VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

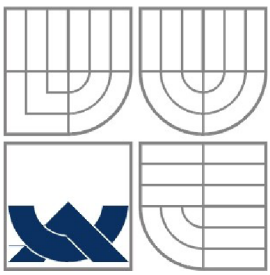
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

NÁVRH A IMPLEMENTACE SOFTWAROVÉHO UART PRO MIKROKONTROLÉRY RODINY HC08

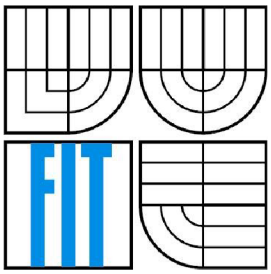
BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — PAVEL KUČERA
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

NÁVRH A IMPLEMENTACE SOFTWAROVÉHO UART PRO MIKROKONTROLÉRY RODINY HC08

DESIGN AND IMPLEMENTATION OF A SOFTWARE UART FOR MICROCONTROLLERS FROM HC08 FAMILY

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

PAVEL KUČERA

VEDOUCÍ PRÁCE
SUPERVISOR

ING. JOSEF STRNADEL, PH.D.

BRNO 2008

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2007/2008

Zadání bakalářské práce

Řešitel: **Kučera Pavel**
Obor: Informační technologie
Téma: **Návrh a implementace softwarového UART pro mikrokontroléry rodiny HC08**
Kategorie: Vestavěné systémy

Pokyny:

1. Seznamte se s mechanismy komunikace po sériové lince, zejména pak s mechanismy asynchronní sériové komunikace a principem činnosti a programováním modulů SCI (realizovaných pomocí obvodů UART) mikrokontrolérů rodiny HC08.
2. Diskutujte výhody a nevýhody možných implementací softwarového UART (sUART). Navrhněte sUART zejména s ohledem na maximální přenositelnost mezi členy rodiny HC08, s ohledem na maximální konfigurační kompatibilitu s rozhraním SCI u HC08 a s ohledem na možnost změny parametrů komunikace bez nutnosti přeprogramování mikrokontroléru.
3. Na základě závěrů učiněných v předchozím bodě a po dohodě s vedoucím sUART implementujte např. na verzích QT/QY mikrokontrolérů rodiny HC08 a prakticky ověřte jeho funkčnost jak při komunikaci s vybraným mikrokontrolérem majícím zabudováno rozhraní SCI, tak s PC přes rozhraní RS-232.
4. Shrňte dosažené výsledky.

Literatura:

- *Freescale Semiconductor* [online]. c2007. http://www.freescale.cz/.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 a 2.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Strnadel Josef, Ing., Ph.D.**, UPSY FIT VUT
Datum zadání: 1. listopadu 2007
Datum odevzdání: 14. května 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

doc. Ing. Zdeněk Kotásek, CSc.
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. Pan

Jméno a příjmení: **Pavel Kučera**
Id studenta: 79328
Bytem: Mezi Zahradami 138, 530 09 Pardubice
Narozen: 18. 09. 1985, Pardubice
(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
bakalářská práce

Název VŠKP: Návrh a implementace softwarového UART pro mikrokontroléry rodiny HC08
Vedoucí/školitel VŠKP: Strnadel Josef, Ing., Ph.D.
Ústav: Ústav počítačových systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:

tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☒ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

.....	.....
Nabyvatel	Autor

Abstrakt

Práce se zabývá problematikou sériové komunikace, zejména pak asynchronní sériové komunikace u mikrokontrolérů rodiny HC08. Obsahuje analýzu modulu SCI, zajišťující asynchronní sériovou komunikaci mikrokontrolérů. Na základě analýzy bylo navrhnuto softwarové řešení UART (sUART) pro řady mikrokontrolérů z rodiny HC08, které modul SCI nemají. Rozhraní sUART bylo úspěšně testováno komunikací s počítačem pomocí rozrhaní RS-232 i komunikací mezi mikrokontroléry navzájem.

Klíčová slova

HC08, komunikace, mikrokontrolér, RS-232, SCI, software, UART.

Abstract

Bachelor's thesis deals with problematics of serial communication, specially of serial asynchronous communication of microcontrollers from HC08 family. Thesis includes analysis of SCI module, which provides serial asynchronous communication of microcontrollers. On a basis of the analysis the software solution of UART (sUART) for microcontrollers from HC08 family, which have no SCI module, was created. SUART interface was successfully tested by communication with a computer using RS-232 interface as well as by communication between several microcontrollers.

Keywords

HC08, communication, microcontroller, RS-232, SCI, software, UART.

Citace

Pavel Kučera: Návrh a implementace softwarového UART pro mikrokontroléry rodiny HC08, bakalářská práce, Brno, FIT VUT v Brně, 2008

Návrh a implementace softwarového UART pro mikrokontroléry rodiny HC08

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Josefa Strnadela, Ph.D. Další informace mi poskytl bc. Jiří Blecha. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jméno Příjmení
Datum

Poděkování

Rád bych poděkoval Ing. Josefu Strnadelovi, Ph.D. za poskytnuté konzultace při vytváření BP a za zapůjčení vývojového kitu JANUS. Dále bc. Jiřímu Blechovi za obecné rady v oblasti problematiky programování mikrokontrolérů a za poskytnutí části informačních zdrojů ([4], [5]).

Obsah

1 Úvod

Obsahem mé bakalářské práce s názvem Návrh a implementace softwarového UART pro mikrokontroléry rodiny HC08 je vytvoření funkčního seriového rozhraní u mikrokontrolérů rodiny HC08 řady NITRON, které sériové rozhraní jako svoji periferii nemají. Přestože mikrokontroléry NITRON sériové rozhraní nemají, lze jej poměrně úspěšně programově simulovat. Základem je analýza principů sériové komunikace, zejména pak principu asynchronní komunikace, toto bude obsahem 2. kapitoly. Jedním z hlavních požadavků na vytvářený softwarový UART (dále jen sUART) je největší možná kompatibilita s periferií SCI u mikrokontrolérů HC08, která obstarává asynchronní sériové rozhraní těchto mikrokontrolérů. Principy činnosti modulu SCI se zabývá 3. kapitola. Ve 4. kapitole se budu zabývat základními principy, kterými je možno problematiku sUART řešit včetně popisu výhod a nevýhod jednotlivých řešení. Provedu analýzu současného stavu problematiky softwarového sériového rozhraní, protože některá řešení již byla vytvořena. Dostupná řešení budu komentovat z hlediska jejich výhod a nevýhod. Na základě požadavků na rozhraní a analýzy současného stavu problematiky navrhnu způsob vlastního řešení. Pátá kapitola pojednává o zvoleném řešení, zabývá se problémy implementace sUART a detailně popisuje ovládání rozhraní. V závěrečné (6.) kapitole diskutuji dosažené výsledky, vlastní přínos do problematiky včetně dalšího možného vývoje.

Pro řešení bakalářské práce mám k dispozici vývojový kit JANUS s mikrokontroléry HC08 řady QT (pouzdro DIP8) a QY (pouzdro DIP16), vývojové prostředí CodeWarrior pro HC08 a nepájivé kontaktní pole s dalšími mikrokontroléry a nezbytnými podpůrnými obvody. Pro testování výsledného řešení budu používat propojení mikrokontrolérů se sUART s mikrokontroléry se zabudovaným modulem SCI a s počítačem vybaveným sériovým rozhraním RS-232.

2 Sériová komunikace

Základní myšlenkou sériové komunikace je co nejnižší počet vodičů v rozhraní. Toho je dosaženo postupným přenosem jednotlivých bitů po jednom datovém vodiči v jednom směru v časovém sledu. Existují dva základní přístupy komunikace po sériové lince - přenos synchronní a asynchronní.

2.1 Synchronní přenos

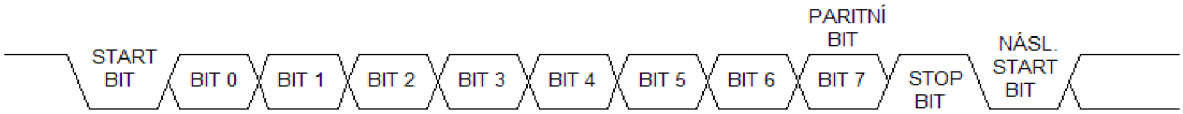
Spolu s daty je přenášen i hodinový signál, pro který je v rozhraní vyčleněn jeden samostatný vodič. Synchronního přenosu se účastní zařízení dvou typů, typu slave (též označovaný jako receiver neboli přijímač) a typu master (též označovaný jako driver). V jedné soustavě může být vždy pouze jeden prvek typu master, který může být připojen k maximálně deseti prvkům typu slave. Master do soustavy generuje hodinový signál, ten je zpravidla periodický obdélníkového typu, nicméně jeho periodicita není nezbytně nutná. Hrany hodinového signálu určují buď okamžik, kdy je na datový vodič vkládán vysílačem další bit přenášené informace, nebo okamžik, kdy je možné informaci z datového vodiče přijímačem (přijímači) bezpečně sejmout.

Výhodou rozhraní je snadná realizace bez velkých nároků na časování díky jednotnému zdroji hodinového signálu, kterému se všechny ostatní prvky v soustavě podřídí. Maximální přenosové rychlosti jsou 10Mbit/s při délce spoje do 12 metrů resp. 100kbit/s při délce spoje 1200m, což je maximální definovaná délka spoje. Nevýhodou je potřeba většího počtu vodičů - konkrétně dva vodiče pro poloviční duplex (přenos dat pouze jedním směrem) resp. tři vodiče pro plný duplex (přenos dat oběma směry)

2.2 Asynchronní přenos

U asynchronního přenosu není hodinový signál přenášen, přijímač si jej proto musí s dostatečnou přesností generovat sám. Přijímač musí být vybaven prostředky, díky kterým dojde k sesynchronizování hodinového signálu přijámače s generátorem hodin vysílače. Z výše uvedeného vyplývá, že je třeba, aby byl jak vysílač, tak přijímač vybaven poměrně přesným generátorem hodinového signálu. Je nezbytné, aby generátor hodinového signálu vysílače i přijímače běžel na stejné (předem dohodnuté v rámci jedné aplikace) frekvenci. K sesynchronizování hodinových signálů dochází pomocí detekce předem dohodnuté změny úrovně na datovém vodiči přijímačem. K tomu je v rámci každého přenášeného rámce vyhrazen časový interval o délce trvání přenosu jednoho bitu, tato dohodnutá změna úrovně nenese žádnou informaci, slouží pouze k synchronizaci hodin. Počáteční změna úrovně je nazývána start bit. Po přenesení úvodního start bitu proběhne

přenos datových bitů od nejméně významného (LSB) po nejvíce významný (MSB) bit. Přenos datových bitů probíhá jednoduše tak, že vysílač přepne datovou linku do úrovně odpovídající logické 0 nebo 1 na dobu přenosu jednoho bitu danou zvolenou přenosovou rychlostí. Přijímač uprostřed tohoto intervalu (čas odpočítávají sesynchronizované hodiny) sejme úroveň na datovém vodiči. Po přenesení datových bitů vysílač přepne datovou linku do výchozí klidové úrovně na dobu jednoho až dvou bitových intervalů, toto přepnutí do klidové úrovně nazýváme stop bit(y). Mezi datovými bity a stop bitem/bity může být ještě vysílačem vložen kontrolní paritní bit. Parita může být buď lichá nebo sudá. Je-li parita lichá, pak počet datových bitů logické úrovně 1 + paritní bit musí být liché číslo, analogicky je-li parita sudá, musí být počet datových bitů v logické 1 + paritní bit sudé číslo. Jeden přenosový rámec je tedy tvořen kompletní skupinou přenášených dat o délce 7 nebo 8 bitů doplněných o start bit, stop bit a případnou paritu. Na obrázku 2.1 je znázorněn jeden přenosový rámec sériové asynchronní komunikace. Rámec je minimální přenosovou jednotkou - data lze přenášet pouze po celých rámcích. Během přenosu jednoho celého rámce se nesmí generátory vzájemně rozejít o více než zhruba polovinu doby jednoho bitového intervalu, pokud by se generátory rozešly o více než polovinu bitového intervalu, došlo by k chybnému vyhodnocení přenosu. Díky tomu, že k synchronizaci dochází v rámci každého přenosu a je možná jistá odchylka, nejsou nároky na přesnost generátorů hodinových signálů až tak vysoké. V krajním případě se mohou kmitočty generátorů lišit o zhruba 4-5%.



