Středoškolská odborná činnost 2006/2007

Obor 10 – elektrotechnika, elektronika, telekomunikace a technická informatika

Tester sériových rozhraní

Autor:
Břetislav Bakala
SPŠ a VOŠ, Karla Čapka 402,
397 01 Písek, třída A4.S

Vedoucí práce:
Mgr. Milan Janoušek
SPŠ a VOŠ, Karla Čapka 402,
397 01 Písek

Zadavatel práce:
Ing. Ladislav Obdržálek
ELO+, s.r.o. Na Spravedlnosti 741,
397 01 Písek

Písek, 2007
Jihočeský kraj

Prohlášení

Prohlašuji, že jsem svou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

Nemám závažný důvod proti užití tohoto díla k výukovým účelům a proti zveřejnění díla na internetových stránkách SOČ.

V Písku dne 31. 3. 2007 podpis: ……

Anotace

Cílem této práce je návrh testeru pro převodníky linky RS-232 na jiná průmyslová rozhraní. Zařízení zajišťuje testování sériových přenosů při rychlostech 1,2 kbaud až 230,4 kbaud, je schopno zachytit chybu přenosu případně nedoručení datové sekvence a podat o tom hlášení uživateli. Součástí nastavení je zvolení použití řídících signálů RTS a CTS dohromady nebo samostatně. Mimo funkci kompletního testu při různých datových sekvencí je zařízení schopno nastavit signály TXD, RXD, DSR, DTR, RTS, CTS, DCD na stálé logické úrovně podle požadavku uživatele. Dále je možné uplně odpojit výstupní signály z portu RTS, TXD, DTR.Ovládání testeru je realizované pomocí softwaru naPC, který komunikuje s testerem pomocí USB. Výsledek testu je možné uložit do textového souboru a detailním zobrazením chyby v přenosu.

Poděkování

Rád bych poděkoval firmě ELO+, s.r.o. za umožnění realizace této práce, rozšíření mých vědomostí a nových dovedností získaných návrhem testeru. Dále vedoucímu práce panu Mgr. Milanu Janouškovi za odborné vedení a rady při práci s mikroprocesory a návrhu plošného spoje a RNDr. Miroslavu Procházkovi za konzultace při řešení problémů v C++ Builderu.

Obsah

1 Úvod

Z témat podaných firmou ELO+, s.r.o. na SPŠ a VOŠ Písek jsem si vybral návrh testeru převodníků sériových rozhraní. Práce byla původně zadána jako dvě témata pro Středoškolskou odbornou činnost s jejím následným využitím v praxi ve firmě ELO+, s.r.o. k testování jejich výrobků. Softwarová část, obsahovala ovládací rozhraní pro tester a druhé téma obsahovalo návrh vlastního testeru. Já jsem tyto dvě úlohy sloučil do jedné a navrhnul řešení včetně vzájemné komunikace.

1.1. Dosavadní tester

Stávající systém používá pro testování starý typ PC 486 na který je připojen původní tester. Sériový přenos je generován ve dvou sériových portech počítače a pomocí testeru je buď vpuštěn do testovaného převodníku nebo provede akci podle nastavení ovládacím softwarem pro OS DOS propojeným s testerem pomocí pararelního portu.

Stávající tester už neodpovídá potřebám firmy ELO+, s.r.o. zejména kvůli složitosti ovládání pomocí příkazového řádku, použitím dvou sériových a zároveň jednoho pararelního portu, kdy při vysílání definované sekvence může dojít k přerušování v důsledku jiných probíhajících procesů na PC. Jeden z dalších důvodů obnovy zařízení je modernizace na současnou HW platformu, připojení přes USB port a nedostatečná rychlost testování, která je u klasického sériového portu maximálně 115200 baudů.

1.2. Požadované schopnosti zařízení

Firma ELO+, s.r.o. měla při zadávání tyto požadavky:

- vytvoření testeru připojitelného přes USB s jednoduchým ovládacím softwarem, který zároveň podává výsledky testu
- verze testeru pro rozhraní RS-232 a pro rychlosti dané tímto rozhraním
- tester musí být schopen plynule vyslat určitou sekvenci dat (paket) danou rychlostí, která je uživatelsky definovaná a vysílání této sekvence opakovat
- datová sekvence může být posílána v řežimu full-duplex nebo half-duplex s použitím harwarového řízení RTS a CTS, tyto možnosti musí být definovatelné
- vyhodnocení času přenosu paketu a určení zda byla doručena v limitu
- umožnit úplné odpojení vysílacích vývodů a pevné nastavení na stav logická nula nebo jednička

- sériový port testeru musí obsahovat signály TXD, RXD, RTS a CTS, ostatní signály standardu jsou nepovinné

2. Analýza zadání

Z pospisu původního testeru a zadání práce vyplývají dílčí úlohy:

- Zajistit v požadované rychlosti plynulý tok vysílaných dat a jejich současný plynulý příjem. Pro potřeby testování je možno omezit celkové množství testovacích dat v jednom plynulém toku na určitý objem. Tím se tato úloha dá zjednodušit na jakýsi krok testu o určíté velikosti. Úkolem navrhovaného testeru by bylo přijmout, zpracovat a poslat výsledky kroku testu. Řízení celého testu by mohlo být úkolem počítače, kde by bylo možno jednotlivé kroky nastavit, určit testovací rychlost a výsledky jednotlivých kroků testu vyhodnotit

- Vytvořit plynulý tok dat během kroku testu na dvou rozhraních RS 232 ve full a half duplexním režimu v požadovaném rozsahu rychlostí. Z důvodu závislosti na správě úloh operačního systému Windows není vhodné tento úkol řešit původní koncepcí a je nutno zajistit obsluhu dvou seriových portů

- Navrhnout vhodnou komunikaci mezi operačním systémem Windows, rozhraním USB a testerem. HW obsluha portu USB je vzhledem ke složitosti rozhraní náročná, přijmutá data je nutno v dostatečné rychlosti zpracovávat. Přenášená data bude nutno rozdělit na řídící data určující chování testeru v dalším kroku, testující sekvence dat, přijmutá data z testovaného zařízení a další stavové signály.

- Navrhnou technické řešení testeru umožňujícím komunikaci s počítačem přes rozhraní USB, řízení a plynulou komunikaci během kroku testu na testovaných rozhraních RS 232. Z výše popsaných úloh, které má nový tester vykonávat bude vhodné použít řešení s mikroprocesorem, který má dva seriové porty a v ideálním případě přímo port USB, nebo paralelní port s možností připojení vhodného převodníku na USB. Mikroprocesor bude nutno naprogramovat . Předpokládám použití mikrokontrolérů PIC firmy Microchip.

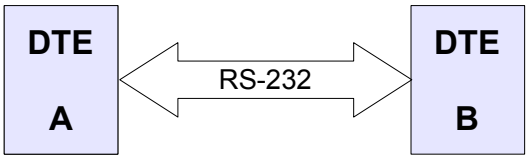
- Vytvořit ovládací software pro platformu Windows s možností jednoduchého grafického uživatelského rozhraní.

3. Návrh

3.1. Stručný popis RS-232C

Rozhraní RS-232C je určeno pro připojení zařízení na vysílání nebo přijem dat (DTE – počítač, tiskárna nebo jiné periferní zařízení) ke koncovému zařízení datových kanálů(DCE – modem). Cílem je spojit 2 zařízení DTE přes jinou linku než je RS-232C(plné schéma zapojení na obrázku 1). Můžeme propojit zařízení DTE přímo mezi sebou pomocí tzv. nulového modemu (viz. obrázek 2). Standard určuje řídící signály rozhraní, přenos dat, elektrické rozhraní a typy konektorů. Standard nabízí synchornní i asynchronní přenos dat, ale porty COM podporují pouze asynchronní režim.

Obrázek 1. Plné schéma připojení RS-232C



Obrázek 2. Připojení RS-232C pomocí nulového modemu

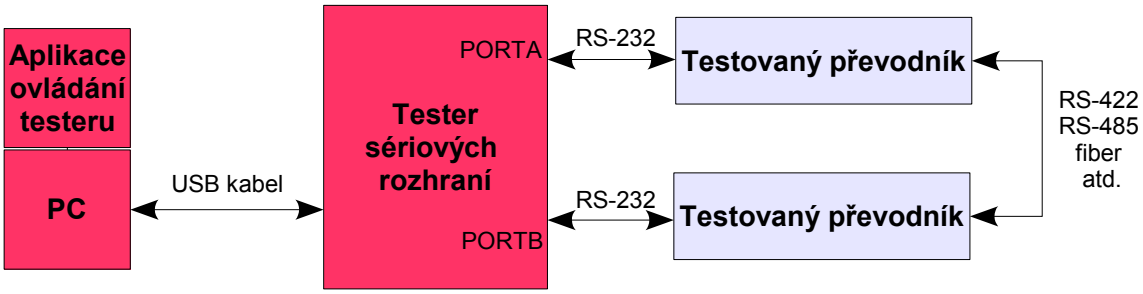
Standard RS-232C definuje asymetrické vysílače a přijímače, u kterých je signál přenášen relativně ke společnému vodiči(uzemněnému obvodu). Rozhraní nepockytuje galvanické oddělení propojovaných zařízení. Napětí v rozsahu -12 V až -3 V odpovídá logické jedničce a napětí od +3 V do +12 V logické nule na signálech RXD(viz tabulka 1). Na signálech TXD(viz tabulka 1) jsou místo hodnot +3 a -3 V prahové hodnoty+5 a -5 V. U řídících signálu potom odpovídá rozmezí +3 V až +12 V stavu ON a -12 V až -3 V stavu OFF. Bližší popis standardu je umístěn na internetové adrese http://rs232.hw.cz.

Tabulka 1: Popis vybraných signálů RS-232C

TXD	Sériová data, výstup vysílače
RXD	Sériová data, vstup přijímače
RTS	Stav ON informuje modem, že terminál má data k přenosu.
CTS	Povolení terminálu, aby odeslal data. OFF zakáže přenos dat. Signál se používá k hardwarovému řízení datových proudů
DSR	Vstup signálu indikuje, že zařízení je připraveno přenášet data
DTR	Výstup signálu informuje, že je terminál připraven vyměnit data.
DCD	Vstup signálu oznamuje, že byl detekován nosný signál vzdáleného modemu.

3.2. Princip řešení

Navrhované řešení se skládá z hardwarové a softwarové části. Softwarová část běží pod operačním systémem Windows na počítači a má za úkol ovládání testeru a zobrazení výsledků testování. Hardwarová část je technicky řešena novým externím zařízením s vlastním napájecím zdrojem. Tento tester je spojen s počítačem pomocí kabelu USB A / USB B. Testované prvky je možno připojit k testeru pomocí dvou rozhraní RS-232. Testují se vždy 2 stejné převodníky. Mezi dva testované převodníky může být vřazen libovolný počet jiných převodníků, podvníkou je však, že na obou koncích testované linky musí být výstup dle standardu RS-232. Jednotlivé části testovacího zařízení a jejich zapojení je vidět na blokovém schématu(viz. obrázek č.3).



