ŽILINSKÁ UNIVERZITA V ŽILINE

FAKULTA RIADENIA A INFORMATIKY

Semestrálna práca

Softvérový UART

Bc. Ján Remeň

5ZIF11

Obsah

1 Zadanie

Zadaním práce je vytvoriť a otestovať knižnicu na softvérovú simuláciu UART, za použitia Mbed štúdia a vývojovej dosky STM32L476.

2 Hw UART

2.1 Popis

UART (universal asynchronous receiver-transmitter), čiže univerzálny asynchrónny prijímač/vysielač, je sériové rozhranie pre prenos dát medzi zariadeniami v oboch smeroch. Používa sa pre komunikáciu medzi MCU, počítačmi a ďalšími zariadeniami.

Pre komunikáciu potrebujú obidve komunikujúce zariadenia vlastný UART. Zariadenie ktoré vysiela dáta, konvertuje paralelné dáta zo zariadenia, napr. CPU, do sériového tvaru, odkiaľ sú vyslané na prijímajúce zariadenie a následne spracované.

Pre komunikáciu sú potrebné len dva vodiče. Jeden, vysielajúci vodič, ktorý sa spojí s prijímajúcim vodičom na druhom zariadení a prijímajúci vodič ktorý sa spojí s vysielacím vodičom na druhom zariadení.

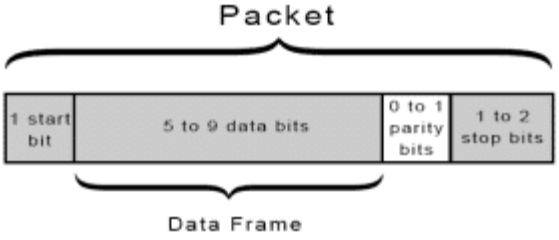
Dáta sú prenášané asynchrónne, čo znamená že pre prenos netreba hodinový signál. Miesto neho sa používa start a stop bit, Tieto bity definujú, kedy má zariadenie očakávať dáta.

Keď zariadenie, ktoré prijíma dáta, detekuje start bit, začne čítať nasledujúce bity určitou frekvenciou, tzv. baudrate. Baudrate je rýchlosť, ktorou sa prenášajú bity za sekundu. Obidve zariadenia musia mať nastavenú rovnakú baudrate. Taktiež musia mať obidve zariadenia nastavené rovnaké ďalšie vlastnosti, ako je dĺžka bitov, dĺžka stop bitu, parita.

2.2 Fungovanie

UART získa dáta zo zbernice, ktoré tam môže poslať pamäť, CPU a podobne. Po prijatí, k ním pridá start bit, parity bit a stop bit, čím vytvorí dátový paket. Následne sa dáta prenesú bit po bit na vysielací vodič. Prijímajúce zariadenie zistí zmenu na vodiči a bit po bite príjme dáta. Tie je následne možné ďalej spracovať.

Tvar paketu je možné vidieť na obrázku.



2.2.1 Popis bitov

Start bit

Ak sa neprenášajú dáta, tak je na vysielacom vodiči logická 1. Pre začiatok vysielania dát sa zmení na logickú 0, ktorú tam drží jeden časový úsek. Ak sa pri prijímacom zariadení zistí zmena z logickej 1 na 0, tak začne čítať bity na základe nastavenej frekvencie.

Dátový rámec

Obsahuje prenášané dáta. Môže byť 5 až 8 bitov dlhý ak sa používa parita. Ak sa nepoužíva, tak môže byť dlhý až 9 bitov. Dáta sa väčšinou posielajú s najmenej významným bitom ako prvým.

Parita

Parita popisuje párnosť alebo nepárnosť dát. Slúži na detekciu zmeny dát pri prenose. Pri prijatí, UART zráta počet bitov s hodnotou jedna a skontroluje či je výsledok párny alebo nepárny. Následne, tento výsledok porovná s parity bitom. Ak je nastavený na 1 (nepárne), tak aj narátaný počet musí byť nepárny. Ak je to tak, prenos nastal bez zmeny. Ak sa to ale nerovná, vieme že dáta boli zmenené.