Obrázek 2.1: Přenosový rámec sériové asynchronní komunikace

Přenosová rychlost u asynchronní komunikace se udává v baudech. Baud je jednotka používaná k měření rychlosti přenosu dat, přičemž udává počet změn signálu za sekundu. U asynchronní sériové komunikace se na datové lince vyskytují pouze dva stavy - logická 0 a 1. Přesto však nelze slučovat pojem bity za sekundu s pojmem baud, protože u asynchronní komunikace, jak bylo výše popsáno, existuje určitá režie přenosu v podobě synchronizačních start bitů, kontrolních bitů parity a stop bitů. Proto např. při přenosové rychlosti 9600 baud při přenosu 8 bitové délky dat bez paritní kontroly je přes datovou linku přeneseno maximálně pouze 7680 bitů dat.

Asynchronní komunikace je vhodné užít zejména tam, kde cena vícevodičové linky hraje důležitou roli a kde není potřeba vysokých přenosových rychlostí na velké vzdálenosti. Dle standardu RS-232 je totiž maximální délka komunikační linky při přenosové rychlosti 19200Bd pouze 15m, avšak při snížení přenosové rychlosti na 2400Bd je již 900m a za dodržení jistých okolností (kapacita vodiče) je možno délku spoje ještě prodloužit.

3 Modul SCI

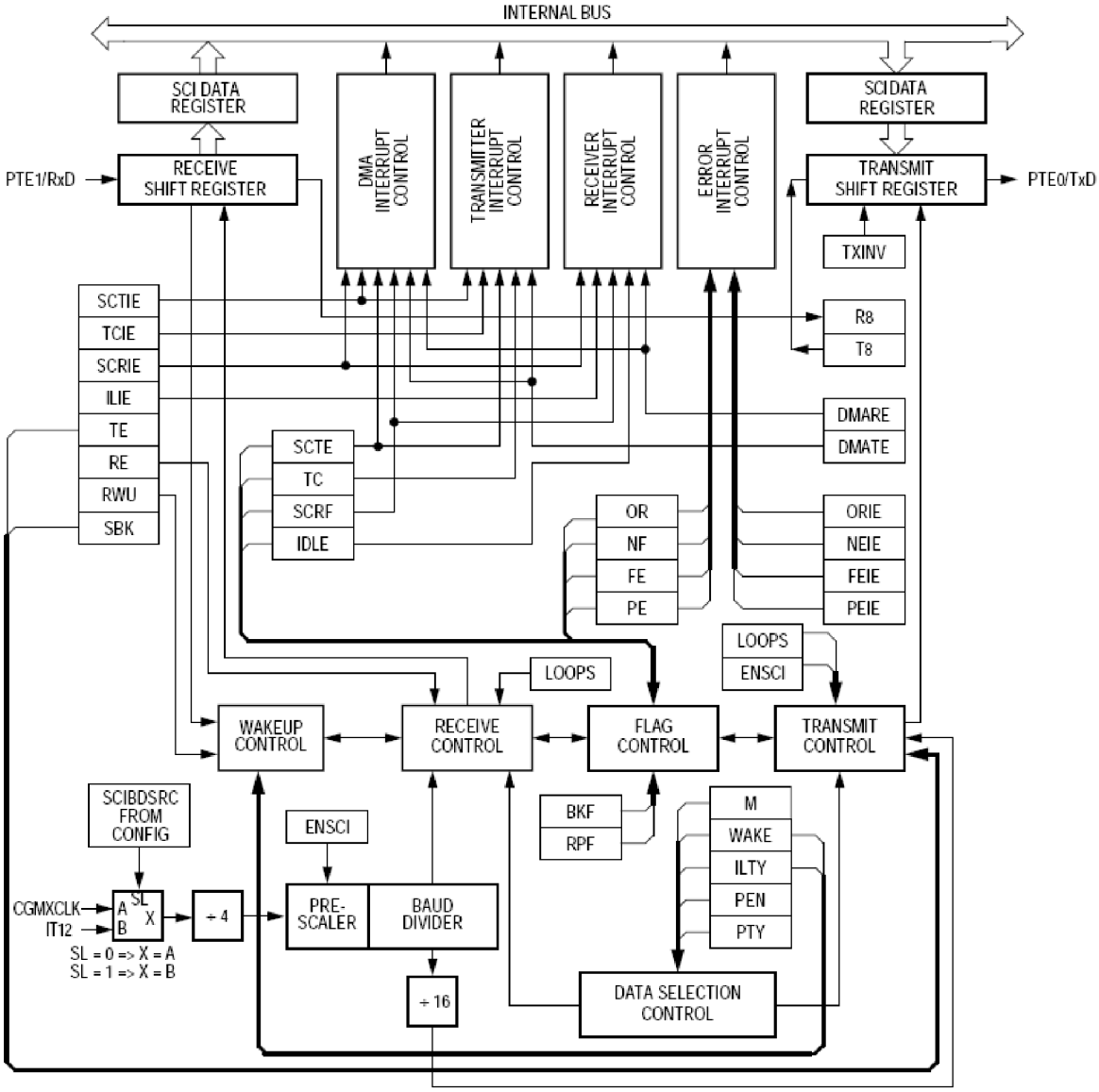
Periferie SCI u mikrokontrolérů rodiny HC08 obstarává asynchronní sériovou komunikaci s okolím. Modul SCI je realizován prostřednictvím obvodu UART (universal asynchronous receiver transmitter - univerzální asynchronní vysílač a přijímač). Modul je ovládán pomocí registrů mapovaných do paměti mikrokontroléru. Rozhraní obsahuje dva stavové registry, tři řídicí registry, jeden datový registr a registr přenosové rychlosti. Od nastavení parametrů přenosu a spuštění periferie je následná činnost plně autonomní, nevyžaduje pozornost procesoru. Modul umožňuje vyvolávat řadu přerušení a automaticky detekuje různé chyby, které mohou během přenosu nastat.

Vlastnostmi periferie je přenos v NRZ (non return to zero) formátu, tedy klidová úroveň linky je v logické 1, plně duplexní režim, takže je možné data vysílat i přijímat nezávisle na sobě v tom samém okamžiku. Přenosovou rychlost je možné nastavit v širokém rozsahu na jednu z 32 běžně používaných přenosových rychlostí, např. u modelu mikrokontroléru GP20 rodiny HC08 je při taktovací frekvenci sběrnice 4,9152MHz možné nastavit přenosové rychlosti od 46Bd do 76800Bd. Programově je možno nastavit délku dat 8 nebo 9 bitů. Vysílač i přijímač generují nezávislá přerušení a dokonce je možno přijímač a vysílač spouštět naprosto nezávisle na sobě. Pokud to aplikace vyžaduje (např. z důvodu redukce množství podpůrných obvodů), je možné programově nastavit polaritu výstupu - záměnu logické 0 a 1, v takovém případě je pak klidová úroveň linky v logické 0. Provoz modulu je řízen pomocí přerušení s osmi různými příznaky - vysílač nevyužit, vysílání dokončeno, přijímač zaplněn, přijímač nevyužit, chyba přetečení dat přijímače, chyba zašumění přijímaných dat, chyba časování přijímaných dat a chyba parity. Detekce zašumění přijímaných dat probíhá trojnásobnou kontrolou úrovně datové linky přijímače zhruba uprostřed bitového intervalu s vzorkovací frekvencí, která je vždy šestnáctkrát vyšší než aktuální přenosová rychlost. Pomocí konfiguračního registru je možno určit zdroj hodinového signálu pro modul SCI. Na obrázku 3.1 je blokové schema periferie.

3.1 Popis registrů modulu SCI

Řízení a kontrola stavu modulu SCI probíhá pomocí registrů mapovaných do paměti. Tyto registry mohou mít odlišné chování při jejich zápisu nebo čtení. Například zápisem do datového registru jsou vysílači určena data k odeslání a čtením téhož registru získáme data přijatá přijímačem. Některé registry, resp. některé jejich bity není možné nastavit pouhým zápisem do nich, ale je k tomu třeba přesně dané posloupnosti instrukcí, například čtení ze dvou různých registrů ve dvou po sobě následujících instrukcích. Následuje popis jednotlivých registrů včetně popisu jejich funkce. Detailní

popis instrukčních sledů pro zápis jednotlivých bitů stavových registrů bude vynechán, případné zájemce odkazuji např. na lit. [1].



Obrázek 3.1: Blokové schema modulu SCI

3.1.1 SCC1 (řídicí registr č. 1)

Nastavením příslušného bitu se zapíná nebo vypíná celý modul SCI, zapíná nebo vypíná se smyčkový režim. Při smyčkovém režimu se přijímač odpojí od příslušného I/O pinu mikrokontroléru a na jeho vstup je připojen výstup vysílače. Je možné nastavit inverzní výstup - záměnu logické 0 a 1, včetně klidové úrovně! Rovněž se zde volí délka dat osm nebo devět bitů. Při devítibitové délce dat slouží devátý bit jako druhý stop bit, adresová značka pro přepnutí přijímače z pohotovostního do normálního režimu, nebo jako paritní bit. Nastavuje se zde, zda se přijímač přepne z pohotovostního

do normálního režimu pomocí adresové značky (nejvíce významný bit přijatého znaku) nebo pomocí klidové úrovně datové linky. Jeden z bitů registru slouží k nastavení okamžiku, odkdy se počítá doba klidového stavu na lince, zda se počítá od start bitu nebo od stop bitu. Zbývající bity jsou určeny pro povolení resp. zakázání paritního bitu a v případě povolení paritního bitu i určení, zda jde o paritu lichou nebo sudou.

3.1.2 SCC2 (řídicí registr č. 2)

Pomocí registru SCC2 se povoluje přerušení vysílače, přerušení při příznaku dokončeného odesílání (transmission complete), přerušení přijímače a přerušení přijímače při nastaveném IDLE bitu. Dále registr slouží k spouštění vysílače a povolení činnosti přijímače. Je zde možno přepnout přijímač do pohotovostního režimu, kdy negeneruje žádná přerušení. Do normálního režimu je opětovně spuštěn pomocí adresové značky nebo klidové úrovně na lince podle toho, jak je to určeno v registru SCC1. Jeden z bitů určuje, jak se má zachovat vysílač v okamžiku, kdy nevysílá žádná data - zda má vysílat samé 1 nebo samé 0.