Obrázek 3: Blokové schéma zapojení testeru

Softwarová část

V aplikaci pro ovládání testeru může uživatel provádět veškeré nastavení testeru a testu. Aplikace obsahuje přehled nastavení testu, tlačítka pro ovládání testu, výpis výsledků testu, popřípadě více testů prováděných za sebou. Součástí základního okna aplikace je i odpojování jednotlivých výstupních vývodů a jejich nastavování do stavů „1“ nebo „0“. U těchto prvků nalezneme i informace o stavu připojení nebo odpojení vývodů stavů signálů.

Hardwarová část

V hardwarové části(Testeru sériových rozhraní) jsou prováděny testy podle uživatelského nastavení a provádí se zde úkony popsané v předchozím odstavci.

3.2.1. Komunikace mezi hardwarem a softwarem

Důležitým prvkem je komunikace mezi oběma částmi testeru. Jelikož podmínkou firmy byla využít pro komunikaci sběrnici USB, zvolil jsem jako nevhodnější způsob převodu signálu USB na paralelní sběrnici mikroprocesoru s možností využití některých z převodníků USB na sériový nebo pararelní(FIFO) port, protože zpracování signálů v mikroprocesoru je mnohonásobně jednodušší, než řešit komunikaci přímo pomocí USB. V počítači přistupujeme k médiu jako k virtuálnímu sériovému portu. Tento způsob lze

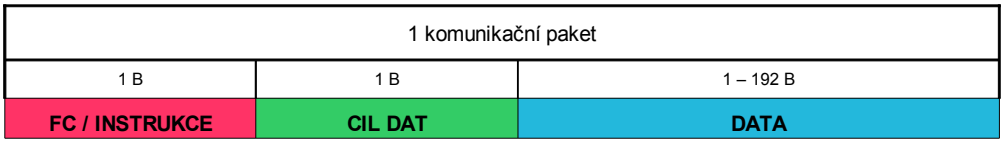
využít, protože nejsou při komunikaci požadované velké rychlosti.

Důležitou součástí komunikace je určení si nějakého protokolu, podle kterého bude probíhat přenos informací. Komunikaci jsem navrhl pomocí osmibitových instrukcí, po jejichž přijetí vykoná tester požadovaný úkon. Hodnoty instrukcí a jejich význam jsou uvedeny v tabulce 2.

Tabulka 2: Seznam instrukcí testeru

Hodnota instrukce (HEX)	Význam instrukce
00	Odpojit vývod TXD1
01	Připojit vývod TXD1
02	Odpojit vývod TXD2
03	Připojit vývod TXD2
04	Odpojit vývod DTR1
05	Připojit vývod DTR1
06	Odpojit vývod DTR2
07	Připojit vývod DTR2
08	Odpojit vývod RTS1
09	Připojit vývod RTS1
0A	Odpojit vývod RTS2
0B	Připojit vývod RTS2
0C	Nastavit TXD1 do logické 0
0D	Nastavit TXD1 do logické 1
0E	Nastavit TXD2 do logické 0
0F	Nastavit TXD2 do logické 1
10	Nastavit DTR1 do logické 0
11	Nastavit DTR1 do logické 1
12	Nastavit DTR2 do logické 0
13	Nastavit DTR2 do logické 1
14	Nastavit RTS1 do logické 0
15	Nastavit RTS1 do logické 1
16	Nastavit RTS2 do logické 0
17	Nastavit RTS2 do logické 1
28	Nastavit test do full-duplexního režimu
29	Nastavit test do half-duplexního režimu
2A	Zapnout používání HW řízení RTS
2B	Vypnout používání HW řízení RTS
2C	Zapnout používání HW řízení CTS
2D	Vypnout používání HW řízení CTS
2E	Odeslat do PC status připojení vývodů a stavů signálů
30	Spustit TEST
31	Paket nedoručen v časového limitu
32	Přepnout ze sériového režimu do režimu nastavování stavů
FC	Hlavička datového paketu

V seznamu instrukcí testeru(viz. tabulka 2) je jedna hexadecimální hodnota instrukce FC . Tato instrukce určuje, že následující přijaté informace nebudou instrukce, ale data. O jaký typ dat a kam mají být uloženy informuje tester hodnota dalšího bytu CIL DAT. Další přijaté byty z PC jsou data. Komunikační paket znázorňuje obrázek 4.



Obrázek 4: Struktura komunikačního paketu

Pole bytů DATA může nabývat různé délky. Při přenosu testovacích nastavení do testeru je délka paketu daná hodnotou bytu CIL DAT a algritmus pro příjem těchto paketů určuje podle jejich struktury kdy má být ukončen(viz. tabulka 3). Dalším typem dat může být uživatelsky nastavená testovací sekvence s proměnnou délkou, v takovém případě se před odesláním sekvence pošle paket obsahující délku sekvence např FC 10 (viz. tabulka 3).

Tabulka 3: Význam komunikačních paketů

FC	CIL DAT	DATA	Význam paketu při toku z PC do TESTERU
FC	1	SPBRG + SPBRGH	Zapnout rozhrani USART o dané rychlosti
FC	2	switchTime	Nastavení časové prodlevy mezi přepnutím směru při Half-duplexním přenosu
FC	10	pocet_bytu	Délka testovacího paketu
FC	11	max 192 B dat	Testovací datová sekvence určena pro odeslání na RS-232 port A a B
			význam paketu při toku z TESTERU do PC
FC	12	Delka paketu1 + Delka paketu2 + TIMERL_C1 + TIMERH_C1 + TIMERL_C2 + TIMERH_C2	Délky přijatých dat na protech A a B + čas potřebný pro přijetí jedné sekvence
FC	13	max192 B dat	Data přijatá na Portu B
FC	14	max192 B dat	Data přijatá na Portu A
FC	20	stat_rele + stat_1 + stat_2	Status nastavení testeru

3.2.2. Průběh testu(způsob testování)

Ze zadání je zřejmé, že testovací sekvence musí být odeslána plnyule a najednou. Z tohoto faktu vyplývá následující princip odesílání a vyhodnocování testovacích paketů.

Jako jeden test jsem zvolil přenesení jednoho uživatelsky vytvořeného testovacího paketu o maximální délce 192B[1] a jeho vyhodnocení. Tento test se podle nastavení opakuje až do ukončení uživatelem, nebo se provede pouze jednou. Při suštění testu se tedy nejprve připojí vývody používané v testu a potom se odešle do paměti testeru testovací paket, který je odeslán na porty A a B testeru. Data přijatá na portech se ukládají do paměti testeru a po přijetí všech odeslaných bytů testovacího paketu se odesílají z paměti do PC, který vyhodnotí jejich správnost. V případě že data nesouhlasí, aplikace vypíše chyby a ukončí testování, v případě že data souhlasí tak podle nastavení opakování testu vydá pokyn(odešle instrukci s hodnotou 30 viz. kapitola 2.1.2. tabulka 1) k provedení nového

1 délka paketu byla zvolena v závisloti ba velikosti čítače mikroprocesoru a nejmenší rychlosti přenosu 1,2 kbaudů

testu se stejnými parametry a testovacím paketem a celý proces se opakuje. Po ukončení každého testu se zobrazí celková statistika testování, tzn. kolik bylo provedeno testů, kolik paketů bylo přenesených špatně a kolik dobře.

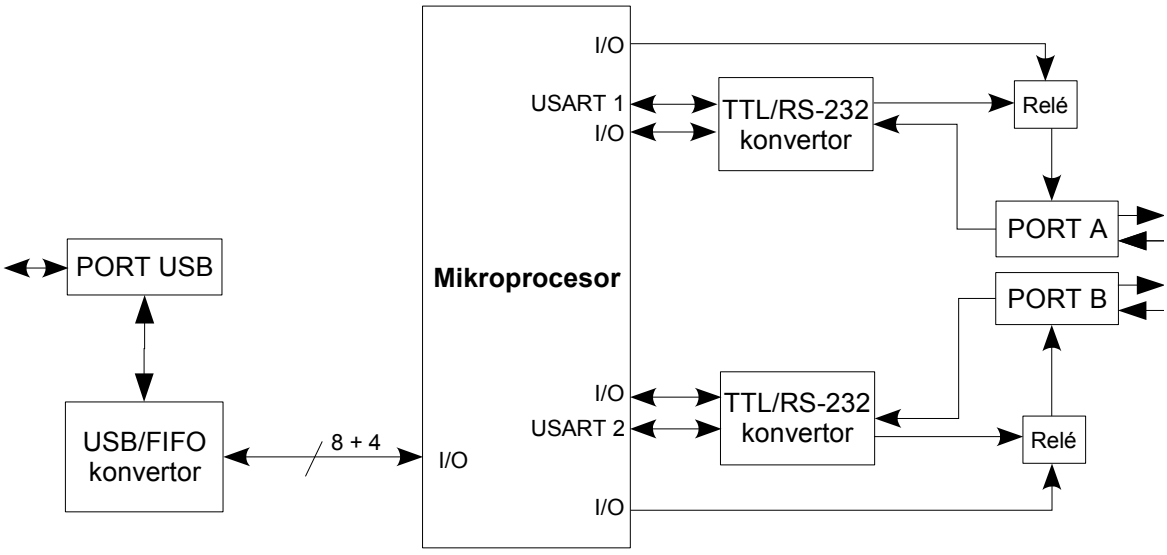
Full-duplexní režim testu znamená odeslání testovacího paketu z obou portů najednou, zatímco při half-duplexním řežimu se odešlou data nejprve z portu A a po přijetí na portu B se odesílají data z portu B. Po přijetí všech dat, tzn. dat na portu A a následně na portu B se odesílají přijaté pakety do PC k vyhodnocení jejich správnosti.

Při obou režimech se při zapnutém HW řízení nejprve nastaví signál RTS portu který je připraven vysílat a poté před odeslání každého bytu v paketu testuje zda je nastaven signál CTS do logické 1. Tento způsob řízení odpovídá zařízení DTE jako je například sériový port COM v počítači.

Na výstupních signálechTXD, DTR, RTS je snížena úroveň výstupního napětí, aby bylo poznat, zda dokáže testovaný převodník rozeznat slabší signál.

3.3. Návrh hardwaru

3.3.1. Blokové schéma



Obrázek 5: Blokové schéma testeru

USB/FIFO konvertor zajišťuje přenos osmibitové informace z procesoru a zpět pomocí USB do virtuálního portu v PC, je proto důležitým blokem zapojení. Mikropocesor je nedílnou řídící a paměťovou jednotkou testeru, spouští test, pomocí I/O portu ovládá relé

pro odpojení výstupních vývodů, nastavuje jednotlivé signály a odesílá naměřená data zpět do PC. Knovertory TTL/RS-232 slouží ke změně napěťových urovní z TTL 5V logiky na normu RS-232.

3.3.2. Výběr základních integrovaných obvodů

Mikroprocesor

Nejdůležitějším prvkem celého testeru je mikroprocesor. Požadavkem na procesor byl dostatečný počet vstupně/výstupních bran, 2 sériové porty USART a vysoká taktovací frekvence procesoru.