Stop bit

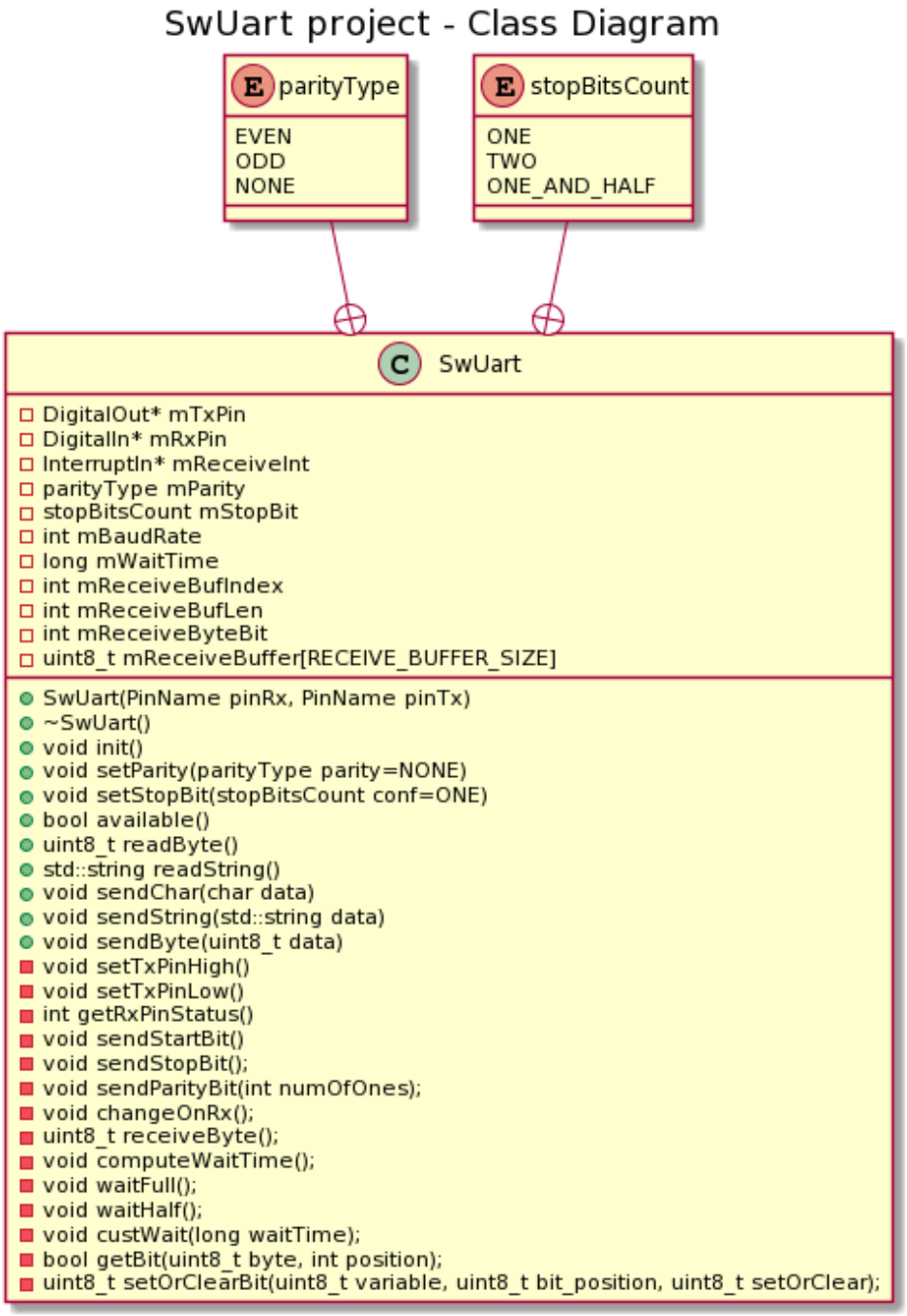
Signalizuje ukončenie prenosu dát. Zariadenie nastaví prenosovú linku z logickej 0 na logickú 1. Dĺžka súvisí s nastavením počtu stop bitov, môže trvať 1,1.5 až 2 cykly

3 Softvérová implementácia

Softvérová implementácia vie pomôcť v prípade, že na zariadení nemáme dosť hardvérových UART-ov, prípadne pri ladení programu.