3.1.3 SCC3 (řídicí registr č. 3)

Ukládá přijatý devátý bit při devítibitovém přenosu. Povoluje přerušení od příznaků přetečení přijímače, zašumění, chybu časování nebo chybu parity.

3.1.4 SCS1 (stavový registr č. 1)

Obsahuje příznaky dokončení přenosu dat z datového registru do posuvného registru vysílače, příznak dokončeného odesílání (včetně stop bitu), příznak dokončeného přesunu posuvného registru přijímače do datového registru - určuje tak okamžik, kdy je možno přijímaná data přečíst. Dále obsahuje příznak klidového stavu na lince přijímače a chybové příznaky přetečení přijímače, zašumění dat, chybu časování a chybu parity.

3.1.5 SCS2 (stavový registr č. 2)

Uchovává pouze dva příznaky. Jeden určuje, zda je přijímač aktivní. Druhý určuje, zda byl přijímačem přijat znak pauzy - samé nuly nenásledované stop bitem.

3.1.6 SCDR (datový registr)

Datový registr rozhraní SCI má tu vlastnost, že data do něj zapsaná slouží k odeslání, zatímco čtením téhož registru jsou získána data přijatá přijímačem.

3.1.7 SCBR (registr přenosové rychlosti)

V registru přenosové rychlosti jsou dva bity vyhrazeny pro nastavení předděličky a tři bity pro samotnou děličku taktovací frekvence. Výsledná frekvence vnitřních hodin modulu SCI (které určují přenosovou rychlost) se odvozuje od vstupího hodinového signálu, kterým zpravidla bývá hodinový signál systémové sběrnice. Jeho postupným dělením v předděličce a následně v děličce hodinového signálu vzniká výsledný hodinový signál periferie SCI. Dělička provádí dělení frekvence ve dvou krocích. Jedním krokem je dělení konstantou, druhum je dělení uživatelsky nastavitelnou hodnotou. Např u mikrokontroléru GP20 je konstanta pevně stanovena na číslo 64. Výsledny vzorec pro výpočet přenosové rychlosti potom vypadá následovně: přenosová rychlost = frekvence sběrnice / (64 x hodnota předděličky x hodnota děličky).

4 Problematika sUART

4.1 Možná řešení sUART

Realizovat sUART je u mikrokontrolérů rodiny HC08 možné pouze s určitými omezeními. Hlavním omezením je pouze poloviční duplex (v jednom okamžiku lze data buď pouze odesílat nebo jenom přijímat) a omezení přenosové rychlosti. Díky polovičnímu duplexu je třeba řešit priority přenosů (přijem/vysílání). Omezení přenosové rychlosti je dáno dvěma faktory. Prvním je režie obsluhy která má při vyšších přenosových rychlostech značný vliv, druhým faktorem omezujícím přenosovou rychlost je omezená přesnost vnitřních hodin. S ohledem na počet vývodů pouzder (viz obrázek 4.1) mikrokontrolérů řady QT (8 vývodů) a QY (16 vývodů), je téměř nemožné nasazení vnějšího krystalu, resp. není nemožné, ale je nepravděpodobné, aby bylo možné využívat další vývody obecných portů těchto mikrokontrolérů. Přesnost vnitřních hodin mikrokontroléru je možné v jistém rozsahu kalibrovat pomocí registru, který je vyhrazen pouze pro tento účel. Přes všechna tato omezení je přesnost vnitřních hodin pro nižší přenosové rychlosti (9600baud) dostatečná a sUART je možné, byť s omezeními, realizovat. Existují dva základní režimy provozu sUART. Konkrétně jde o režim dotazování (též známé pod pojmem polling) a režim s využitím přerušení vnitřního časovače.

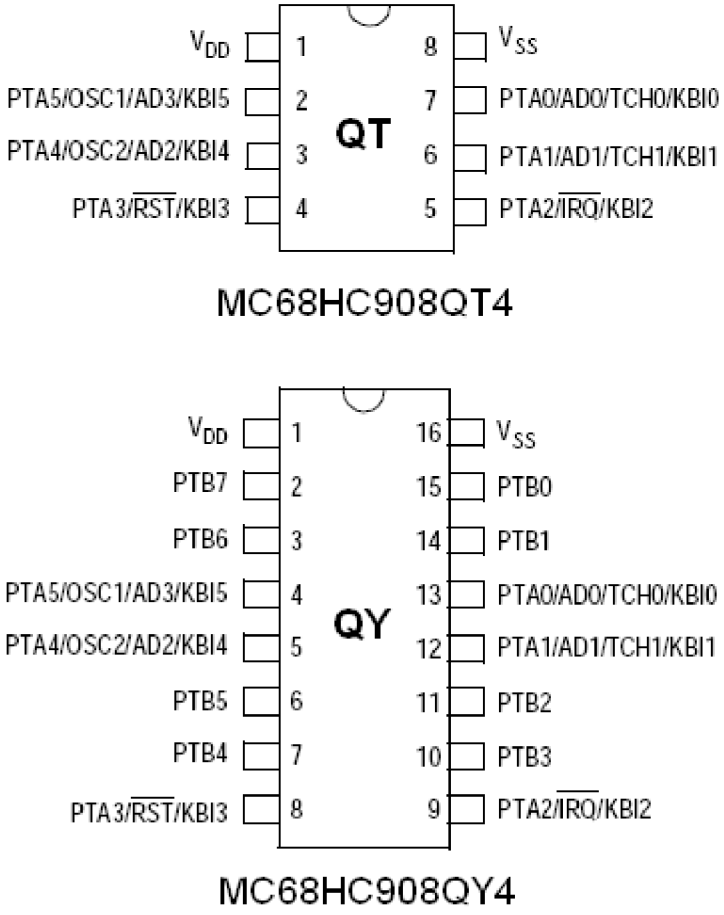
4.1.1 Režim dotazování

V okamžicích nečinnosti využívá čekací smyčky, která plně zaměstnává procesor. Hlavní výhodou je jednoduchost takové implementace, nicméně přináší několik nevýhod. Přepínání mezi během hlavní smyčky programu a čekací smyčkou vyžaduje jistou režii procesoru, se kterou je třeba počítat a kompenzovat ji patřičným upravením časových intervalů samotné čekací smyčky. Druhou nevýhodou je plné vytížení procesoru a tím znemožnění konání užitečné práce během přenosů. Navíc díky různé době vykonávání instrukcí dochází v rámci jednoho rámce ke zkreslení, které se postupně zvětšuje!

4.1.2 Režim s přerušením obecného časovače

K odpočítávání prodlev není užito čekací smyčky, ale využije se přerušení, které může generovat vestavěná obecný časovač. Výhodou je možnost obsluhovat další procesy během činnosti sUART. Nevýhodou je náročnost takové implementace. I u režimu s využitím přerušení obecného časovače je třeba počítat s určitou režií během přepínání, protože k přepnutí nedochází ihned v okamžiku, kdy časovač vygeneruje přerušení, ale až po dokončení prováděné instrukce. V rámci jednoho bitového intervalu může docházet k drobným nepřesnostem, ale narozdíl od dotazovacího režimu se toto

zkreslení neakumuluje, protože časovač vygeneruje přerušení, ale čítá bez zastavení další časový interval.



Obrázek 4.1: Pouzdra a umístění pinů mikrokontrolérů

4.2 Existující sUART

Pro osmibitové mikrokontrolery Motorola se mi podařilo najít dvě varianty sUART z nichž jedna je primárně určena pro jádro HC05 (zdroj www.freescale.com), se kterým je jádro HC08 po stránce zdrojových kódů kompatibilní, druhá varianta je určena přímo pro jádro HC08 pro modely QT a QY. Druhá varianta je dostupná na doprovodném CD k publikaci [2].

Varianta pro HC05 je založena na principu režimu dotazování (pollingu). Jde o prostou simulaci UART bez výrazné podobnosti s rozhraním SCI, nastavování parametrů přenosu je možné pouze zásahem do zdrojového kódu samotné implementace sUART. Tato varianta neumožňuje téměř žádné nastavení. Přínosem této varianty je třínásobné testování na stav linky uprostřed bitového intervalu, čímž se připodobňuje hw realizaci alespoň v tom smyslu, že může detekovat případné zašumění. Kromě zašumění detekuje chybu časování rámce. Varianta pro HC08 je založena na

principu přerušení obecného časovače. Již je zde snaha o podobnost s konfiguračním rozhraním modulu SCI, avšak není to dotaženo do konce, jsou zde sice obsaženy pseudoregistry v operační paměti (dva datové a jeden stavový), stále však chybí řídicí registry a registr přenosové rychlosti. Navíc stavový registr má pouze omezený rozsah příznaků. Tyto příznaky indikují: aktivní vysílání, datový registr vysílače zaplněn, přijímač zaplněn, přetečení přijímače, přijímač aktivní a chyba časování. Jak je vidět, chybí příznak chyby zašumění, který by zde ani neměl význam, protože tato varianta při příjmu kontroluje stav na lince pouze jednou uprostřed bitového intervalu. Obě výše zmíněné varianty umožňují pouze osmibitový přenos.

4.3 Způsob řešení vlastního sUART

Na základě analýzy stávajícího stavu a dostupných řešení sUART jsem vytyčil požadavky a způsob řešení vlastního sUART tak, aby vytvořené řešení odstranilo nedostatky již existujících řešení a využilo jejich předností. Mé řešení bude využívat přerušení obecného časovače. Protože mikrokontroléry QT a QY obsahují pouze jeden nezávislý časovač, je možné docílit pouze poloviční duplexu při přenosu. Je proto třeba řešit otázku priorit přenosu. V mém řešení jsem se rozhodl v případě konfliktu upřednostnit přijímaná data před odesílanými a po dokončení příjmu opakovat případné přerušené odesílání dat. Nastavování sUART bude probíhat pomocí pseudoregistrů v operační paměti. Narozdíl od existujících řešení budu ve vlastním řešení usilovat o větší podobu s modulem SCI a to včetně nastavení přenosové rychlosti, které je u předchozích řešení možné pouze zásahem do kódu vlastní implementace sUART. V rámci nastavení přenosové rychlosti budu pravděpodobně donucen omezit nejvyšší přenosové rychlosti, nicméně předpokládám bezproblémovou funkci při rychlostech do 19200baud. Při příjmu bude úroveň na lince detekována opakovaně uprostřed bitového intervalu, aby bylo možné detekovat chybu zašumění. Dále bude možné nastavit paritu při vysílání a při příjmu detekovat chybu parity. Rozhraní sUART bude namapováno na port A obecného užití, který je u obou modelů (QT, QY) shodný. Při přenášení na jiné mikrokontroléry rodiny HC08 je možné, že bude třeba využít jiných portů.