S ohledem na složitost návrhu a na související požadavky na procesor jsem zvolil mikrokontrolér Microchip PIC18F6722. Hlavním kritériem výběru tohoto obvodu byly 2 seíriové porty EUSART[2]. Další vlastnosti mikroprocesoru jsou uvedeny v tabulce 4.

Tabulka 4: Microchip PIC18F6722

Paměť flash	128 kB
Paměť RAM	3936 B
Taktovací frekvence	40 MHz (10 MIPS)
Časovačů 16-bit/8-bit	3/2
I/O portů	A, B, C, D, E, F, G
Zdrojů přerušení	28
Priority přerušení	2

USB/FIFO konvertor

Výběr tohoto převodníku nebyl složitý, protože na trhu se jich nevyskytuje takové množství jako mikrokontrolérů. Zvolil jsem si výrobce FTDI Chip a obvod FT245BM. Výběr byl dán také dostupností integrovaného obvodu přímo ve firmě ELO+, s.r.o. Zapojení obvodu je jednoduché a splňuje kladené nároky. Potřebné ovladače pro PC jsou volně stažitelné na internetové adrese výrobce www.ftdichip.com. Ovladače je nutné ručně nainstalovat ještě před připojením obvodu k USB. Základní informace o obvodu jsou uvedeny v tabulce 5.

2 *Enhanced Universal Synchronous Receiver Transmitter – rozšířený unverzální synchronní přijímač vysílač*

Tabulka 5: FTDI Chip FT245BM

Rychlost přenosu	až 300kbit/s při VCP[3]
Rychlost přenosu	až 1Mbit/s při D2XX[4]
Vstupní vyrovnávací paměť	128 B
Výstupní vyrovnávací paměť	384 B
Kompatibilita	USB 1.1, USB 2.0
Podporované OS	Windows 98/2000/XP, Linux, Mac OS

TTL/RS-232 konvertor

Protože logická 1 má v RS-232 úroveň napětí -15V až -3V a logická 0 úroveň 3V až 15V musíme na tyto hodnoty převést logický signál vedený z mikroprocesoru K tomu jsem použil asi nejznámější a nejpoužívanější obvod MAX232A. Obsahuje 2 konvertory pro výstup na linku a 2 pro vstup z linky RS-232 . Výhodou tohoto konvertoru je podporovaná rychlost 200 kbit/s, která se občas jako nadstandard také využívá. Kondezátory pro napěťové pumpy nemusí být 1μF jako u normálního MAX232, ale stačí použít 100nF s menším pouzdrem v keramickém provedení.

Relé

Typ relé jsem měl dán firmou ELO+, s.r.o. Použil jsem 5V přepínací relé EZ0015. Tyto relé byly použity ve staré verzi testeru a pro novou plně dostačují.

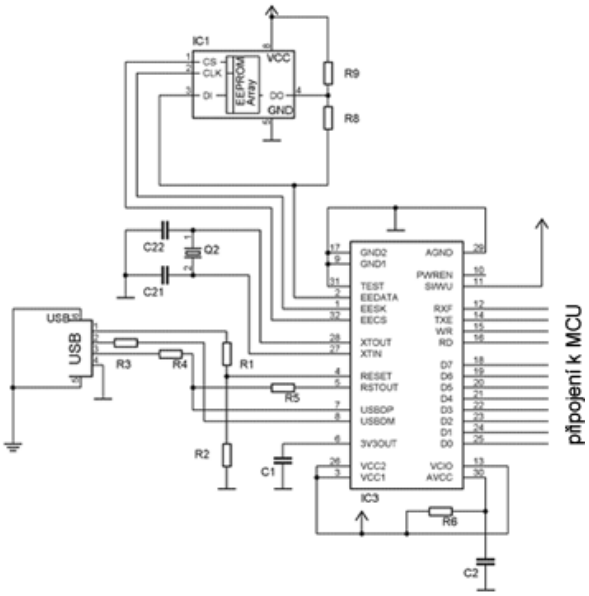
3.3.3. Návrh obvodu

Celé schéma zapojení testeru, popsané v této kapitole, je v příloze 1.

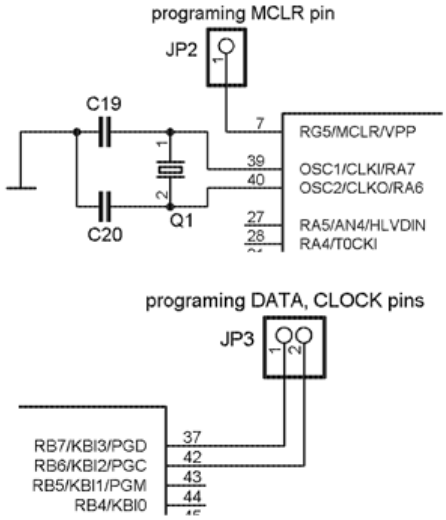
Základní jednotkou je mikrokontrolér PIC18F6722. Na vývodech 39 a 40 je připojen externí krystal k vnitřnímu oscilátoru(viz.obrázek 7). Kmitočet krystalu je 10 MHz, který je po zpracování ve vnitřním oscilátoru mikroprocesoru 4x násoben fázovým závěsem PLL. Mikrokontrolér má vyvedené programovací piny MCRL, DATA, CLOCK(viz. obrázek 7). Navíc jsem do základního obvodu mirkoprocesoru přidal na volné vývody 35, 34, 33 piny pro libovolnou aplikaci, například pro připojení LED nebo pro ladící účely při programování.

Na vstupně/výstupní bránu E jsou připojeny datové signály převodníku USB/FIFO, na první 4 výstupy brány D potom řídící signály. Připojení USB a paměti EEPROM k FT245BM(viz. obrázek 6) je realizováno podle doporučeného katalogového zapojení pro oddělené napájení v technické dokumentaci k obvodu.

3 *Virtual Com Port – virtuální sériový port v PC*
4 *USB direct drivers – přímý přístup k USB pomocí dll knihoven*

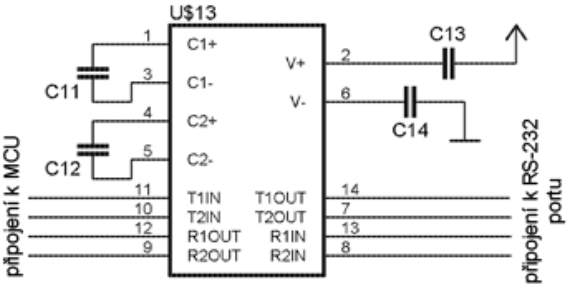


Obrázek 6.: Zapojení FT245BM



Obrázek 7.: Zapojení krystalu MCU a programovacích pinů

Na výstupní a vstupní piny USART1 a USART2 jsou připojeny obvody MAX232A, ze kterých výstupní signály vedou přes odpojovací relé na 9-pinový konektor a vstupí signály do testeru přímo z převodníku na konektor. Tím je zajištěno vypínání výstupních signálů z testeru do testované linky. Do výstupních signálů byl ještě vřazen odpor 1kΩ kvůli snížení výstupní úrovně napětí. Ostatní řídící signály portů jsou vedeny z I/O[5] pinů přes konvertory úrovní MAX232A(obrázek 8) na piny jednotlivých konektorů. Na které I/O[5] piny jsou signály připojeny je určeno návrhem plošného spoje, aby byly spoje jednoduše taženy.



Obrázek 8.: Zapojení MAX232

Relé nemůžou být k mikrokontroléru zapojena přímo, protože jejich spínací proud by mohl zničit I/O[5] brány kontroléru. Použil jsem proto budič sběrnice 74HC245 pro který je také navržený plošný spoj. V kapitole Oživení 2.2.5. je popsána zkušenost s použitím tohoto obvodu pro buzení spínací cívky relátek.

5 *I/O – Input/Output – vstupně výstupní brána nebo vývod*

Napájení je realizované pomocí stabilizovaného 5V vnějšího adaptéru. Před vstupem napájení do dalších obvodů je vřazena dioda v propustném směru, aby v případě otočení polarity napájení nedošlo ke zničení veškeré elektroniky.

3.3.4. Návrh plošného spoje

Z důvodů použití již vyrobené krabičky na jeden z výrobků firmy ELO+, s.r.o. bylo potřeba dodžet při návrhu splošného spoje rozměry plošného spoje a umístění konektorů, montážních děr a indikační led diody pro napájení na DPS[6]. Umístění součástek jsem provedl podle návrhu plošného spoje, pro který je krabička určena.

Kromě výše zmíněných požadavků bylo nutné při návrhu v programu EAGLE r4.16 dodržet tato pravidla:

- používat pravoúhlé nebo 45 stupňové vedení spojů,
- nevytvářet duplicitní spoje[7]
- dodržet dostatečný odstup a tlošťku spojů[8](odstup spojů byl v tomto případě dán roztěčí vývodu mikrokontroléru)
- vyplnění prázdných ploch spojem připojeným na GND[9]
- dodžet mřížku při vytváření nových knihoven v návrhovém programu

Schéma plošného spoje je v příloze 2.

Součástí návrhu plošného spoje bylo i vytvoření knihovny pro součástky PIC18F6722 s pouzdrem TQFP 64, MAX232A v pouzdře SOIC150“ a pro relé EZ0015 .

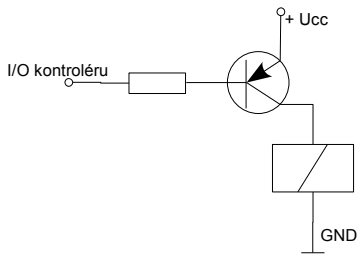
3.3.5. Oživení

Po osazení plošného spoje[10] a připojením k napájení se rozsvítila indikační dioda napájení PWR. Po proměření klíčových napětí na napájecích pinech integrovaných obvodů jsem považoval hardware za funkční. Klíčový okamžik nastal v momeňtě, kdy jsem pomocí první verze firmwaru sepnul všechna relé současně. Integrovaný obvod 74HC245 se začal zahřívat. Zařízení jsem vypnul a proměřil odpory spínacích cívek relátek. Jelikož jsem při

6 *DPS – deska plošného spoje*
7 *stejné signály umístěné nad sebou*
8 *udává třídu přesnoti pro výrobu DPS a také výrobní cenu plošného spoje*
9 *GND – ground - zem*
10 *Osazený plošný spoj je vidět na fotografii v příloze 3*

návrhu předpokládal, že všechna relátka mají stejný odpor, tak jsem nepředpokládal toto chování, ale jak se změřením odporů ukázalo relé byly vyrobeny podle dvou norem. Jedna má odpor spínací cívky 80Ω a druhá 160Ω. Při návrhu jsem změřil pouze jedno relé s hodnotou 160Ω a předpokládal, že jsou stejná. Tento stav vedl ke zvýšení proudu procházejícím každým vývodem integrovaného obvodu.