Naša softvérová implementácia umožňuje príjem aj vysielanie dát. Môže byť nastavená na ľubovoľnom, vstupno-výstupnom pine, podmienkou je, že prijímací pin musí podporovať prerušenia.

Vytvorenú triedu je možné vidieť na nasledujúcom UML diagrame.



Trieda umožňuje pomocou konštruktora vytvoriť UART emuláciu na ľubovoľných pinoch. Umožňuje nastaviť použitie parity, ako aj dĺžku stop bitov. Pre jednoduchosť je baudrate pevne nastavený na 9600 bps a dĺžka dát je pevne stanovená na 8 bitov.

3.1 Popis implementácie

Pre UART je potrebné správne časovanie. Náš program je nastavených na 9600 bps, čo znamená, že jeden čakací cyklus má trvať 104us.

Vysielanie dát prebieha nasledovným spôsobom.

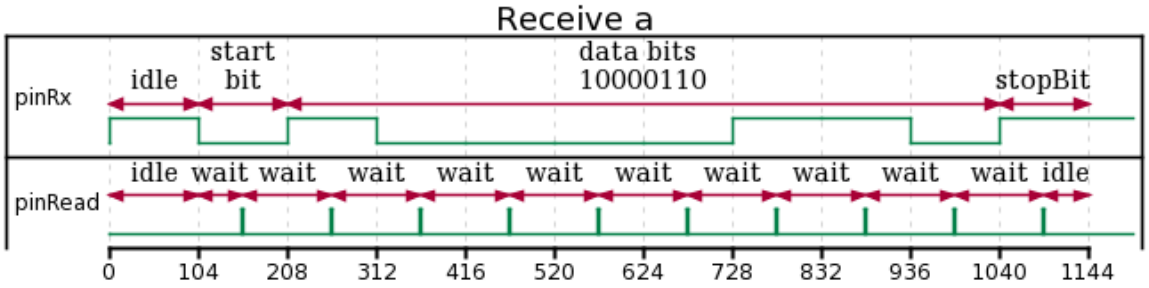
1. pošle sa start bit, ten zmení úroveň na vysielajúcom pine na logickú 0 po dobu jedného cyklu, t.j. 104us
2. začnú sa postupne čítať jednotlivé bity dát od najmenej významného. Ak má bit hodnotu 1, tak sa na výstupnom pine nastaví logická 1, inak sa nastaví 0, po každom nastavení sa čaká jeden cyklus, 104us
3. Ak máme povolenú paritu tak sa zráta počet jednotkových bitov a na základe nastavenej parity sa na výstupnom pine nastaví logická 0 alebo 1
4. pošle sa stop bit, nastaví sa linka z logickej 0 na 1 a čaká sa dobu, ktorá súvisí s tým aký počet stop bitov je nastavený
    a. ONE – 104us
    b. TWO – 208us
    c. ONE_AND_HALF – 156us
5. ak nie sú k dispozícii ďalšie dáta, tak je vysielanie ukončené, inak sa opakuje krok 1.

Prijímanie dát prebieha nasledovným spôsobom.

1. Vznikne prerušenie na prijímacom pine, následne sa čaká pol cyklu, 52us, a prečíta sa hodnota na vstupe. Ak je na vstupe logická 0, vieme že prišiel start bit a pokračujeme ďalším krokom, inak ignorujeme
2. čakáme celú periódu, 104us, a následne čítame stav vstupu. Podľa tejto hodnoty nastavíme príslušný bit a tento krok opakujeme 8 krát
3. ak je nastavená parita, tak na základe bitov si vyrátame očakávanú hodnotu a následne prečítame stav linky. Ak sa hodnoty nerovnajú, vieme že prišlo k zmene dát a dáta ignorujeme.

4. prečítame hodnotu stop bitu, načítané dáta nastavíme do buffera prijatých dát a zvýšime ich počet o 1

Čakacie cykly je možné vidieť na nasledovnom obrázku, kde prijímame písmeno a. Cieľom je čítať stav linky uprostred nastavených bitov aby sa predišlo chybám.