5 Vlastní sUART

Na základě požadavků vytyčených v předchozí kapitole jsem navrhl a implementoval sUART. Namapování sUART na piny mikrokontroléru jsem provedl tak, že přijímač je na bitu 0 portu A, protože tento pin je možné využít k detekci sestupné hrany obecným časovačem, a vysílač je na bitu 1 portu A. Během implementace jsem byl donucen přistoupit k jistým omezením, aby byla zachována požadovaná funkčnost. Následuje popis principu činnosti rozhraní, návod k používání, popis testování funkčnosti, popis jednotlivých pseudoregistrů a nakonec omezení, jež se nepodařilo odstranit a se kterým je potřeba za jistých okolností počítat.

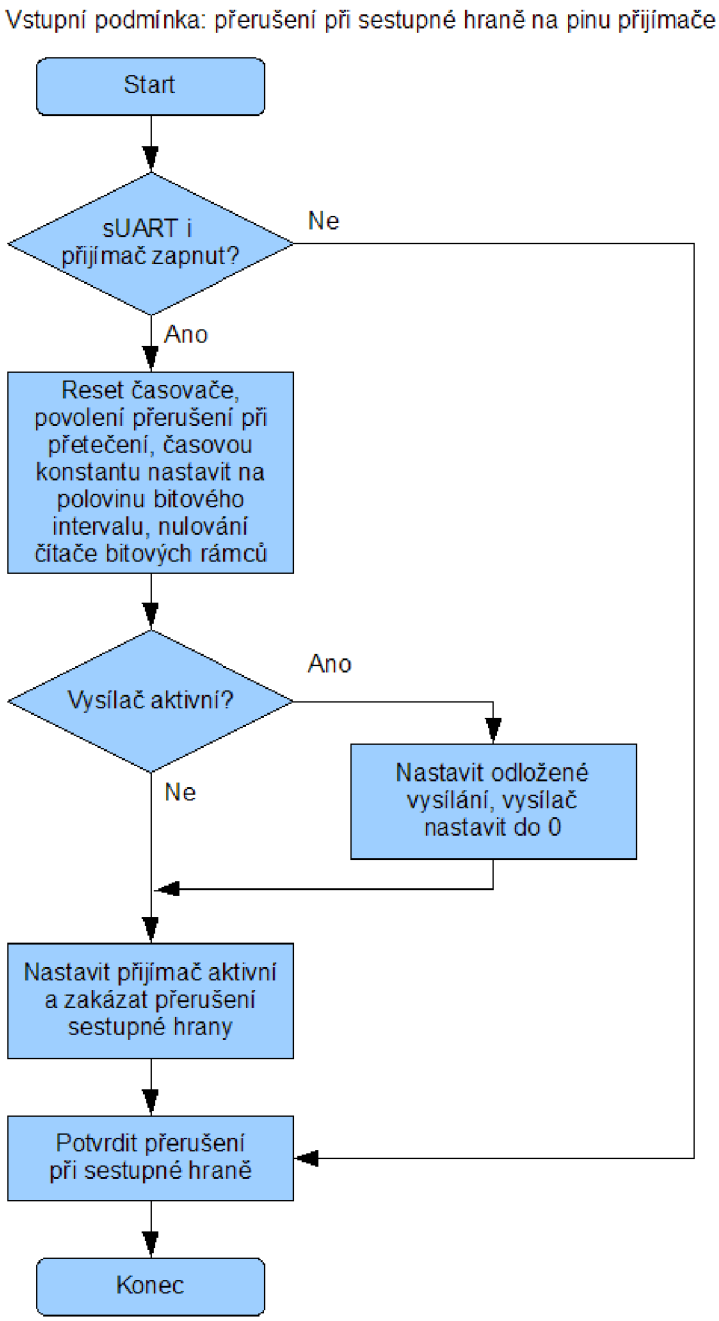
5.1 Principy činnosti vlastního sUART

Po resetu mikrokontroléru dojde k inicializaci všech pseudoregistrů, které sUART využívá, na výchozí hodnoty. Poté je nastaven obecný časovač tak, aby generoval přerušení při přetečení a při detekci sestupné hrany na bitu 0 portu A. Přerušení při přetečení je nastaveno na délku jednoho bitového intervalu odpovídajícího nastavené přenosové rychlosti. Jakmile je toto dokončeno, je řízení předáno hlavní smyčce programu.

Nastavení přenosové rychlosti probíhá ve dvou krocích. Nejprve je na základě hodnoty registru přenosové rychlosti načtena z paměti ROM hodnota přetečení obecného čítače odpovídající bitovému a polovině bitového intervalu do příslušných proměnných, odkud jsou, podle potřeby aplikace, kopírovány do registrů TMODH a TMODL. Toto dvoufázové načtení je užito zejména z důvodu časové náročnosti načtení konstant z paměti, kde se užívá indexovaného přístupu do paměti.

5.1.1 Přijímač

Při příchodu přerušení kanálu 0 obecného časovače (záchytu sestupné hrany) je kontrolováno, zda je sUART i přijímač zapnutý či nikoliv. Pokud je jak sUART, tak přijímač zapnutý, proběhne inicializace proměnných využívaných přijímačem a nezbytná nastavení časovače. U časovače je zakázáno generování přerušení při sestupné hraně kanálu 0 časovače a je nastavena hodnota přetečení časovače odpovídající poloviční délce bitového intervalu, aby detekce úrovně na lince probíhala přibližně uprostřed bitového intervalu. Pokud není buď celý sUART nebo přijímač zapnutý, vynuluje se příznak přerušení a řízení je předáno hlavní smyčce programu. Na obrázku 5.1 je znázorněn vývojový diagram obsluhy přerušení generovaného při sestupné hraně na pinu přijímače.



Obrázek 5.1: Vývojový diagram obsluhy přerušení při sestupné hraně

Při příchodu přerušení při přetečení časovače je nejprve kontrolováno, zda je přijímač aktivní, pokud ano, dojde k sejmutí úrovně linky, pokud přijímač aktivní není, je předáno řízení vysílači. Sejmutí úrovně (viz Kód 5.1) je provedeno pomocí instrukce podmíněného skoku, která nastavuje příznak carry. Je-li splněna podmínka (úroveň linky je v 1) dojde ke skoku a nastaví se příznak carry. Návěští skoku je před následující instrukcí přičítání s přenosem. Sled instrukcí je zde tedy stejný jak v případě splnění, tak v případě nesplnění podmínky skoku - instrukce podmíněného skoku je zde využita pouze pro případné nastavení příznaku carry.

```
          clra                          ; Nulování ACC
          brset 0,PORTA,rx_samp1        ; Načtení prvního vzorku
rx_samp1: adc #0
          brset 0,PORTA,rx_samp2        ; Načtení druhého vzorku
rx_samp2: adc #0
          brset 0,PORTA,rx_samp3        ; Načtení třetího vzorku
rx_samp3: adc #0

V ACC jsou na bitu 1 filtrovaná data, na bitu 0 informace o zašumění.
```

Kód 5.1: Sejmutí úrovně linky přijímačem

Teprve poté je testováno, zda je přijímač a sUART zapnutý (uživatel jej mohl mezitím před dokončením přenosu vypnout), pokud je oboje zapnuto, vyhodnotí se úroveň linky a případný šum na lince. Vyhodnotí-li přijímač v místě startbitu úroveň linky v log. 1, je toto považováno za falešný startbit, časovač se opět nastaví na záchyt hrany a řízení je předáno hlavní smyčce. Úspěšně přijatý startbit následuje přenastavení časovače na generování přerušení při přetečení po vypršení času odpovídajícího jednomu bitovému intervalu, pokud je aktivní vysílač, je též nastaven příznak odloženého vysílání. Při přijetí datového bitu je uložena hodnota linky a informace o zašumění, viz kód 5.2. Po úspěšném přijetí stopbitu je zkontrolována parita, je-li nastavena, a případně nastaveny chybové příznaky – chyba parity, chyba zašumění a chyba časování. Chyba zašumění je kontrolována pomocí exkluzivní disjunkce přijatých filtrovaných dat a šumové informace, viz kód 5.3. Je-li úroveň linky v místě stopbitu vyhodnocena jako log. 0, je nastaven příznak chyby časování. Na závěr jsou zkopírována data do datového registru přijímače, pokud uživatel předchozí data mezitím vyzvedl. Pokud ne, je nastaven příznak přetečení přijímače a právě přijatá data jsou ztracena. Po dokončení příjmu dat je nastaven časovač na generování přerušení při záchytu sestupné hrany. Činnost přijímače při přerušení přetečení časovače je znázorněna na vývojovém diagramu na obrázcích 5.2 a 5.3. V kódu 5.4 je znázorněno, jakým způsobem dochází k větvení v závislosti na fázi přenosu (startbit, datové bity, stopbit).

```
                    :
          lda temp_byte             ; Načtení sejmutých úrovní
          rora                      ; Rotace A - informace o šumu do C
          ror su_nr                 ; Šum z C do registru zašumění
          rora                      ; Rotace A - data do C
          ror su_shreg              ; Data z C do posuvného registru
          brclr 7,su_shreg,rx_end   ; Pokud data v 0, skok na konec
          inc su_paritycnt          ; Data v 1 - zvýšení čítače parity
rx_end:   inc su_bitcount           ; Zvýšení hodnoty čítače b. rámců
                    :

Pozn.: C znamená příznak přenosu, využívaný u neznaménkových aritmetických operací,
       zde s výhodou využit pro jednoduchý přenos krajního bitu z jednoho registru do druhého
```

Kód 5.2: Princip uložení po sejmutí úrovně linky

```
                    :
          lda su_shreg              ; Načtení přijatých dat
          eor su_nr                 ; Exkluzivní OR s informacemi šumu,
                                    ; pokud není šum, výsledkem je 0
          beq rx_nonoise            ; Není zašuměno, skok dále
          bset 7,su_sr2             ; Je zašuměno - nastavit příznak
rx_nonoise:           :
                      :
```