Tester byl funkční včetně spínání relátek, ale nespolehlivý, rozhodl jsem se tedy vyměnit IO 74HC245 za 6 spínacích tranzistorů. Jelikož jsem měl relé zapojené na spínání proti zemi, použil jsem univerzální tranzistory BC327. Jedná se o tranzistor typu PNP v zapojení podle schématu na obrázku 9. Hodnotu odporu jsem podle vlastností tranzistoru uvedených v katalogovém listu zvolil 5K6.Realizace tohoto zapojení na plošném spoji je vidět na fotografii v příloze 3.



Obrázek 9.Zapojení tranzistoru pro buzení relé

3.4. Firmware testeru

Další velmi důležitou částí testeru je firmware nahrávaný do mikrokontroléru. Firmware řídí veškerou činnost mikrokontroléru a rozlišuje jaká instrukce přišla a který příkaz vykonat.

3.4.1. Výběr prostředí a programovacího jazyka

S použitím řady mikrokontrolérů PIC18 se nabízí možnost programovat firmware v jazyku C. S výběrem programovacího jazyka souvisí i výběr vývojového prostředí.

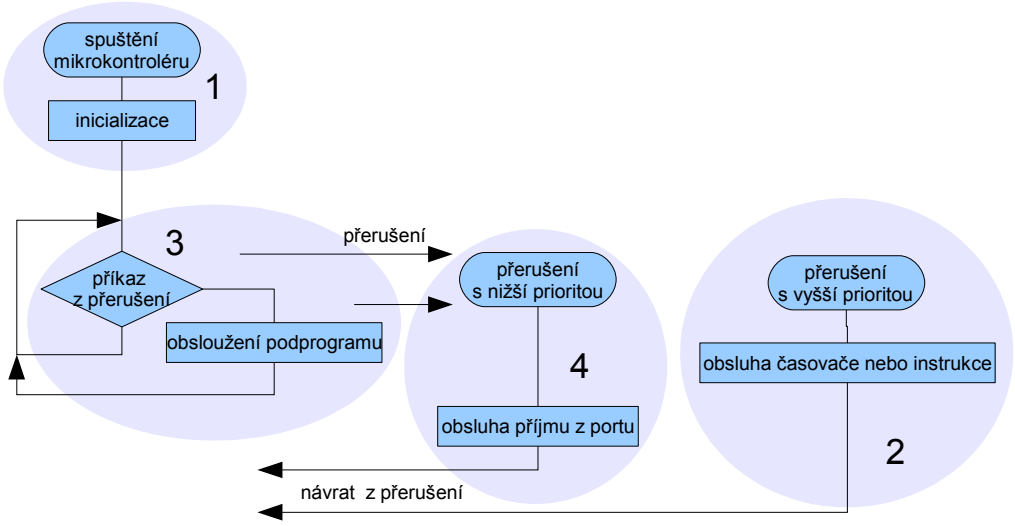
Možnost volby jsem měl z HI-TECH PICC, což je vývojové prostředí a úprava jazyka C pro mikrokontroléry. V nekomerční verzi HI-TECH PICC je program pro řadu PIC18 omezen na maximální velikost programu 8kB.

Další nástroj, ten který jsem si zvolil, je jazyk C18 implementovaný do vývojového prostředí MPLAB od Microchipu. V jazyce C18 od Microchipu je omezen na dva měsíce funkčnosti optimalizace přeloženého kódu.

Programování mikrokontroléru v jazyce C může vést ke zbytečným příkazům a složitějším algoritmům při překladu do assembleru, ale v poměru ke zjednodušení programování složitějších aplikací je to zanedbatelné.

3.4.2. Základní struktura firmwaru

Běh programu by se dal rozdělit na 4 části(obrázek 10) podle procesů a úloh které řeší.



Obrázek 10. Základní struktura fimrwaru

Část 1 - INICIALIZACE

V této části dojde ke spuštění mikroprocesoru a proběhne proces inicializace mikrokontroléru. Nastaví se směry všech vstupně/výstupních bran, provede se nastavení potřebných přerušení a jejich priorita, nastavení 16-bitového módu pro časovač a pro taktovací děličku sériového portu.

Část 2 – OBSLUHA PŘERUŠENÍ S VYŠŠÍ PRIORITOU

Zde dochází k příjmu instrukcí z PC. Obsahuje rutinu pro příjem bytu z převodníku USB/FIFO. Pokud je instrukce jednoduchá provede se ihned v přerušení, ale pokud se jedná o složitější instrukci nastaví se pouze hodnota podle které pozná nekonečná smyčka v části 3 jaký podprogram má spustit. Další úlohou toho procesu je obsluha přerušení z přeplněného časovače, což znamená že testovací paket nebyl doručen v časovém limitu.

Část 3 – HLAVNÍ CYKLUS

Po inicializaci se program dostane do části 3 a zacyklí se do nekonečné smyčky, ve které je připraven vykonávat podprogramy určené přijatou instrukcí přerušením v části 2.

Část 4 – OBSLUHA PŘERUŠENÍ S NIŽŠÍ PRIORITOU

Poslední 4. částí je obsluha přerušení s nižší prioritou. Tato rutina se stará pouze o příjem testovacích bytů na portech testeru A a B. Rozlišuje také zda byli přijat stejný počet bytů jako počet odeslaných bytů.

3.4.3. Inicializace

Hned po spuštění mikrokontroléru se nakonfigurují všechny potřebné registry. Konfigurace probíháv následujícím pořadí:

Za prvé je potřeba nastavit směr jednotlivých vstupně/výstupních pinů. Nastavení provedeme pomocí registrů TRISx. Za písmeno „x“ dosadíme písmeno dané brány, kterou chceme nastavovat. K jednotlivým vývodům brány potom přistupujeme jako k bitům registru TRISx. Pokud je bit nastaven na logickou 0 je daný vývod výstupní, pokud na logickou 1 je vývod vstupní. V testeru je provedeno nastavení následujícím způsobem:

```
TRISC = 0x80;
TRISA = 0x00;
TRISB = 0x05;
TRISD = 0xC3;
TRISE = 0xFF;
TRISF = 0x14;
TRISG = 0x14;
```

Dalším bodem inicializace je nastavení přerušení a jejich priority.

```
INTCON = 0b11110000;
INTCON2 = 0b00000100;
INTCON3 = 0x00;
T0CON = 0b00000111;
PIR1 = 0x00;
PIR2 = 0x00;
PIR3 = 0x00;
PIE1 = 0b00100000;
PIE2 = 0x00;
PIE3 = 0b00100000;
IPR1 = 0x00;
IPR2 = 0x00;
IPR3 = 0x00;
RCON = 0b10000000;
```

Výše uvedený kód nastavuje použití priority přerušení, povoluje globální přerušení, přerušení při přijetí bytu na portech EUASRT 1 a 2, přerušení při přetečení hodnoty v časovači/čítači a externí přerušení na pinu RB0(první pin brány B). Priority přerušení čítačem a při sestupné hraně na RB0 je nastaveno na vyšší prioritu a přerušení od EUSART 1 a 2 je nastaveno na nižší prioritu. Priority jsou nastaveny tak, aby nemohlo přerušení EUSARTU zamaskovat přerušení časovače, ke kterému by nedošlo a tester by nerozeznal zda paket dorazil v limitu, či ne. Priorita externího přerušení na RB0 je dána výrobcem v architektuře procesoru vždy na vyšší prioritu.

Důležité je také nastavení časovače který musí počítat nejdelší čas který může. Toho docílíme nastavením 16-bitového módu čítače a předděličky na hodnotu 256. Nastavení provedeme následujícím přiřazením.

```
T0CON = 0b00000111;
```

Nastavení 16 bitové deličky pro časování EUSARTů provedeme takto:

```
BAUDCON1bits.BRG16 = 1;
BAUDCON2bits.BRG16 = 1;
```

Na konci inicializace nastavíme také výchozí hodnoty registrů. Odpojíme všecha relé a Vynulujeme výstupní řídící signály a registry pro testovací paket. Nulování registrů testovacího paketu provedeme následujícím jednoduchý cyklem.

```
for(x = 0; x <= 192; x++){
  usart1IN[x] = 0;
  output[x] = 0;
  usart2IN[x] = 0;
}
```

3.4.4. Definování vlastních funkcí

Kódy vužívané v programu častěji je vhodné definovat jako funkce, které potom voláme v potřebném okamžiku. Definoval jsem si rutiny pro odesílání dat přes USB do PC. Celkem jsem potřeboval 4 funkce:

otevri_send ()

```
static void otevri_send (){
    NTCON = INTCON & 0b11101111;
   TRISE = 0x00;
   Delay10TCYx (1);
}
```

Funkce potřebná pro správnou funkčnost funkcí *usb_send ()* a *usbdata_send ()*. Její zavolání otevírá odchozí komunikační kanál a zakazuje v tomto okmžiku externí přerušení využívané pro indikaci příchozích dat z PC.

zavri_send ()

```
static void zavri_send (){
   TRISE = 0xFF;
   Delay10TCYx (1);
          INTCON = INTCON | 0b00010000;

}
```

Ukončení odchozího komunikačního kanálu otevřeného funkcí *otevri_send ()*.

usb_send ()

```
static void usb_send (char usb_data){
  while(TXE != 0){
            }
            PORTE = usb_data;
            WR = 0;
            Delay1TCY ();                    //delay 100ns
            WR = 1;
}
```

Funkce slouží k odeslání jednoho bytu do převodníku USB/FIFO a následně pak přes USB do PC. Tato funkce je využita v další funkci.

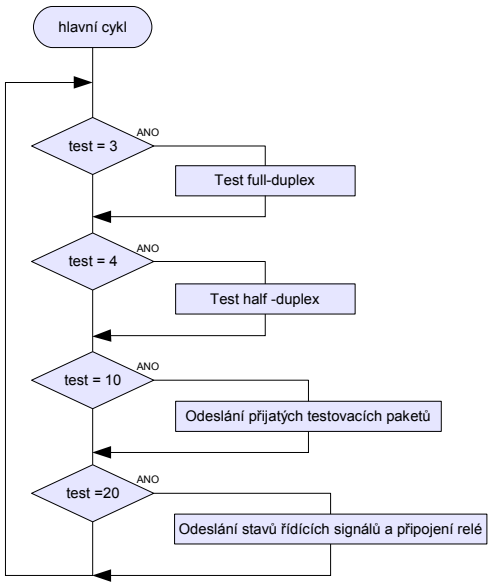
usbdata_send ()

```
static void usbdata_send (char *usb_datastream, char pocetbytu, char cil){
unsigned char pocetOUT = 0;
  usb_send(0xFC);
  usb_send(cil);
  for(; pocetOUT <= pocetbytu; pocetOUT++){
            usb_send(usb_datastream[pocetOUT]);
  }
}
```

Tato funkce odesílá obsah datového pole s určitým významem dat. Je založena na cyklu který postupně vybírá byty z pole a odesílá pomocí funkce *usb_send ()*.