Čítanie dát z buffera:

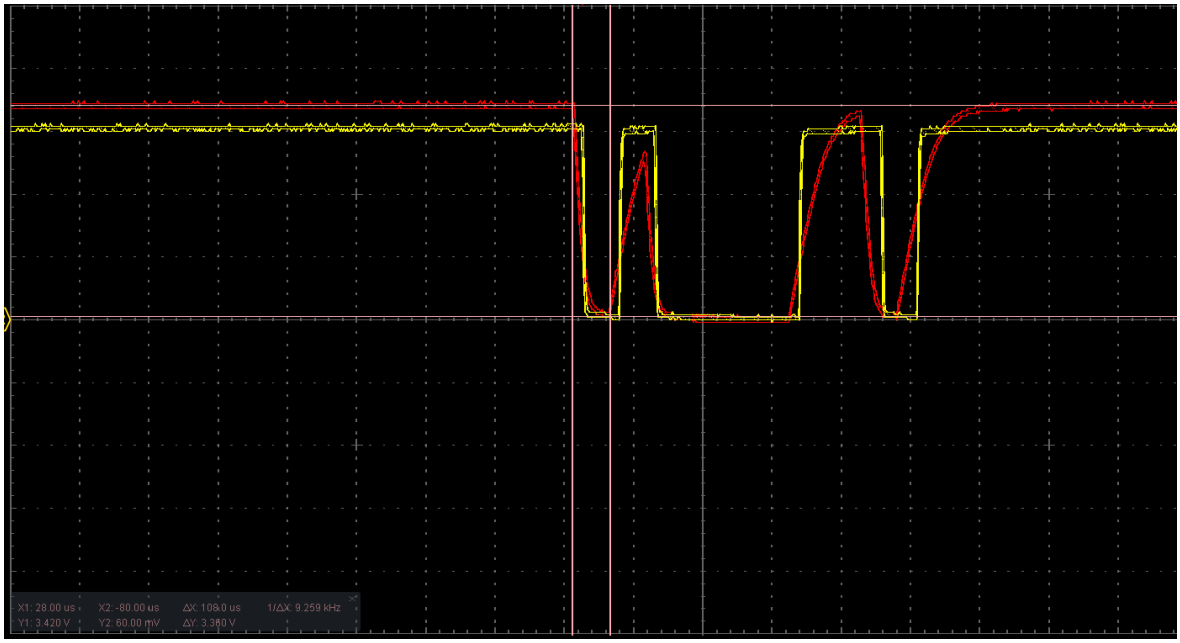
1. ak máme v prijímacom bufferi nejaké dáta, tak metóda available () vráti true. Následne pomocou metódy readByte(), prečítame posledný byte v bufferi a zmeníme jeho index o jedna. Preto môžeme prečítať celý buffer, pokiaľ sú dostupné dáta. Môžeme tiež použiť metódu readString(), ktorá prečíta celý buffer a poskladá z neho text