Kód 5.3: Kontrola zašumění dat při příjmu

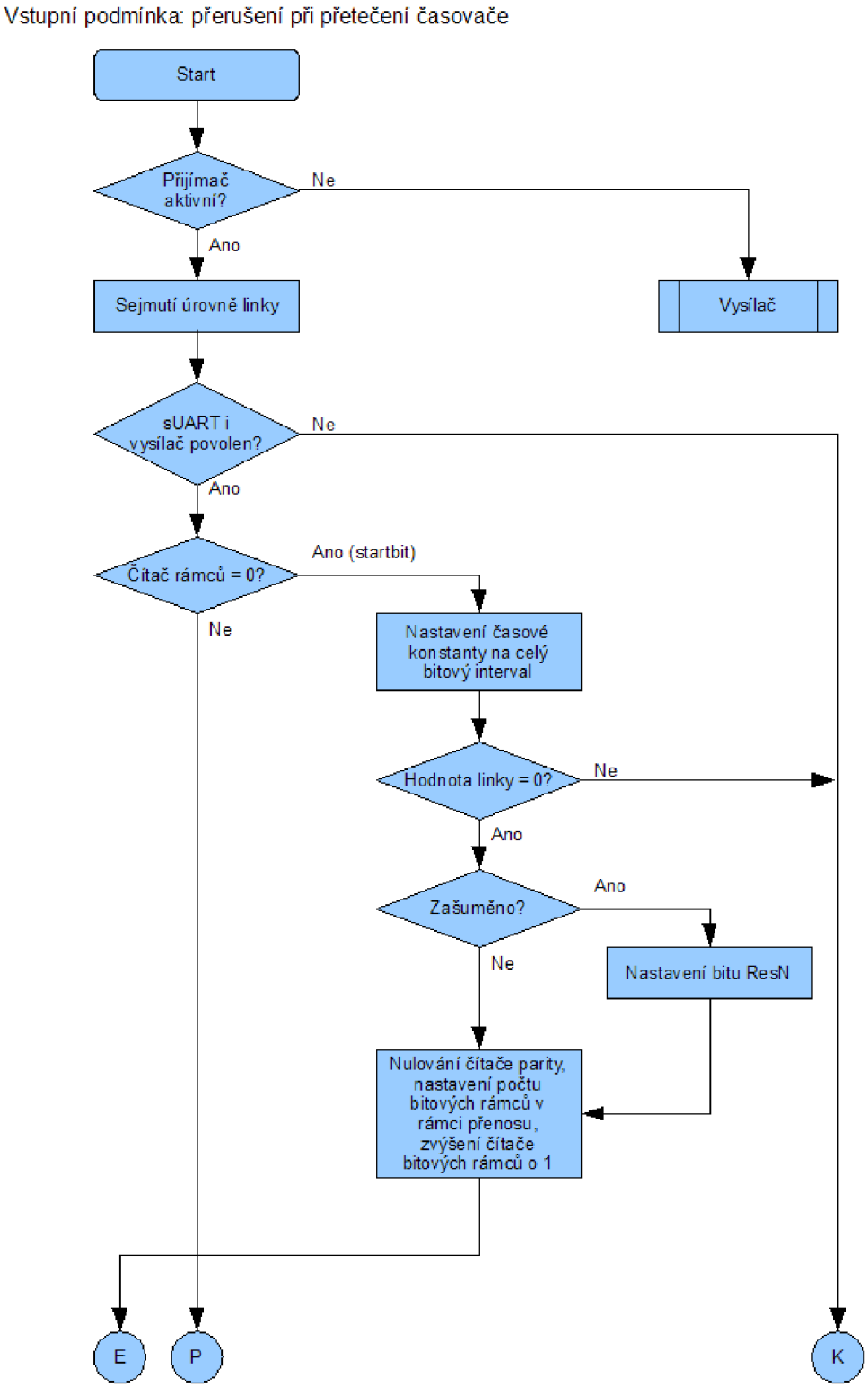
```
                    :
rx_enabled: lda su_bitcount         ; Načtení čítače bit. rámců
            cbeqa #$00,rx_startbit  ; Je-li čítač v 0, skok na startbit
            cmp su_datacount        ; Porovnání čítače s počtem b.r.
            bhi rx_stopbit          ; Čítač je větší - jde o stopbit
            beq rx_lastbit          ; Čítač je stejný - poslední bit
rx_databit:                         ; Čítač je menší - datový bit
            lda temp_byte
                    :

Pozn.: su_bitcount představuje čítač bitových rámců k určení fáze přenosu, su_datacount
       představuje počet bitových rámců během celého přenosu v závislosti na režimu (8b/9b)
```

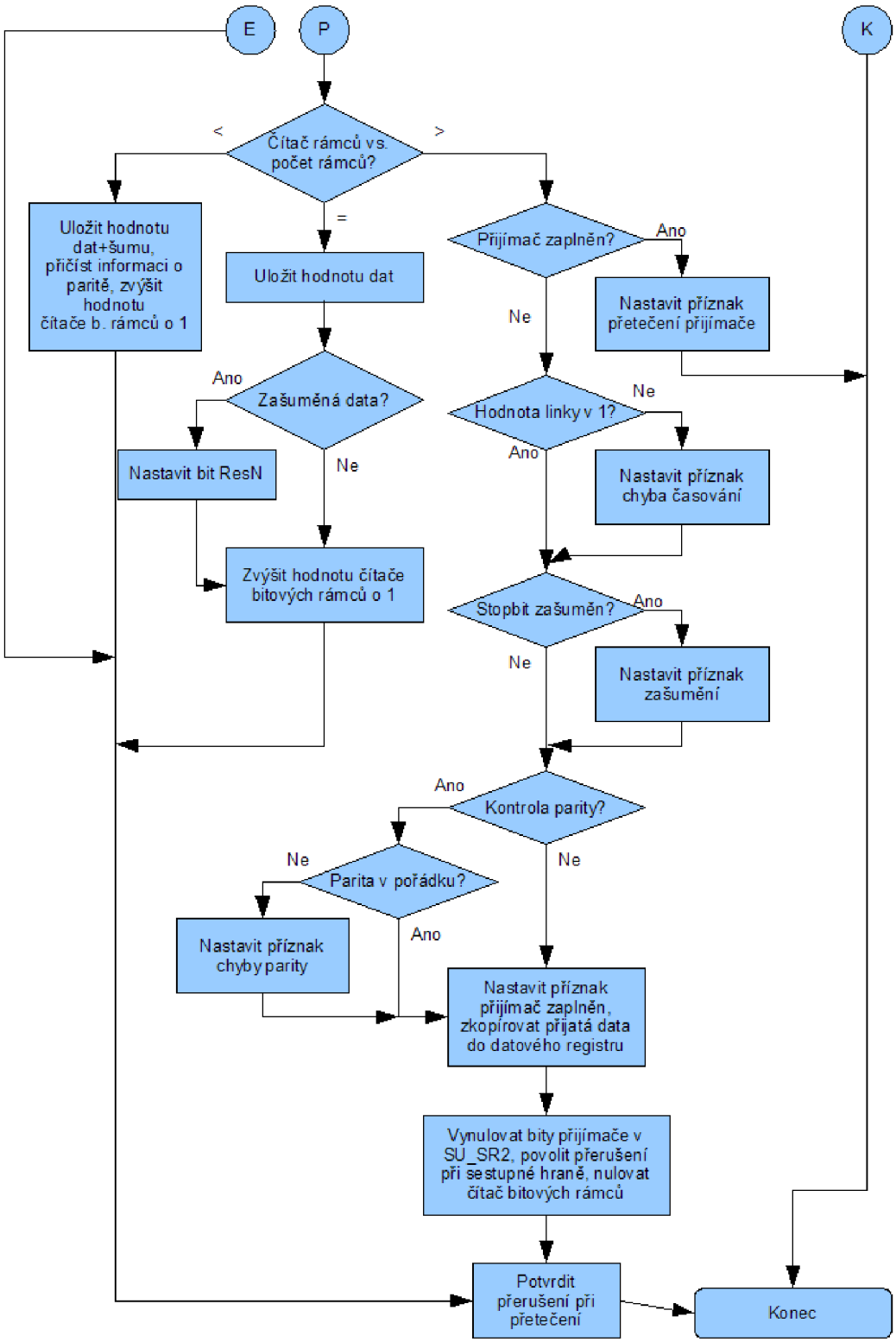
Kód 5.4: Větvení programu v rámci přenosu - přijímač

5.1.2 Vysílač

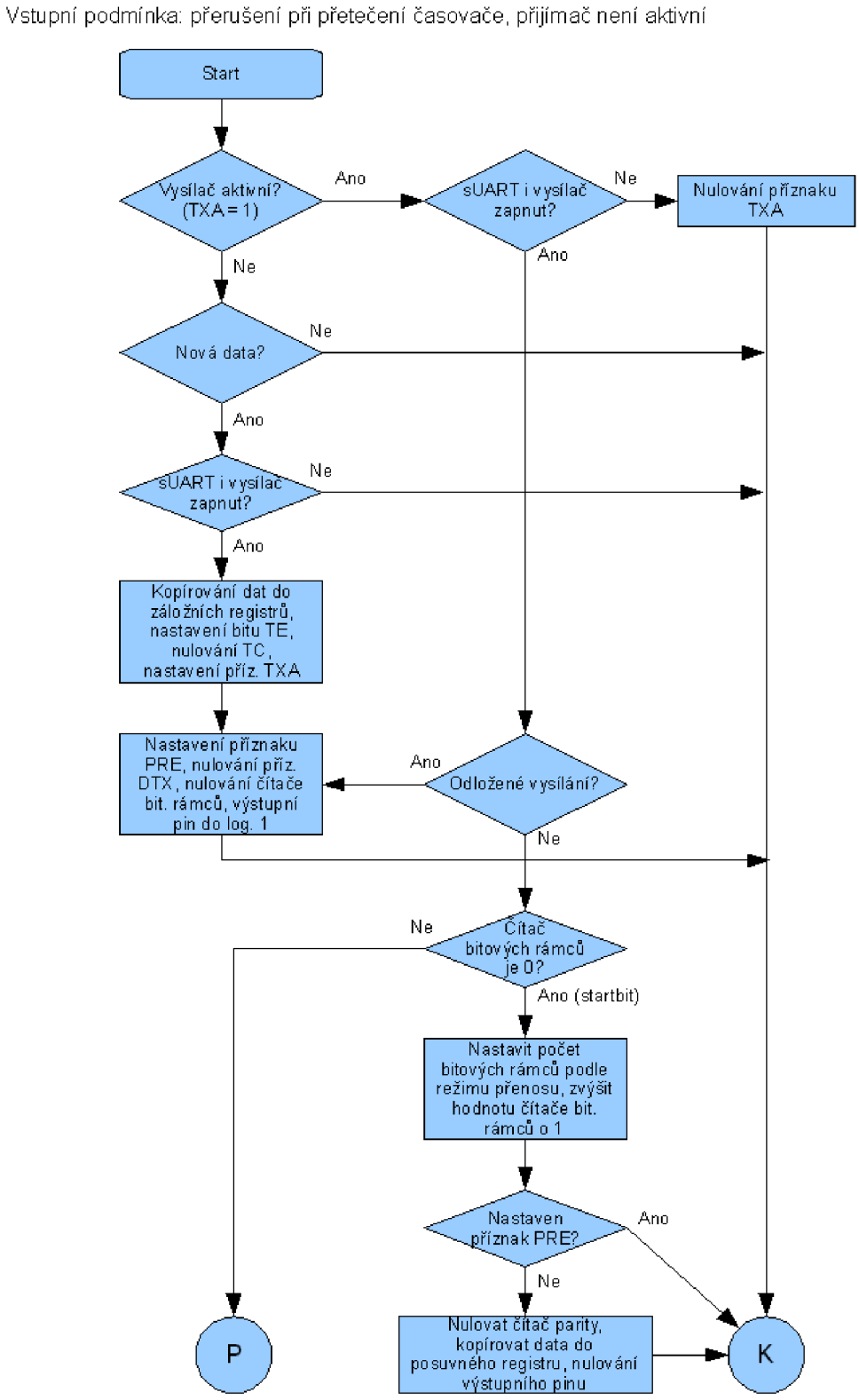
Při přerušení od přetečení časovače a neaktivním přijímači je předáno řízení vysílači. Pokud není vysílač a sUART v registru SU_CR povolen, je rutina opuštěna a řízení je předáno hlavní smyčce programu, jinak se kontroluje, zda je vysílač aktivní. Pokud je vysílač aktivní, nastaví se úroveň linky do úrovně odpovídající příslušenému vysílanému bitu. Při odesílání datových bitů je průběžně počítána parita, aby bylo možné v případě povolené kontroly parity automaticky nastavit hodnotu paritního bitu. Pokud vysílač aktivní není, kontroluje se, zda uživatel zapsal nová data (bit TE v registru SU_SR1 je vynulován), pokud ano, je nastaven příznak aktivního vysílače a dojde k inicializaci vysílače. K inicializaci dojde též po vrácení řízení vysílači po odložení vysílání přijímačem. Po inicializaci se vysílá sekvence samých log. 1, aby druhá strana správně detekovala nově přijatý znak. Pokud není vysílač aktivní a nebyla zapsána nová data, je rutina opuštěna a řízení je předáno hlavnímu programu. Činnost přijímače je znázorněna na vývojovém diagramu na obrázcích 5.4 a 5.5. Princip kontroly parity při vysílání je ukázán v kódu 5.5.