3.4.5. Obsluha hlavního cyklu

Po inicializaci se hlavní program zacyklí do nekonečné smyčky a testuje proměnnou *test*, zda její hodnota odpovídá některé z podmínek(viz. obr. 11). Pokud odpovídá provede se daný podprogram. Proměnná *test* se nastavuje v přerušení EUSARTu nebo externího při příchodu instrukce.

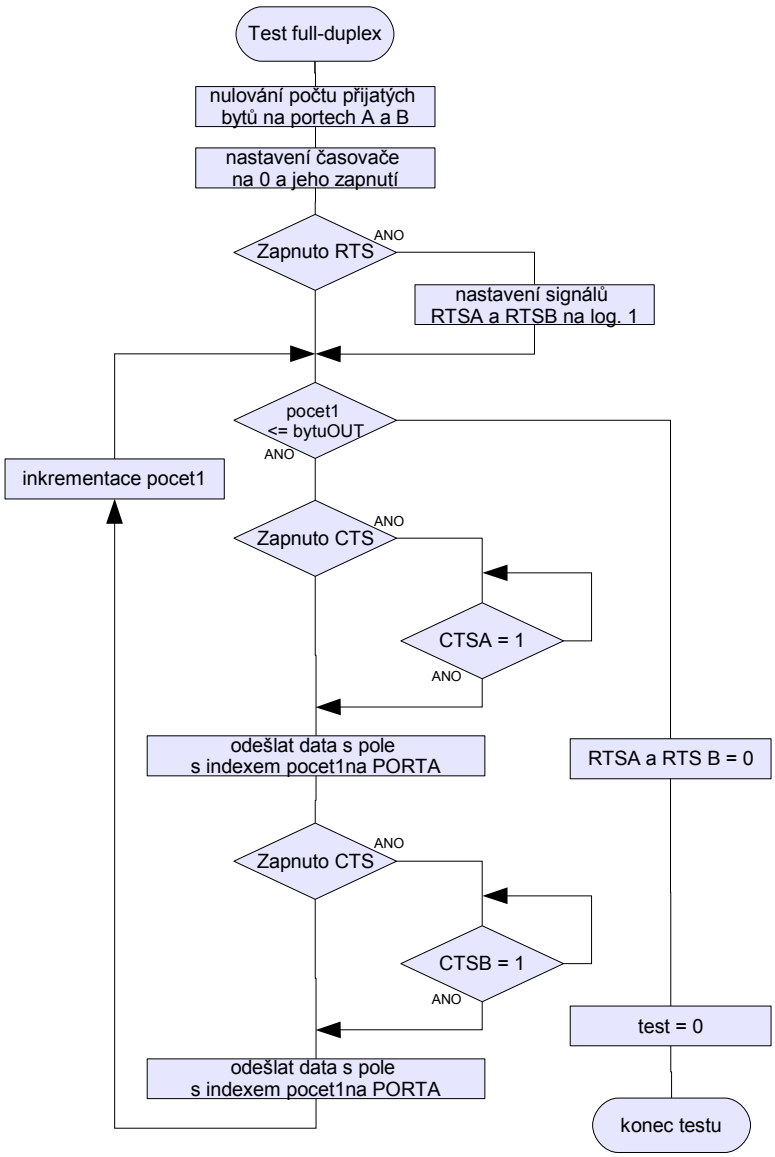


Obrázek 11: Vývojový diagram hlavího cyklu

Test full-duplex

Při spuštění full-duplexního testu se provádí proces podle algoritmu na obrázku 12.

Po vstupu do testu se vynulují registry čítající počet přijatých bytů na portech A a B, hodnota v časovači se nastaví na nulu. Pokud je v testu zvolena možnost testování s HW řízením RTS nastaví se jednotlivé signály RTS na obou portech na logickou 1. Proces pokračuje cyklem ve kterém se postupně odesílají jednotlivé byty z pole registrů[11] s indexem *pocet1* až do doby kdy se odešlou všechny. Zda se odeslaly všechny byty je určeno hodnotou v proměnné *bytu_OUT*. Před odesláním bytu na každý z portů se ještě testuje, zda je zapnuto HW řízení CTS a pokud ano, tak se čeká do doby než se na vysílacím portu na signálu CTS objeví logická 1.



Obrázek 12: Vývojový diagram full-duplexního testu

11 *z pole registrů testovacího paketu*

Test halfduplex

Princip half-duplexního testu je velmi podobný full-duplexnímu, s rozdílem, že se data neodesílají najednou, ale postupně. Nejprve se odešle celý testovací paket jedním směrem a po jeho přijetí se teprve začne vysílat druhým směrem, využívají se tedy 2 cykly pro vysílání. Změna směru toku dat je ještě doplněna o uživatelsky nastavitelnout prodlevu v milisekundách.

Odeslání přijatých testovacích paketů

Tento podprogram je využíván při posílání přijatých dat na portech A a B do PC k vyhodnocení. Při odeslání dat se odešlou také naměřené hodnoty časovačem, které musí být rozděleny z 16-bitového čísla na dvě 8-bitové a potom se mohou odeslat. Odeslání se provádí funkcemi ovedenými v kapitole 2.3.4. Následující kód zobrazuje posloupnost odesílaných dat a nakonec nuluje již odeslané pole registrů přijatých testovacích paketů z portů A a B.

```
TIMERl_C1 = (char) (TIMER_C1 & 255);
TIMERh_C1 = (char) (TIMER_C1>>8);
TIMERl_C2 = (char) (TIMER_C2 & 255);
TIMERh_C2 = (char) (TIMER_C2>>8);
otevri_send ();
usb_send(0xFC);
usb_send(0x12);
usb_send(bytuIN_usart1);
usb_send(bytuIN_usart2);
usb_send(TIMERl_C1 );
usb_send(TIMERh_C1);
usb_send(TIMERl_C2);
usb_send(TIMERh_C2);
usbdata_send(usart1IN, bytuIN_usart1, 0x13);
usbdata_send(usart2IN, bytuIN_usart2, 0x14);
zavri_send ();
test = 0;
for(x = 0; x <= 192; x++){
  usart1IN[x] = 0;
  usart2IN[x] = 0;
}
```

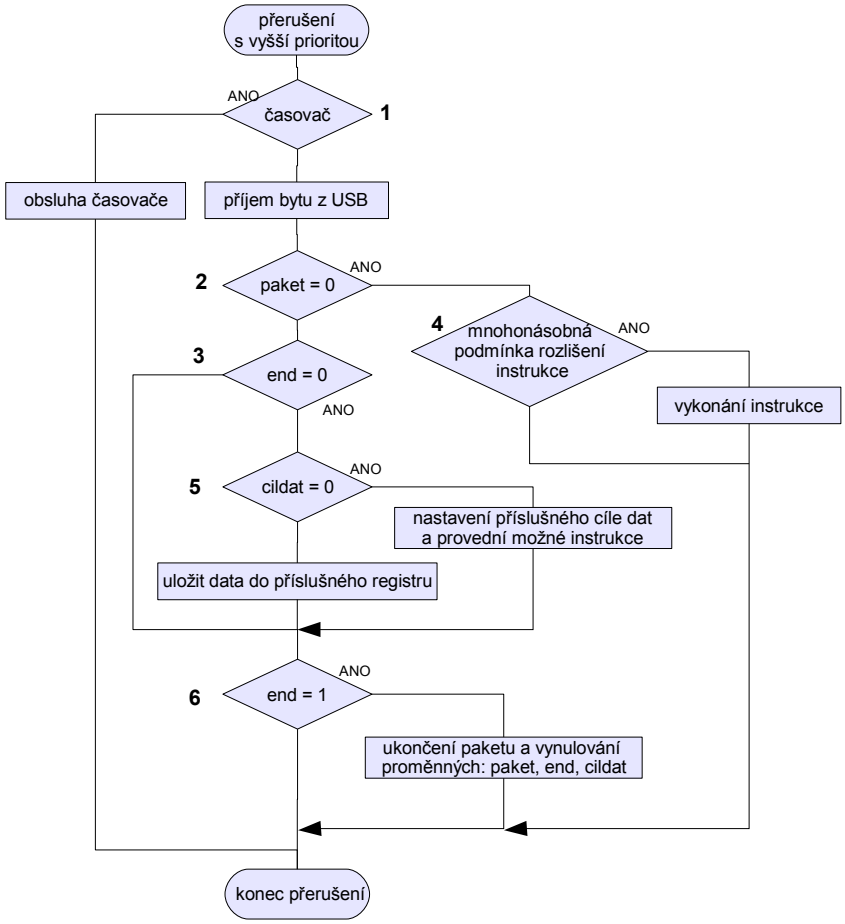
Odeslání stavů řídících signálů a připojení relé.

V tomto podprogramu se provede detekce všech řídících signálů na portech a stavy připojení relátek. Získané údaje se zapíšou do 3 registrů, které jsou potom pomocí výše zmíněných funkcí odeslány do PC. Následující kód ukazuje získání tří bytů z více vstupně/výstupních bran pomocí logických součtů a součinů

```
stat_rele = ((PORTA & 0b00001111) | ((PORTF & 0b00000011)<<4));
stat_1 = ((((PORTB & 0b00001110)>>1) | ((PORTC & 0b11000000)>>3)) | ((PORTD & 0b11000000)>>1));
stat_2 = (((PORTG & 0b00011110)>>1) | ((PORTF & 0b00011100)<<2));
```

3.4.6. Obsluha přerušení s vyšší prioritou

V obsluze přerušení s vyšší prioritou se rozlišují dvě přerušení. První se detekuje zda nedošlo k přerušení časovačem a pokud nedošlo, tak došlo k externímu přerušení příchodem dat do vyrovnávací paměti převodníku USB/FIFO. Na obrázku 13 je zjednodušený vývojový diagram obsluhy přerušení popsaný níže.



