zadnjima dvema parametroma določimo število bajtov v prenosu ter maksimalni čas prenosa (angl. timeout) v milisekundah. V primeru, da je prenos zaključen znotraj maksimalnega dovoljenega časa, funkcija vrne `HAL_OK`, v nasprotnem primeru pa `HAL_TIMEOUT`. Primer uporabe omenjene funkcije je prikazan spodaj. Funkcija za sprejemanje `HAL_UART_Receive` je podobna funkciji za pošiljanje, s to razliko da podamo kazalec na spremenljivko kamor naj se shranijo podatki. Obe funkciji sta blokirajoči, kar pomeni, da lahko program nadaljuje šele, ko je pošiljanje ali sprejemanje zaključeno. To pa je pogosto neželjeno obnašanje, zato bomo v nadaljevanju spoznali tudi neblokirajoče klice, ki se izvajajo s pomočjo prekinitev.

```
1 uint8_t buffer[4];
2 // ali uint8_t buffer[] = "test";
3 HAL_UART_Receive(&uart, buffer, sizeof(buffer), HAL_MAX_DELAY);
4 HAL_UART_Transmit(&uart, buffer, sizeof(buffer), HAL_MAX_DELAY);
```

Priprava projekta

Za to, da lahko uporabimo UART funkcije moramo v projekt v STM32Cube dodati datoteke iz STM32 HAL knjižnice. Vse datoteke knjižnice se nahajajo na:

- Windows:

  `C:/Users/<username>/STM32Cube/Repository/STM32Cube_FW_F7_V1.16.0/Drivers/STM32`

- Unix:

  `/home/username/STM32Cube/`

Za uporabo UART funkcij kopirajte `Src/stm32f7xx_hal_uart.c` iz knjižnice v projekt na lokacijo `Drivers/STM32F7xx_HAL_Driver/Src`.

Nato skopirajte še datoteko `Inc/stm32f7xx_hal_uart.h` iz knjižnice v projekt na lokacijo `Drivers/STM32F7xx_HAL_Driver/Inc`.

Ko dodate datoteki, v `Inc/stm32f7xx_hal_conf.h` odkomentirajte vrstico, ki definira `HAL_UART_MODULE_ENABLED`.

Naloga

Inicializirajte USART1, da bo deloval z Virtual COM Port ter bo sprejemal ukaze v obliki `LED X Y`, kjer je X oznaka LED (od 0 do 2), Y pa kakšno naj bo stanje LED diode (0 = ugasni, 1 = prižgi).