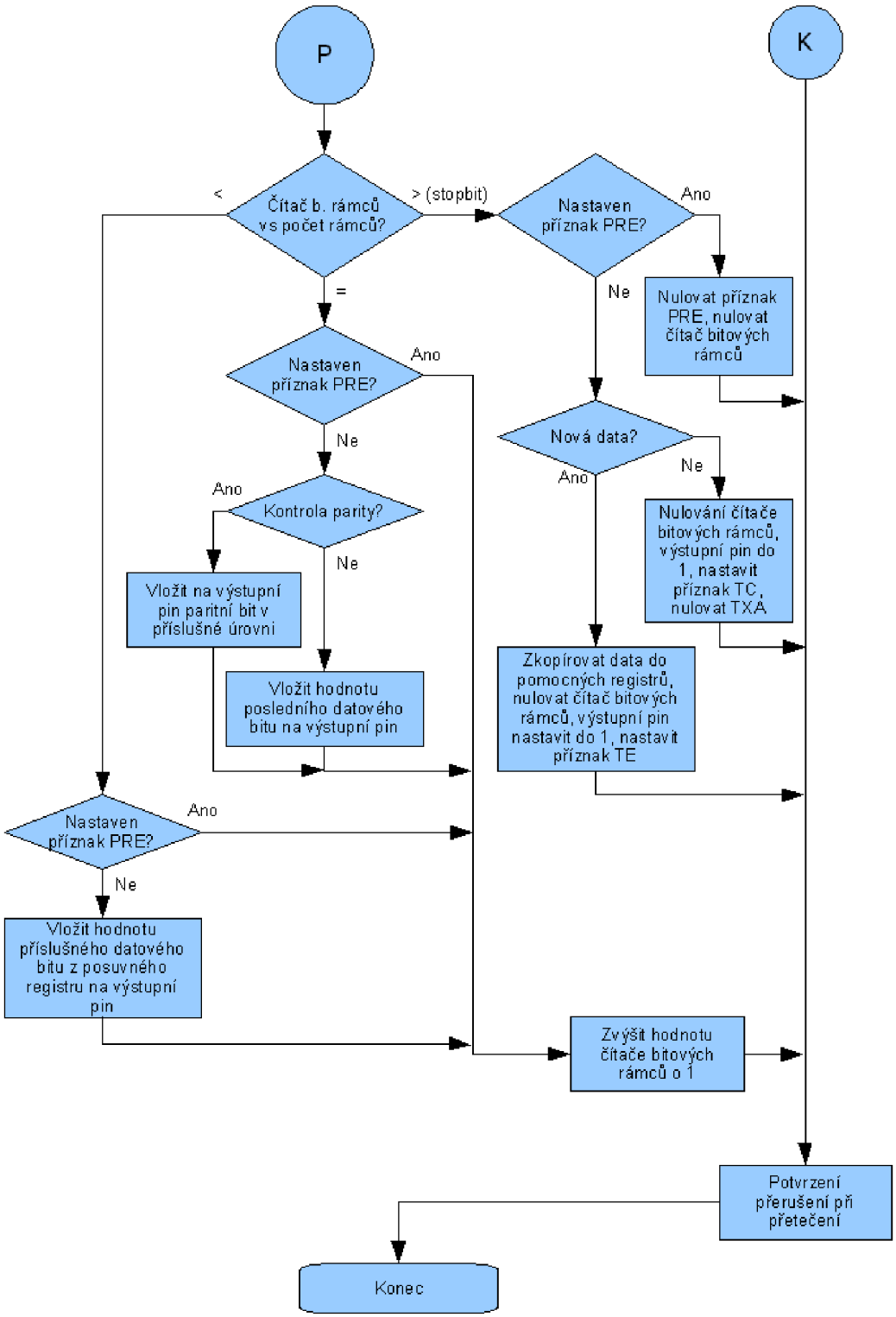
Obrázek 5.2: Vývojový diagram obsluhy přerušení při přetečení – přijímač, 1.část



Obrázek 5.3: Vývojový diagram obsluhy přerušení při přetečení – přijímač, 2.část



Obrázek 5.4: Vývojový diagram obsluhy přerušení při přetečení – vysílač, 1.část



Obrázek 5.5: Vývojový diagram obsluhy přerušení při přetečení – vysílač, 2.část

```
          :
lda su_cr             ; Načtení řídicího registru
and #$01              ; Vymaskování bitu požadované parity
eor su_paritycnt      ; Hodnota paritního bitu je nyní 0. bit v A
rora                  ; Rotací se dostane paritní bit do C
bcc tx_output0        ; Parita v 0 - skok a nastavit výstup do 0
bset 1,PORTA          ; Nastavení výstupního pinu do 1
inc su_bitcount       ; Zvýšení hodnoty čítače b. rámců
          :
```

Kód 5.5: Kontrola parity při vysílání

5.2 Návod k používání

Ovládání je možné dvěma způsoby – buď využitím připravených maker (doporučeno), nebo pomocí přímého zápisu do registrů, přičemž je třeba důsledně kontrolovat hodnoty souvisejících registrů, aby nedošlo k neočekávané chybě v důsledku nesprávné obsluhy. Přímý zápis registrů (týká se zejména datových registrů) lze doporučit pouze zkušeným uživatelům.

5.2.1 Obecné

Nastavení přenosové rychlosti je možné načtením požadované hodnoty registru přenosové rychlosti do střadače a zavoláním makra su_baudselect nebo zápisem do registru SU_BD a při neaktivním vysílači i přijímači zavolat obslužnou rutinu su_set_baudrate. Přenosovou rychlost je možné nastavit na jednu z 16 přenosových rychlostí v rozsahu od 100 do 19200 baudů pomocí nastavení bitů P1 a P0 předděličky a pomocí bitů R2, R1 a R0 děličky frekvence, kde vstupní frekvencí je frekvence sběrnice a výstupní frekvencí je frekvence hodin přenosové rychlosti.

Nastavení parametrů přenosu se provádí v registru SU_CR. Nastavením bitu M do 0 se volí 8 bitový režim, nastavením bitu M do 1 je zvolen 8 bitový režim přenosu. Při devítibitovém režimu může devátý bit sloužit jako datový bit, paritní bit nebo jako druhý stopbit. Více viz Tabulka 5.1. Nastavením bitu PEN do 1 se povolí kontrola parity, je-li bit PEN v 0, kontrola parity při příjmu se neprovádí. Při nastavené kontrole parity může přijímač generovat příznak chyby parity PE v registru SU_SR1, pokud se paritní bit neshoduje s odpovídající hodnotou paritou přenosu. Bitem PTY se volí parita přenosu. Je-li v 0, je nastavena sudá parita, je-li v 1, je nastavena parita lichá. Pokud je bit PEN v 0, nebere se ohled na hodnotu bitu PTY. Před zápisem řídicích bitů M, PEN, PTY je třeba kontrolovat, zda je vysílač i přijímač neaktivní – bity RXA a TXA registru SU_SR2. Předpokládané nejběžnější nastavení viz kód 5.6.

```
mov #$00,su_cr          ; Vypnutí sUART, vysílače i přijímače,
                        ;   8 bitový režim, vypnutá parita
mov #$01,su_br          ; Přenosová rychlost 9600bd
jsr su_set_baudrate     ; Podprogram k nastavení přenosové rychlosti
```

Kód 5.6: Obecné nastavení sUART

5.2.2 Odesílání

Doporučuje se využít makra su_write, jehož vstupem je střadač A, případný devátý bit je předáván v indexregistru X. Výstupem makra jsou pak nastavené všechny parametry, data budou odeslána bez nutnosti další obsluhy. Konec přenosu bude uživateli signalizován příznakem TC v log. 1 v registru SU_SR1. Možnost zapsat nová data bude uživatli signalizována nastavením příznaku TE do log. 1. Při manuální obsluze odesílání je třeba zapsat data a povolit činnost vysílače nastavením bitů SU_EN a TX_EN do log. 1 v registru SU_CR. Další data je možné zapisovat pouze po testování příznaku TE v registru SU_SR1, který svou úrovní 1 signalizuje připravenost vysílače odeslat nová data. Po zapsání dat je třeba příznak TE vynulovat. Příklad manuální obsluhy sUART viz kód .

```
            lda su_cr               ; Načtení řídicího registru
            ora #$C0                ; SU_EN a TX_EN do 1
            sta su_cr               ; Uložení řídicího registru
main_loop:  brclr 4,PORTA,*         ; Čekání na spouštěcí podmínku
                                    ; zde přivedení log 1 na pin 4 portu A
            mov #$66,su_td          ; Vložení znaku 'f' do datového registru
            bclr 7,su_sr1           ; Nulování TE - zahájení vysílání
            bra main_loop           ; Skok na začátek
```

Kód 5.7: Příklad obsluhy vysílání

5.2.3 Přijímání

Při přijímání dat se mohou vyskytnou různé chybové stavy a je plně na uživateli, jakým způsobem naloží s přijatými daty v případě chyby. Zde je proto ponecháno vše na uživateli. Příklad obsluhy přijímání znaku viz kód 5.8. Doporučený postup pro přijetí znaku je následující:

1. Nastavit parametry přenosu
2. Vynulovat bit RF v registru SU_SR1
3. Povolit přijímač a periferii sUART nastavením bitů SU_EN a RX_EN v SU_CR do 1
4. Testovat bit RF v registru SU_SR1 na hodnotu log. 1
5. Otestovat bity registru OVR, NE, FE, PE registru SU_SR1, zda jsou v log. 0, pokud ano, zkopírovat přijatá data v registru SU_RD, případně bit R8 v registru SU_CR, pokud některý z chybových bitů je v log. 1, vynulovat jej
6. Nulovat příznak RF

```
cteni_znaku:  bclr 5,su_sr1          ; Nulování RF - umožní přijetí znaku
              lda su_cr              ; Načtení řídicího registru
              ora #$A0               ; SU_EN a RX_EN do 1
              sta su_cr              ; Uložení řídicího registru
              brclr 5,su_sr1,*       ; Do přijetí znaku čekej zde
              lda su_sr1             ; Načtení stavového registru 1
              and #$0F               ; Vymaskování bitů OVR, NE, FE, PE
              beq bez_chyby          ; Pokud přijatý znak bez chyby, skok
              lda su_sr1             ; Načtení stavového registru 1
              and #$D0               ; Nulovat bity OVR, NE, FE, PE
              sta su_sr1             ; Uložení stavového registru 1
              bra cteni_znaku        ; Skok na začátek
bez_chyby:    lda su_rd              ; Načtení datového registru pro práci
                        :            ;   s přijatým znakem
                        :            ; Zde zpracování přijatého znaku
                        :
              bra cteni_znaku        ; Skok na začátek
```