3.2 Overenie funkčnosti

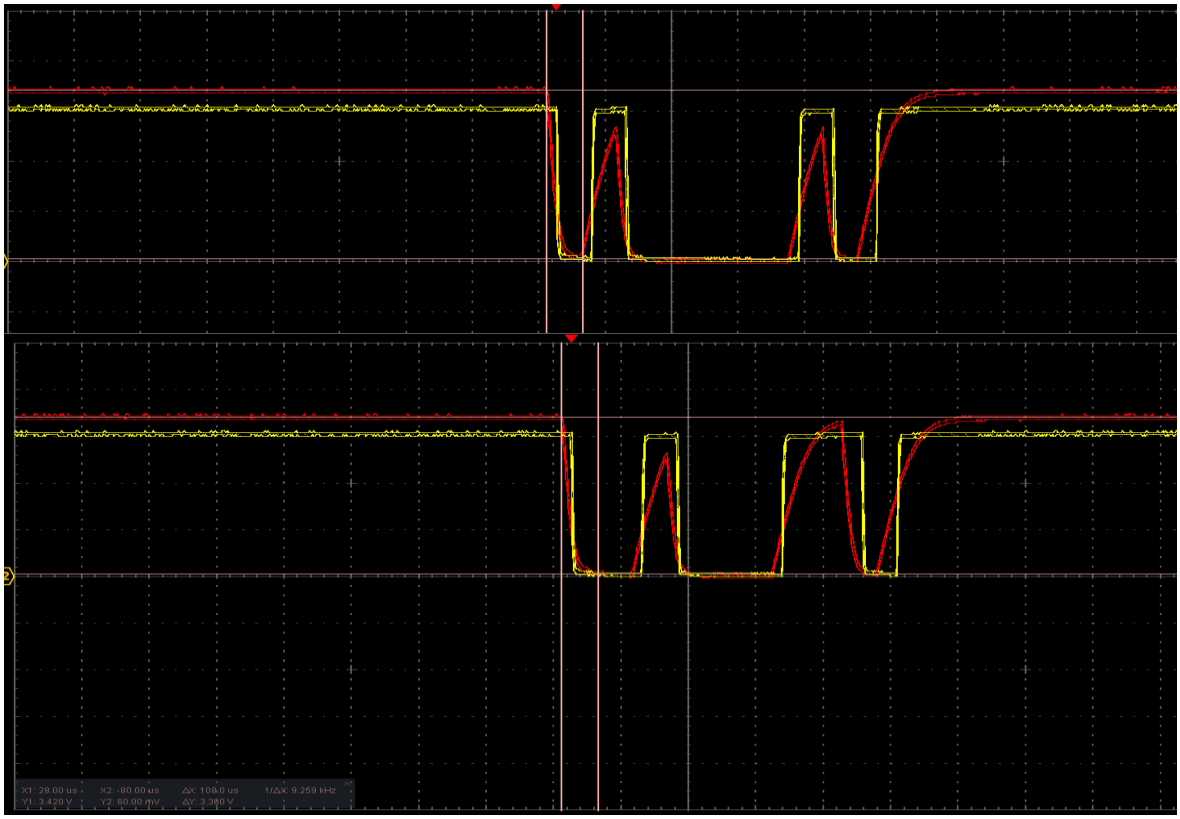
Overenie prebiehalo dvoma spôsobmi.

3.2.1 Tera Term

Prvým bolo použitie USB TO UART konvertora, ktorý sme pripojili na piny PA_1 a PA_2. Následne sme pripojili osciloskop a sledovali sme hodnoty na linke. Testovací program pozostával z príjmu dát z konvertora a následným poslaním týchto dát nazad na konvertor, tým sa vytvorila funkcia echa.



Červenou farbou je signál vyslaný konvertom, žltou je signál vytvorený našou knižnicou. Jeden cyklus trvá 104us čo zodpovedá nastaveniu. Na nasledujúcich obrázkoch je možné vidieť veľké a a písmeno b.



3.2.2 Pomocou hw Uart na doske

Druhý spôsob overenia prebiehal priamo na samotnej doske. Prepojili sme vodiče hw UART-u s pinmi nášho softvérového UART-u. Následne sme pomocou softvérového UART-u posielali text „hello“, ak ho prečítal hw UART, tak sme zmenili stav zelenej LED. Takisto sme pomocou hw UART poslali text „ahoj“, ak ho prečítal softvérový UART , tak sme zmenili stav červenej LED.

Tento proces sa opakoval v cykle každých 5 sekúnd. Výsledkom boli blikajúce LED, čo potvrdilo funkčnosť nášho riešenia.

4 Záver

V rámci tejto práce sme sa do hĺbky zoznámili s UART protokolom, pochopili sme ako funguje a naučili sme sa ho softvérovo implementovať. Navrhli sme triedu, ktorá ho umožňuje emulovať na ľubovoľnom vstupno-výstupnom pine. Pomocou testovania sme overili, že výsledná knižnica je funkčná a použiteľná pre ďalšie projekty.

5 Zdroje

https://os.mbed.com/

https://www.circuitbasics.com/basics-uart-communication/

https://www.analog.com/en/analog-dialogue/articles/uart-a-hardware-communication-protocol.html

https://os.mbed.com/platforms/ST-Discovery-L476VG/