Obrázek 13: Zjednodušený vývojový diagram obsluhy přerušení s vyšší prioritou

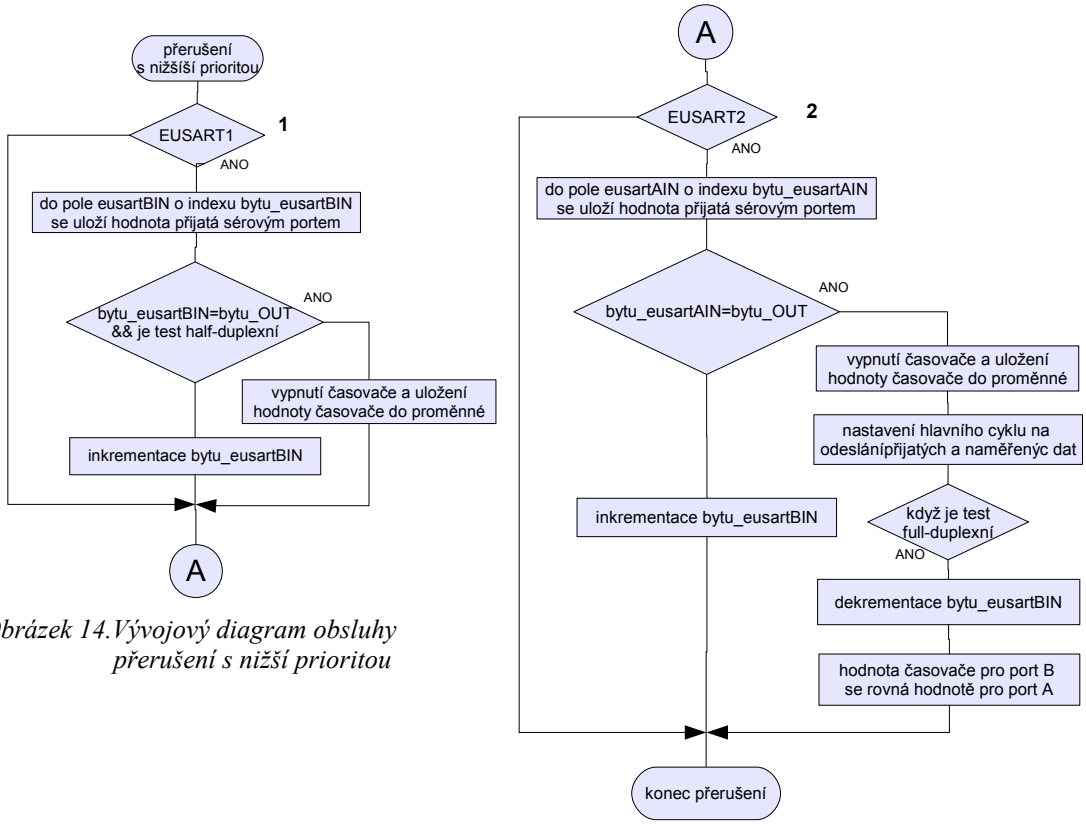
Při přerušení vyvolaném časovačem se na podmínce 1 přesune program na obsluhu časovače. V této části se vynulují všechny registry čítající počet přijatých bytů na sériových portech A a B a zastaví se testování. Dojde také k odeslání informace do PC, že testovací paket nebyl doručen.

V případě externího přerušení od USB podmínka 1 není splněna a pokračuje se obsluhou příjmu bytu z USB. Po přijetí bytu následuje rozlišení bytu. Pokud je splněna podmínka 2 znamená to, že se jedná o instrukční byte a ne o datový a obsluha je přesunuta na mnohonásobnou podmínku 4, ve které dochází k rozlišení instrukce a následně k jejímu provedení. V případě, že přišla instrukce FC, což podle protokolu znamená hlavička datového paketu, nastaví se proměnná *paket* na 1 a ukončí se přerušení. Při příjmu dalšího bytu už mikrokontroler ví, že se jedná o data a na podmínce číslo 2 přepne obsluhu na test

podmínky 3, která platí, protože end je ve výchozím stavu v 0. Obsluha pokračuje na podmínku č.5, která zjišťuje zda se jedná o 2. byte v komunikačním paketu CIL DAT, pokud ano, tak se nastaví proměná *cildat* hodnotou ve 2. bytu komunikačního paketu a při třetím přijatém bytu, už algoritmus ví, že se jedná o data a díky proměnné *cildat* ví, kam je má uložit. Poslední krok je opakován do doby než se nastaví proměnná *end* na 1 a v podmínce 6 nebo 3 po načtení všech dat dojde k ukončení paketu vynulováním proměnných *paket*, *cildat* a *end.*

3.4.7. Obsluha přerušení s nižší prioritou

K přerušení s nižší prioritou dochází pouze při příjmu bytu na jeden z EUSART portů. Obsluha je řešena pod algoritmu na obrázku 14.



Obrázek 14.Vývojový diagram obsluhy přerušení s nižší prioritou

Při vyvolání přerušení se testuje, zda bylo vyvoláno sériovým portem 1(podmínka 1 ve vývojovém diagramu) nebo portem 2(podmínka 2). V případě že dojde k přerušení portem 1, skočí běh programu do obsluhy tohoto přerušení, kde se uládají přijaté byty do pole registrů, inkrementuje se index pole a testuje se, zda byli přijaty všechny odeslané byty. Po přijetí všech bytů se zastavuje časovač a jeho hodnota se uloží do proměnné, která je po provedení celého testu odeslána do PC k vyhodnocení. Je možné, že během obsluhy přerušení z prvního portu dojde k příjmu bytu na druhém portu, ale toto přerušení není vyvoláno. K zajištění vyýběru dat z přijímacího registru, dojde po ukončení obsluhy portu 1 stejným způsobem.

3.4.8. Upload firmwaru a debugging

Programátorů pro řadu PIC18 neexistuje na internetu mnoho. Většinou je potřeba využít komerční programátory, hledal jsem proto nejlevnější řešení nahrání firmwaru do paměti mikrokontroléru. Narazil jsem přitom na stránkách www.cadsoft.de v sekci projekty na zjednodušený návrh programátoru a debuggeru Microchip ICD 2[12]. Sestavil jsem podle návodu a nahrál do ovládacího mikrokontroléru potřebný firmware. Postavené zařízení nahrazuje komerční verzi ICD 2. Fotografie programátoru a debuggeru najdete v příloze 4.

Tento nástroj byl velmi cenným až postradatelným pomocníkem. Nejen, že dokáže naprogramovat použitý mikrokontrolér, ale jak z jeho názvu vyplývá, je možné s ním provádět ladění aplikace. Tato funkce umožňuje zastavení běhu programu na určeném místě a nahlédnutí do obsahu paměti mikrokontroléru.

Pro využití ICD 2 jako ladícího nástroje je nutno ponechat v paměti mikroprocesoru volných 512kB paměti RAM a vývody určené k programování nechat pouze pro připojení k ICD 2. Vývody jsou zobrazeny na obrázku 5 v kapitole 2.2.3.

3.5. Software

Důležitým prvkem testeru je ovládací aplikace. Zajišťuje nastavení, vyhodnocení testu a ovládání celého testeru. Aplikaci jsem vytvořil v prostředí C++ Bilder 6.

3.5.1. Komponenta AdpComPort

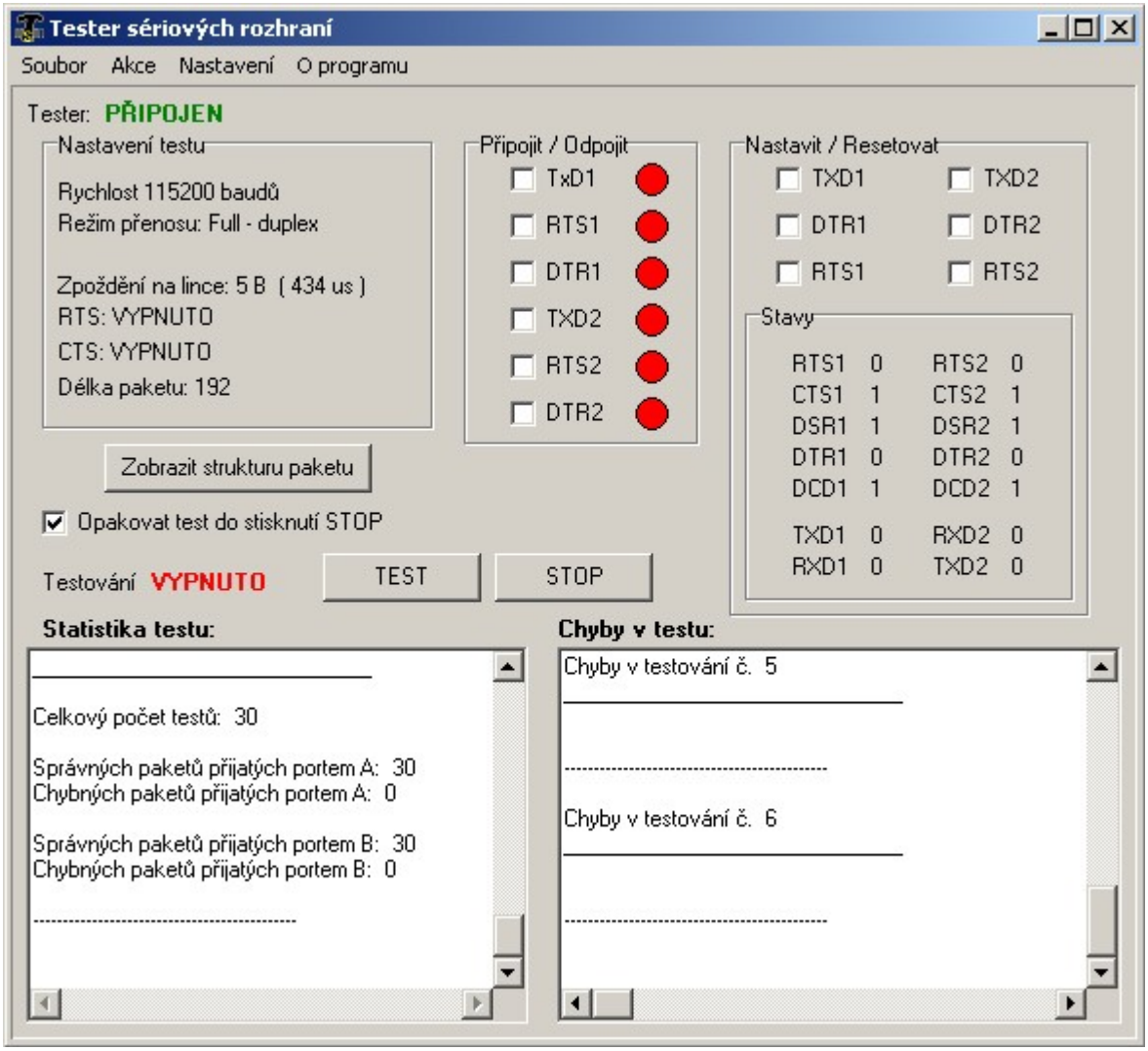
Pro komunikaci softwaru s hardwarem přes USB bylo třeba zajistit přístup softwaru k virtuálnímu sériovému portu. Využil jsem k tomu komponentu AdpComPort se souboru komponent APRO stažených ze serveru www.sourceforge.net. Touto komponentou se značně zjednodušil přístup k sériovému portu, jeho inicializace a obsluha.

Při volání této komponenty a otevírání sériového portu je nutností zvolit číslo virtuálního sériového portu vytvořeného testerem.

12 *ICD 2 – In Circuit Debbuger 2*

3.5.2. Koncepce softwaru

Po spuštění ovládacího programu se otevře základní okno aplilace(obrázek 15).