Kód 5.8: Příklad obsluhy při přijímání znaku

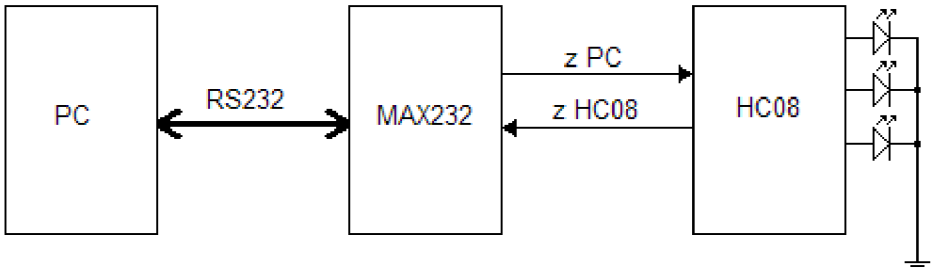
5.3 Testování funkčnosti

Před zahájením vlastního testování bylo třeba zkalibrovat vnitřní oscilátor mikrokontroléru. Vnitřní oscilátor totiž vzhledem k odchylkám ve výrobě negeneruje frekvenci sběrnice přesně 3,2MHz, ale s určitou odchylkou, ve výrobě je kalibrován pouze hrubě. Jemné doladění by zbytečně prodražilo výrobu a tím zvýšilo cenu hotových mikrokontrolérů. Navíc ve většině případů není potřeba generovat časové úseky s absolutní přesností, ale stačí přibližné hodnoty. Ke kalibraci je vyhrazen registr OSCTRIM. Jeho výchozí hodnota je $80, se kterou oscilátor generuje frekvenci sběrnice 3,2MHz s přesností ±25%, zvýšením nebo snížením hodnotry registru o 1 se změní perioda vnitřních hodin přibližně o 0,2%. Hodnota registru po jemném vyladění se liší kus od kusu čipu, ale u jednoho čipu je v rámci jeho životního cyklu konstantní. K změření periody v domácích podmínkách bez laboratorních přístrojů dobře poslouží jednoduchá aplikace, kdy je časovač nastaven generování přerušení při přetečení, je počítána maximální hodnota časovače s nastaveným dělitelem předděličky na 64. S tímto nastavením generuje časovač přerušení přibližně každých 1,31s. V rutině přerušení se pak s každým přerušením invertuje jeden z výstupních pinů na který je připojena např. LED. Příklad programu je uveden v kódu 5.9, kde s každým přerušením je změněn stav jedné diody, u druhé diody dochází ke změně stavu každé desáté přerušení. Pro usnadnění počítání je možno nastavit invertování dalšího z pinů např. jednou za deset cyklů. Pomocí takovéto aplikace se pak snadno změří doba např. 100 cyklů časovače a tato je porovnána s očekávanou hodnotou (zde 100 x 1,31s = 131s) a podle potřeby se změní hodnota registru OSCTRIM.

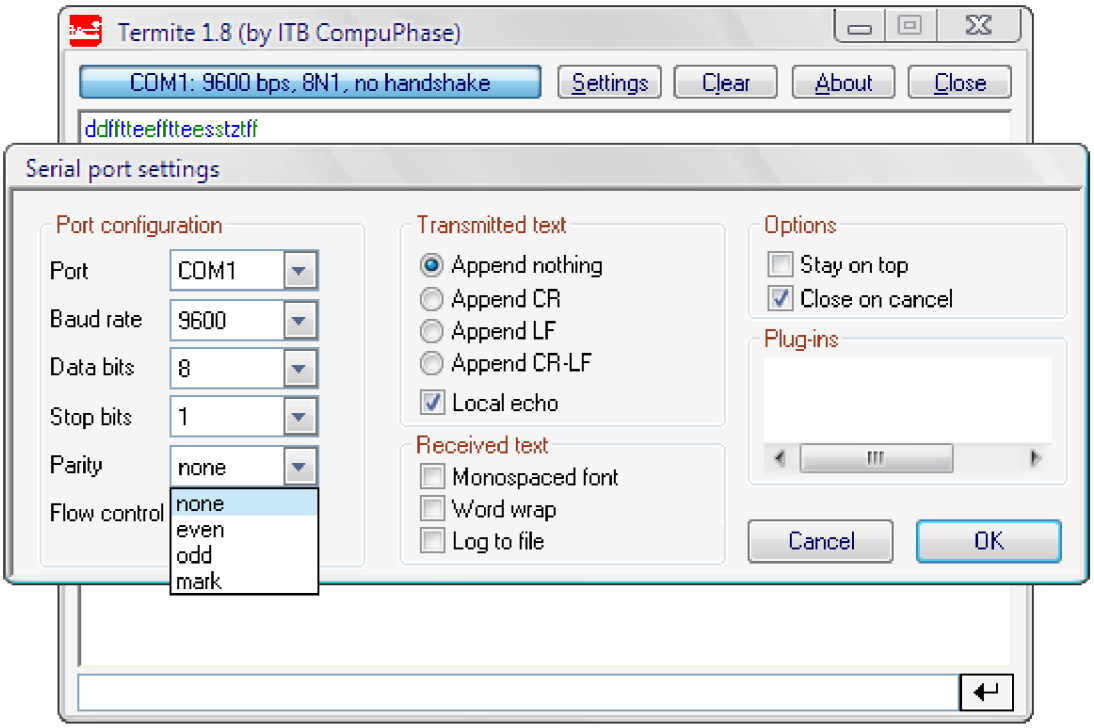
```
timovr_isr:                     ; oslužná rutina přerušení při přetečení
            inc counter         ; zvýšení hodnoty čítače o 1
            lda counter         ; načtení hodnoty čítače
            cbeqa #10,counter10 ; pokud je čítač 10, skok
            bra exit_rti        ; skok na konec
counter10:  lda PORTA           ; načtení hodnoty portu A
            eor #$02            ; změna stavu jedné diody
            sta PORTA           ; uložení hodnoty portu A
            clr counter         ; nulování čítače
exit_rti    lda PORTA           ; načtení hodnoty portu A
            eor #$08            ; změna stavu druhé diody
            sta PORTA           ; uložení hodnoty portu A
            bclr 7,TSC          ; potvrzení přerušení při přetečení
            rti                 ; opuštění obslužné rutiny přerušení
```

Kód 5.9: Obsluha přerušení při přetečení kalibračního programu

Po zkalibrování jsem provedl základní testování propojením s osobním počítačem pomocí sériového rozhraní RS-232. Viz obrázek 5.6. Ke komunikaci s mikrokontrolérem jsem užil volně šiřitelného terminálového programu pro obsluhu sériového portu Termite 1.8, který umožňuje řadu nastavení parametrů přenosu, viz obrázek 5.7. Program je přiložem na doprovodném datovém nosiči (lze jej stáhnout např. z http://www.compuphase.com). Propojení mikrokontroléru s počítačem bylo provedeno prostřednictvím převodníku úrovní max232 ve standardním zapojení. K testování bylo užito krátkého programu mikrokontroléru, který nejprve čeká na přijetí znaku při nastavené přenosové rychlosti, poté vysílač odešle sérii několika znaků. Tento test byl postupně proveden při nastavených přenosových rychlostech 4800, 9600 a 19200bd. Přenosová rychlost je nastavena vždy stejně v terminálovém progarmu i v mikrokontroléru. Tímto je otestována jednak schopnost vysílače odesílat kontinuální blok dat, jednak schopnost přijímače na dané rychlosti přijmout jednotlivý znak.



Obrázek 5.6: Blokové schema propojení mikrokontroléru a PC



Obrázek 5.7: Terminálový program Termite - nastavení

Stejného zapojení jako v předchozím testu bylo užito i u testu následujícího. Mikrokontrolér byl nastaven tak, aby při přijetí určitého znaku přijímačem rozsvítil kontrolní diodu. Případné chyby při přenosu jsou indikovány rozsvícením další diody. V terminálovém programu je odeslána posloupnost několika znaků najednou. Pokud posloupnost obsahuje předem nastavený znak, dojde k rozsvícení diody. Po přijetí každého znaku jsou testovány chybové příznaky pro případné rozsvícení kontrolní diody. Tímto testem je kontrolována schopnost přijímače přijmout souvislý blok dat, případně detekovat chybu. Test je postupně opakován při přenosových rychlostech 2400, 4800, 9600 a 19200bd. Přenosová rychlost je vždy stejná jak v terminálovém programu, tak v mikrokontroléru. Při nastavených rychlostech 2400 a 4800bd probíhala komunikace naprosto bez problémů. Při přenosové rychlosti 9600bd docházelo při přenášení delšího bloku dat k chybám. Tuto chybu přisuzuji dvěma faktorům. Prvním je ne zcela přesně zkalibrovaný vnitřní oscilátor, protože po doladění se podařilo vznik chyby oddálit ze čtvrtého přibližně na osmý přijímaný znak v rámci jednoho souvislého bloku přijímaných dat, druhým faktorem je časová náročnost provádění instrukcí sUART, která může zapříčinit změškání následujícího startbitu v případě, kdy přepnutí do obslužné rutiny přerušení trvá o několik cyklů déle – vždy je totiž třeba počkat na dokončení právě prováděné instrukce. Je třeba si uvědomit, že ke kontrole úrovně dochází uprostřed bitového intervalu, takže v případě stopbitu je k dokončení všech operací a nastavení záchytu sestupné hrany k dispozici pouze

čas poloviny bitového intervalu. Pokud se v rámci tohoto časového intervalu nestihnou potřebné operace dokončit, dojde ke zmeškání následujícího startbitu. Při přenosové rychlosti 19200bd docházelo k chybě již u druhého přijímaného znaku v rámci jednoho souvislého bloku dat, zde je však hlavní příčinou časová náročnost provádění instrukcí po stop bitu, takže ke zmeškání následujícího startbitu dojde vždy. Při přijímání jednotlivého znaku byl tento vždy správně a bezchybně rozpoznán a výše popisované chyby se neprojevily! Tento test byl zopakován s nastavením dvou stopbitů. Při nastavených dvou stopbitech již nedocházelo k chybám ani při přenosu dlouhého souvislého bloku dat při přenosové rychlosti 19200bd.

```
            bset 1,su_cr            ; Povolení kontroly parity
            bset 0,su_cr            ; Lichá parita
            ;bclr 0,su_cr           ; Sudá parita

main_loop:  bclr 5,su_sr1           ; Nulování příznaku rx_full
            brclr 5,su_sr1,*        ; Čekání na přijatý znak
            bclr 5,PORTA            ; Zhasnutí kontrolní diody
            lda su_sr1              ; Načtení stavového registru
            and #$01                ; Vymaskování bitu PE
            beq main_loop           ; Je-li PE=0, skok na začátek
            bset 5,PORTA            ; Rozsvícení kontrolní diody
            bclr 0,su_sr1           ; Nulování příznaku PE
            bra main_loop           ; Skok na začátek
```