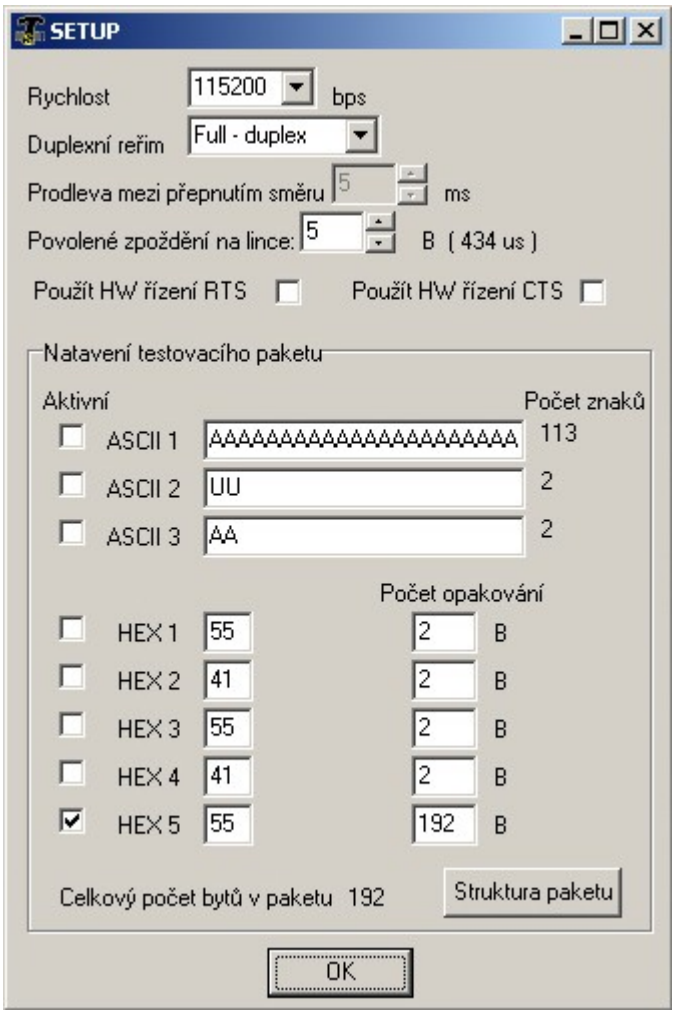
Obrázek 15.: Základní okno ovládací aplikace

Výchozí okno zobrazuje aktuální nastavení testu a tlačítko pod ním umožňuje zobrazit strukturu testovacího paketu, panel s možností odpojit a připojit výstupní vývody, další blok umožňuje nastavit výstupní signály na logickou jedničku nebo nulu a zobrazit stavy na všech vývodech portu. Dalšímy prvky jsou tlačítka pro ovládání testování a výstupní panely pro zobrazení statisky a chyb testu.

Při pohledu do horní části aplikace nalezneme roletové menu *Soubor,* kde můžeme uložit statisku a chyby v testu, menu *Akce* pro přpojení[13] a odpojení hardwaru, pro automatické načítání stavu, menu Nastavení, kdy po otevření okna Nastavení testu(obrázek 16)

13 *Připojení hardwaru je nutno provést před každým testováním a jakoukoliv prací se softwarem*

můžeme zvolit testovací paket a nastavení testu a nakonec menu O porgramu, které zobrazí informace o programu.



Obrázek 16.: Okno natavení testu

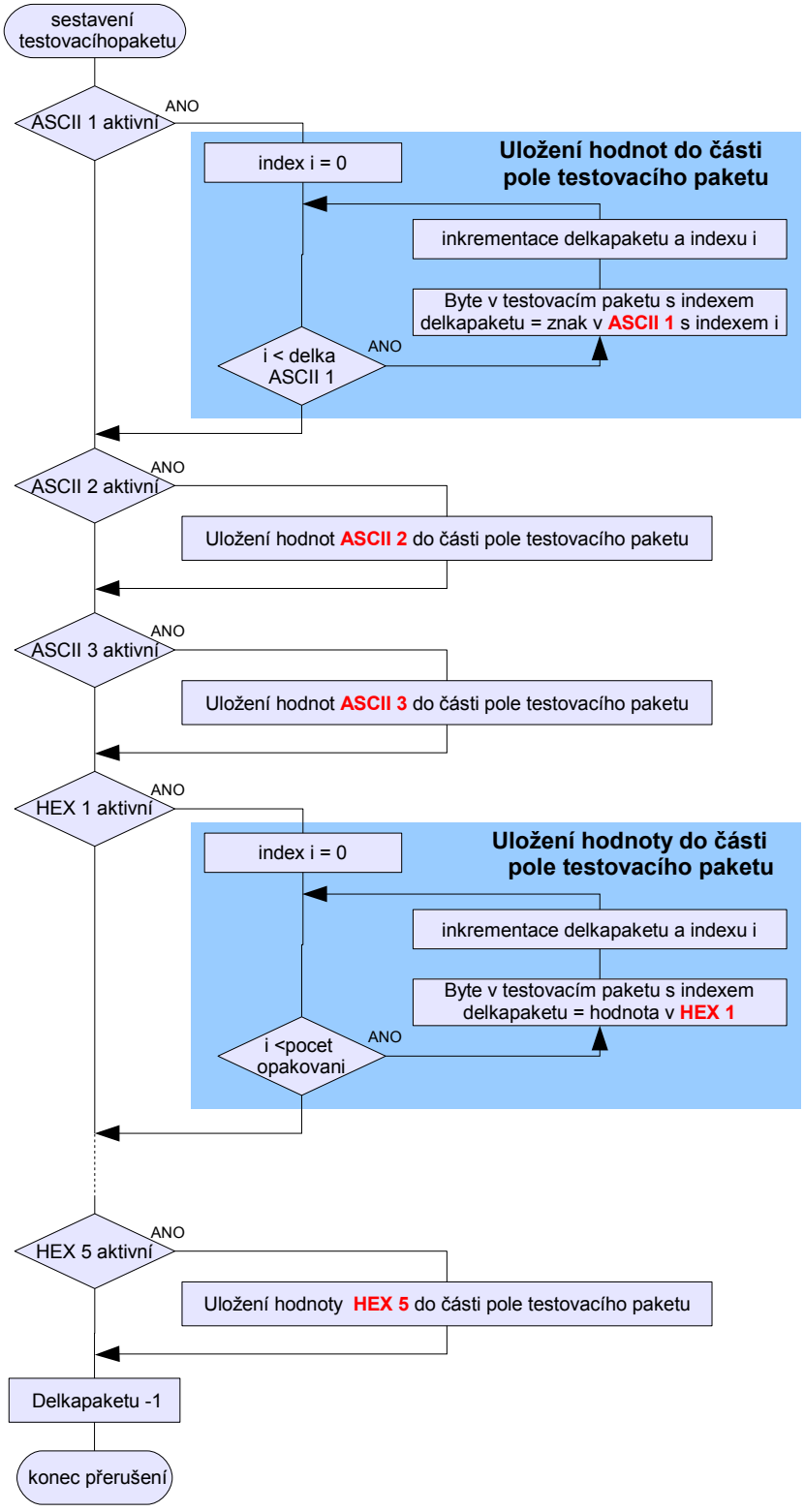
V natavení testu si zvolíme rychlost testu, režim přenosu, prodlevu při přepínání směru v half-duplexním reřimu, povolené zpoždění na lince a zapnutí hardwarového řízení RTS a CTS. V panelu *Nastavení testovacího paketu* můžeme sestavit libovolný testovací paket.

3.5.3. Vytvoření testovacího paketu

Sestavení testovacího paketu z nastavení probíhá podle následujícího algoritmu.

Program postupně zjišťuje, zda je dané vstupní pole pro sestavení paketu aktivované a pokud ano uloží hodnoty nastavené v poli do testovacího paketu. Princi pvysvětlím na příkladu, kdy bude požívané pole ASCII 1 a HEX 1.

Prní podmínka testuje zda je aktovované pole ASCII 1, podmínka je splněna a provede se zapis hodnot v poli do testovacího paketu. Děje se tak pomocí ckylu který postupně ukládá jednotlivé znaky z ASCII 1 do pole testovacího paketu do doby než jsou všechny zapsané znaky uloženy. Pro indexaci vnitřního cyklu se používá proměnná, která se při každéme zápisu jednoho ze vstupních polí nuluje. Pro indexaci testovacího paketu je použita proměnná *delkapaketu* která se nenuluje celý podprogram a na konci určuje délku testovacího paketu. Tím docílíme při splnění další podmínky HEX 1 jako aktivní, že další hodnoty se budou v poli testovacího paketu přidávat za již vložené z předchozího vstupního pole. Hodnoty hex se do testovacího paketu přidávají také cyklem, který je dán počtem

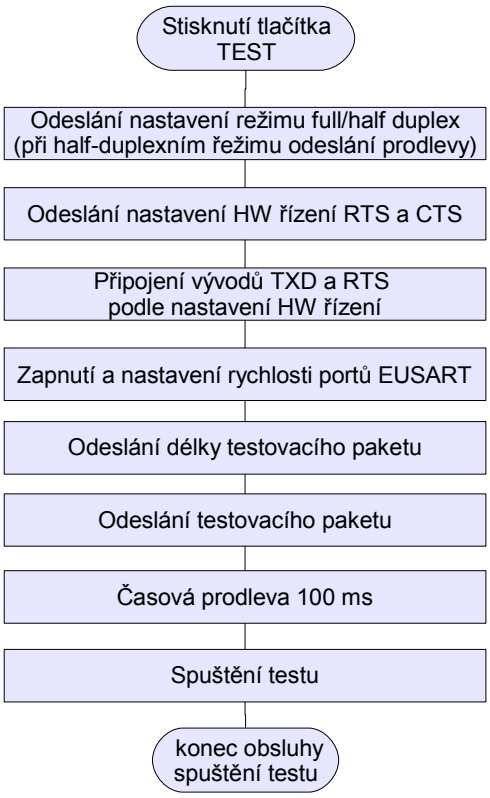


Obrázek 17: Vývojový diagram sestavení testovacího paketu

opakování hodnoty zadané v nastavení HEX 1(viz obrázek 16 kaitola 2.4.2). Vnitřni cyklus se používá u všech vstupních polí ASCII 1 až ASCII 3 a HEX 1 až HEX 5 stejný. Na konci celého podporgramu se od proměnné délka paketu odečte hodnota 1, protože pozdější využití této proměnné počítá i s hodnotou 0.

3.5.4. Spuštění testu

Při stisknutí tlačítka TEST(viz obrázek15 kapitola 2.4.2) se provedou příkazy podle následující posloupnosti:



Obrázek 18: Obsluha spuštění testu

Při odesílání všech testovacích informací se využívají komunikační pakety definované v tabulce 2, kapitole 2.1.2.

Zaškrnutím volby *Opakovat test do stisknutí tlačítka STOP* definujeme, že po vyhodnocení přijatých dat se má znovu odeslat instrukce pro spuštění testu a nová testovací data se do testeru už neposílají. Při stisknutí tlačítka STOP se dokončí prováděný test a další už se neprovádí. Při jakémkoliv ukončení testu, ať z důvodů špatného spojení mezi portem A a B, nebo stisknutím tlačítka stop, nebo při vyplé volbě opakování testu, se vypíše statistika testování.

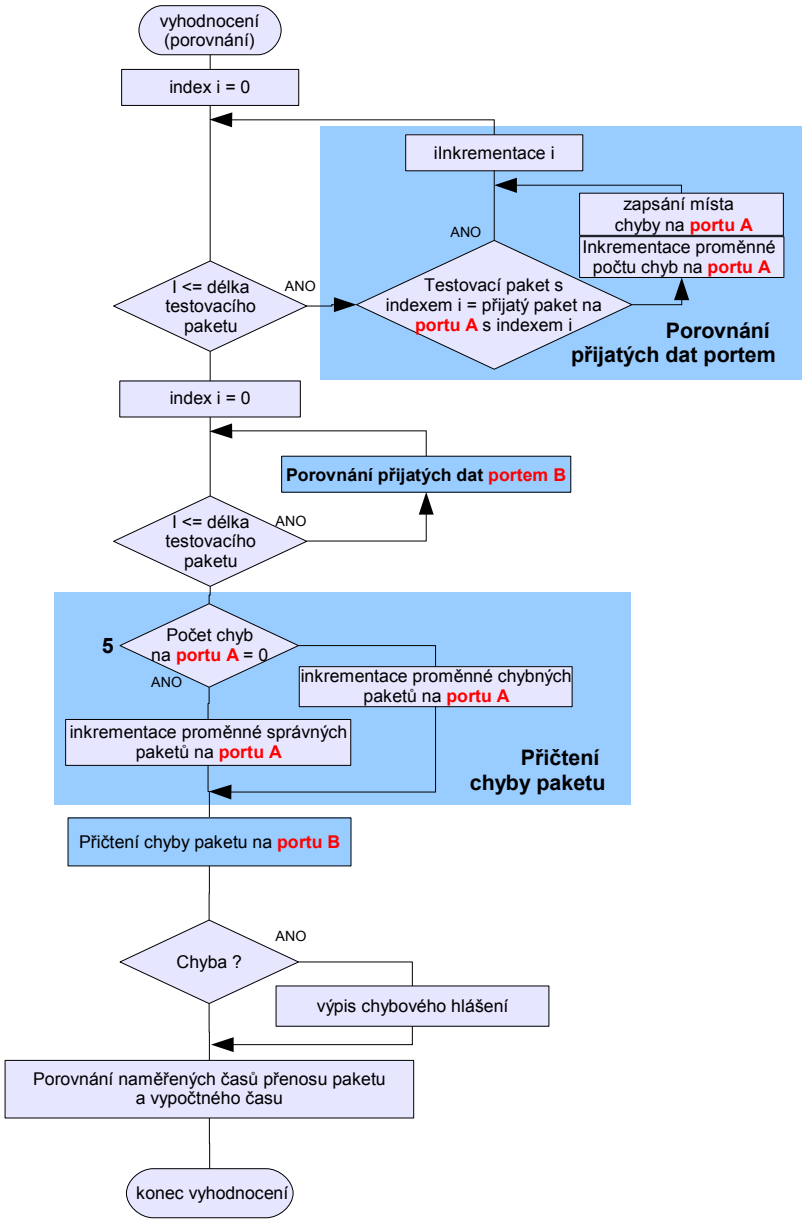
3.5.5. Příjem dat z testeru

Při příchodu bytu nebo sekvence bytu na virtuální sériový port se ukládají byty do vyrovnávací paměti a po přijetí celistvého paketu dat dojde k vyvolání události příchodu dat na sériový port. Vybíraní bytů z paměti se potom realizuje cyklem dokud není paměť prázdná. Uvnitř cyklu je rozlišení typu dat. Rpzlišení probíhá stejným způsobem jako ve firmwaru testeru popsaném v kapitole 2.3.6.

3.5.6. Porovnání dat

K porovnávání přijatých dat dochází po příjmu posledního přijatého bytu naměřených dat z testeru. Dějej se tak podle algoritmu na obrázku 19.

Oba přijaté testovací pakety jsou cykly porovnávány s odeslaným testovacím paketem, v případě chyby v bytu se inkrementuje proměnná s počtem chybových bytů příslušného portu a zapíše se do paměti číslo chybového bytu v paketu. Po porovnání obou přijatých paketů se testuje zda byl nějaký z paketů chybný a pokud ano, inkrementuje se proměnná při zaznamenání počtu chybných paketů daného portu. Takto se provede pro port A i B a testuje se zda došlo vůbec k nějaké chybě. Pokud došlo k chybě na jednom ze dvou protů nebo na obou vypíše se výpis chybovíh hlášení do bloku Chyby v testu (viz. Obrázek 13 kapitola 2.4.2).



Obrázek 19: Vývojový diagram vyhodnocení výsledků

Po vyhodnocení se vynulují pole přijatých testovacích paketů a potom testování zda se má test opakovat nebo ne. Pokud se má test opakovat, vyšle program instrukci pro spuštění testu a vše výše uvedené se opakuje dokud

se nestiskne tlačítko STOP. Pokud má být testování ukončeno vypíše se statiska testování, odpojí se všechny výstupní vývody a načte se stav testeru.

3.5.7. Popis statistiky a chybového hlášení

Ve statistice testu se vypisují následující údaje takto formátované:

Číslo testování: 1

Celkový počet testů: 1

Správných paketů přijatých portem A: 1
Chybných paketů přijatých portem A: 0

Správných paketů přijatých portem B: 0
Chybných paketů přijatých portem B: 1

Číslo testování zobrazuje počet testů spuštěných tlačítkem TEST od spuštění programu. Položka celkový počet testů ukazuje kolikrát bylo provedeno odeslání a přijetí testovacího paketu. Zbylé 4 položky ukazujou špatné a dobré pakety přijaté na jednotlivých portech.

Chybové hlášení se vypisuje pouze pokud nastane chyba při vyhodnocení popsaném v předchozí kapitole. Struktura chybového hlášení je následující:

Chyby v testování č. 1

Test číslo: 1

Počet chyb v paketu přijatých portem A: 0
Chyby na znaku/cích:

Počet chyb v paketu přijatých portem B: 1
Chyby na znaku/cích: 1,

Testovací paket: 55 55 55 55
Přijatý paket na portu A: 55 55 55 55
Přijatý paket na portu B: FF 55 55 55

Položka Chyby v testování č. X říká při jakém spuštění testování došlo k chybě nebo chybám. Test číslo X určuje při kolikátém testu došlo k chybě. Následující řádky potom zobrazují počet chybných bytů v jakém čísle bytu byla chyba. Pro porovnání jsou zobrazeny testovací paket a pakety přijaté na jednotlivých portech pod sebou. Hodnota je na tomto výpisu zobrazena hexadecimálně.

4. Výsledky řešení

Dle výše uvedeného návrhu splňuje práce zadání specifikovaná v úvodu. Umožňuje jednoduchým softwarovým rozhraním ovládat tester. Připojovat a odpojovat jednotlivě výstupní vývody obou RS-232 portů a nastavovat jejich hodnoty. Dále umožňuje uživatelsky nastavit testovací paket, režim testování, rychlost testování, umožňuje měnit časový limit pro přenos testovacího paketu, umožňuje zapnout nebo vypnout hardwarové řízení RTS a CTS. Aplikace je schopna podat důležité informace o chybách a statistiku celého testování.

Deska plošného spoje byla navržena tak, aby mohla být namontována do krabičky používané firmou ELO+, s.r.o. pro převodníky metalických sériový linek na optické.

Základní technické parametry zařízení:

Napájecí napětí	stabilizovaných 5 V
Napájecí proud	280 mA
Rozhraní pro propojení s PC	1 x USB 2.0
Testovacích portů	2x RS-232
Maximální testovací rychlost	230 400 baud
Rozměry	110 x 80 x 25 mm

Tabulka 5: Technické parametry testeru sériových rozhraní

Použité přístroje

Digitální osciloskop EZ Digital DS 1150
True RMS multimeter FLUKE 110
PC

Použitý software

Borlad C++ Builder 6
Microchip Technology Inc. MPLAB® IDE Integrated Development Environment v7.5
Microchip Technology Inc. MPLAB® C18 Compiler v3.1 Student Edition
OpenOffice.org 2.0

5. Závěr

Návrh testeru uvedený v této práci je s malou úpravou schopný testovat rozhraní RS-422 a RS-485 na rychlostech vyšších než 230 400 baudů. Předpokládá se pokračování ve vývoji těchto dalších modifikací.

Díky této práci jsem získal zkušenosti s mikrokontrolérem PIC18 a programováním v jazyku C18. Upevnil a rozšířil jsem si znalosti programování a práci v prostředí C++ Builder 6, vyzkoušel jsem si práci s technologií SMT, návrh oboustranného prokoveného plošného spoje s nepájivou maskou a tvorbu nových knihoven součástek v prostředí Eagle. Získal jsem zkušenosti s návrhem rozhraní USB pomocí virtuálního komunikačního sériového portu.

Seznam obrázků

Seznam tabulek

Seznam příloh

Použitá literatura

Microchip Technology Inc.. *PIC18F8722 Family Data Sheet* [online]. 2004 [cit. 2007-03-18]. Dostupný z WWW: <http://ww1.microchip.com/downloads/en/DeviceDoc/39646b.pdf>.

Future Technology Devices Intl. Ltd.. *FT245BM USB FIFO (USB - Parallel) I.C.* [online]. 2005 [cit. 2007-03-18]. Dostupný z WWW: <http://www.ftdichip.com/Documents/DataSheets/DS_FT245BM.pdf>.

Maxim Integrated Products. *+5V-Powered, Multichannel RS-232* [online]. 2006 [cit. 2007-03-18]. 19-4323. Rev 15. Dostupný z WWW: <http://datasheets.maxim-ic.com/en/ds/MAX220-MAX249.pdf>.

HW server. *RS232* [online]. 2003 [cit. 2003-03-18]. Dostupný z WWW: <http://rs232.hw.cz/index.html>.

GOOK, Michael. *Hardwarová rozhraní Průvodce programátora*. Mikulaštík Jakub. 1. vyd. Brno : Computer Press, 2006. 457 s. ISBN 80-251-1019-2.

HEROUT, Pavel. *Učebnice jazyka C*. 3. upr. vyd. České Budějovice : Kopp, 1997. 266 s. ISBN 80-85828-21-9.

Microchip Technology Inc.. *MPLAB® C18 C COMPILER GETTING STARTED* [online]. 2005 [cit. 2007-03-18]. DS51295F. Dostupný z WWW: <http://ww1.microchip.com/downloads/en/DeviceDoc/MPLAB_C18_Getting_Started_51295f.pdf>.

Microchip Technology Inc.. *MPLAB® C18 C COMPILER USER'S GUIDE* [online]. 2005 [cit. 2007-03-18]. DS51288J. Dostupný z WWW: <http://ww1.microchip.com/downloads/en/DeviceDoc/MPLAB_C18_Users_Guide_51288j.pdf>.

Microchip Technology Inc.. *MPLAB® C18 C COMPILER LIBRARIES* [online]. 2005 [cit. 2007-03-01]. DS51297F. Dostupný z WWW: <http://ww1.microchip.com/downloads/en/DeviceDoc/MPLAB_C18_Libraries_51297f.pdf>.