Kód 5.10: Test parity

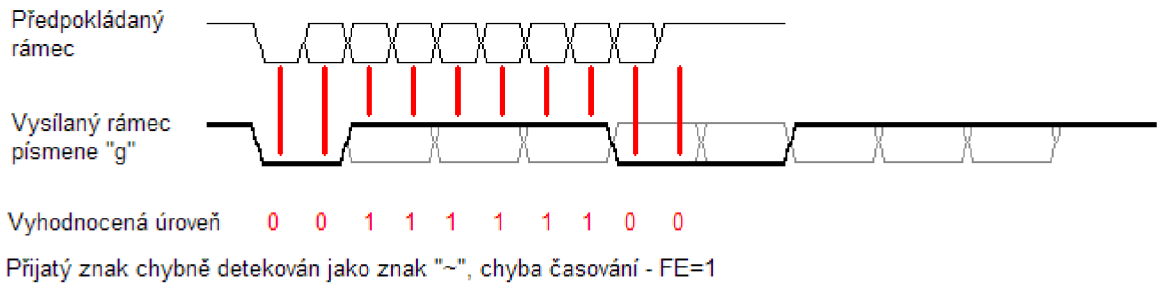
S využitím výchozího zapojení dle obr. 5.6 jsem provedl test mechanismů kontroly parity. Princip testu spočívá v nastavení kontroly parity v mikrokontroléru a následné vysílání terminálovým programem nejprve v souhlasné paritě a poté v opačné paritě, než je nastaveno v mikrokontrléru. Dojde-li během přenosu k chybě parity, je rozsvícena kontrolní dioda, viz kód 5.10. Nastavení terminálového programu je třeba zvolit odpovídajícím způsobem ke zvolenému režimu přenosu. Je-li přenos osmibitový, je třeba nastavit počet datových bitů na hodnotu 7 a zvolit příslušnou paritu. Přenos byl testován ve všech paritních kombinacích – tzn. sudá-sudá, sudá-lichá, lichá-lichá, lichá-sudá (terminál-mikrokontrolér). Při nesouhlasném nastavení došlo ke správné detekci chyby a rozsvícení kontrolní diody.

Test parity s drobnou obměnou byl zopakován. Byl zvolen devítibitový přenos, kontrola liché parity. Při chybě se změnil stav kontrolní diody. Pokud nedošlo během příjmu k chybě, byl přijatý znak odeslán terminálovému programu jako ozvěna. Terminálový program je třeba nastavit na 8 datových bitů a příslušnou paritu. Příklad tohoto testu je znázorněn v kódu 5.11, nastavení terminálového programu dle ukázky bylo 9600bd a 8O1 pro souhlasné nastavení (k detekci chyby nedochází) a 8N1 pro nesouhlasné nastavení (k detekci chyby dochází s každým přijatým znakem).

```
            mov #$01,su_br        ; Přenosová rychlost 9600bd
            jsr su_set_baudrate   ; Nastavení rychlosti
            mov #$F3,su_cr        ; Povolení sUART, vysílače,
                                  ; přijímače, 9bit režim, kont-
                                  ; rola parity, lichá parita
main_loop:  bclr 5,su_sr1         ; Nulování příznaku RF
            brclr 5,su_sr1,*      ; Čekání na přijetí znaku
            lda su_sr1            ; Načtení stavového registru 1
            and #$0F              ; Vymaskování chybových bitů
            beq no_err            ; Pokud bez chyby, skok
            lda su_sr1            ; Načtení stavového registru 1
            and #$F0              ; Nulování chybových bitů
            sta su_sr1            ; Uložení stavového registru
            lda PORTA             ; Načtení hodnoty portu A
            eor #$08              ; Změna stavu kontrolní diody
            sta PORTA             ; Uložení hodnoty portu A
            bra main_loop         ; Skok na začátek
no_err:     lda su_rd             ; Načtení přijatých dat
            sta su_td             ; Uložení dat k odeslání
            bclr 7,su_sr1         ; Nulování příznaku TE
            brclr 7,su_sr1        ; Čekání na dokončení vysílání
            bra main_loop         ; Skok na začátek
```

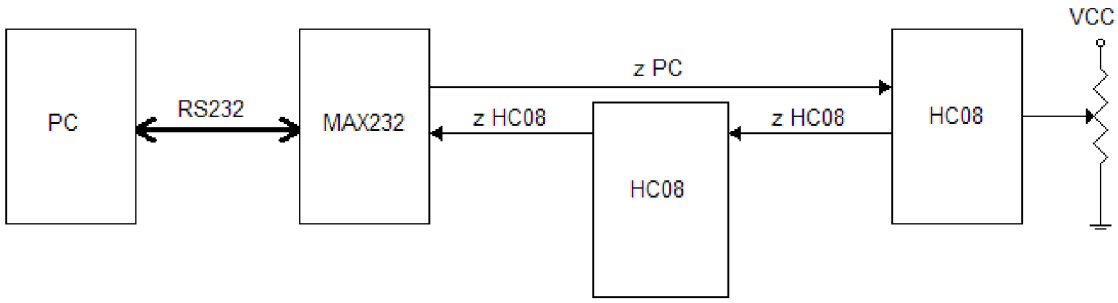
Kód 5.11: Test parity - 9bitový režim

Další test je zaměřen na kontrolu funkčnosti testování chyby časování (framing error). Tato chyba nastane, je-li vyhodnocena úroveň v místě očekávaného stop bitu jako log 0. Toho lze docílit nastavením rozdílné přenosové rychlosti v terminálovém programu a v mikrokontroléru a odesláním vhodného znaku (např. znak „g“ - 67h) terminálovým programem s nastavenou nižší přenosovou rychlostí. K testování lze využít zapojení dle obrázku 5.6. Na obrázku 5.8 je znázorněn takový test, kdy terminálovým programem s nastavenou přenosovou rychlostí 4800bd je vyslán znak „g“, zatímco mikrokontrolér je nastaven na přenosovou rychlost 9600bd. Červeně jsou znázorněny okamžiky, kdy dochází ke snímání úrovně linky mikrokontrolérem. Chyba časování je indikována rozsvícením kontrolní diody. Test proběhl přesně podle očekávání, přijetí znaku, kde na pozici bitu 3 vysílaného znaku je 0, vede k detekci chyby časování.



Obrázek 5.8: Testování detekce chyby FE

Poslední test byl zaměřen na test schopnosti komunikace více mikrokontrolérů mezi sebou. Testovací zapojení obsahovalo dva mikrokontroléry z nichž jeden pouze přeposílal data ze vstupu na výstup a druhý na základě výzvy přijatým znakem provedl převod analogové veličiny na číselnou hodnotu, kterou následně odeslal. Zapojení je znázorněno na obrázku 5.9. Pro testování bylo použito přenosové rychlosti 9600bd bez parity.



Obrázek 5.9: Blokové schema testovacího zapojení více mikrokontrolérů

5.4 Popis pseudoregistrů

Následuje popis jednotlivých pseudoregistrů, včetně popisu významu jejich jednotlivých bitů a způsobu užití.

5.4.1 SU_CR (řídicí registr)

Tento registr slouží k zapínání nebo vypínání funkčnosti celého sUART a oddělenému vypínání nebo zapínání vysílače a přijímače.

	Bit 7	6	5	4	3	2	1	Bit 0
Čtení: Zápis:	SU_EN	TX_EN	RX_EN	M	R8	T8	PEN	PTY
Reset:	0	0	0	0	0	0	0	0

Obrázek 5.10: Řídicí registr SU_CR

SU_EN – Povolení sUART

Pomocí tohoto bitu dochází k povolení nebo zakázání periferie sUART. Je-li tento bit nulový, nemá hodnota dalších bitů vliv na funkci periferie. Je-li tento bit nastaven, záleží dále na hodnotách bitů TX_EN a RX_EN. Po resetu je bit SU_EN vynulován.

TX_EN – Povolení vysílače

Nastavením tohoto bitu se povolí činnost vysílače. Nulováním tohoto bitu se činnost vysílače zakáže. Je-li vynulován bit SU_EN, neovlivňuje hodnota tohoto bitu činnost vysílače a tento je zakázán. Po resetu je tento bit vynulován.

RX_EN – Povolení přijímače

Nastavením tohoto bitu se povolí činnost přijímače. Nulováním tohoto bitu se činnost přijímače zakáže. Je-li vynulován bit SU_EN, neovlivňuje hodnota tohoto bitu činnost přijímače a tento je zakázán. Po resetu je tento bit vynulován.

M – Režim přenosu

Nastavením tohoto bitu je aktivován devítibitový režim. Viz. Tabulka 5.1. Devátý bit může sloužit jako datový bit nebo jako paritní bit. Po resetu je tento bit vynulován.

R8 – Devátý bit přijímaných dat

Při devítibitovém režimu slouží tento bit pro uložení deváteho bitu přijatého přijímačem. Při nastaveném osmibitovém režimu je tento bit kopií bitu R7 registru SU_RD. Po resetu je tento bit vynulován.

T8 – Devátý bit odesílaných dat

Při devítibitovém režimu slouží tento bit pro uložení odesílaného devátého bitu, je-li nastaven bit PEN, nebere se na hodnotu tohoto bitu zřetel. Při osmibitovém režimu se na hodnotu tohoto bitu nebere zřetel. Po resetu je tento bit vynulován.

PEN – Povolení parity

Nastavením tohoto bitu se aktivuje kontrola parity. Viz. Tabulka 5.1. Je-li tento bit nastaven kontroluje se při příjmu parita přijímaných dat. Chyba parity přijímaných dat je indikována bitem PE registru SU_SR1. Při odesílání dat je paritní bit nastavován automaticky a nebere se zřetel na uživatelem definovanou hodnotu nejvíce významného bitu – konkrétně bitu T8 při devítibitovém režimu, resp. bitu T7 při osmibitovém režimu. Po resetu je tento bit vynulován.

PTY – Parita

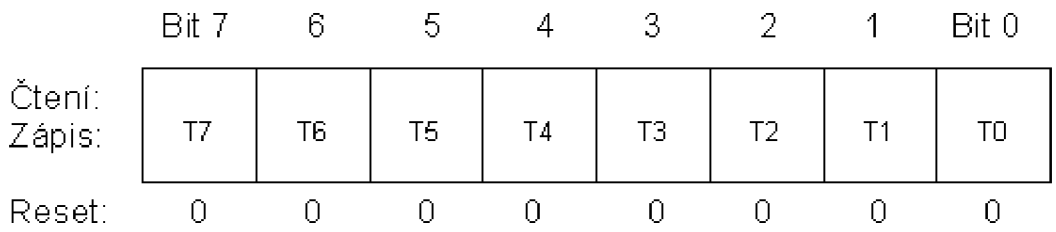
Tento bit určuje, jaká bude parita přenosu. Viz. Tabulka 5.1. Je-li PTY nastaven, je nastavena lichá parita, je-li vynulován, je nastavena sudá parita. Lichá parita znamená, že počet bitů v log. 1 plus paritní bit rovná se liché číslo. Sudá parita znamená, že počet bitů v log. 0 plus paritní bit rovná se sudé číslo. Po resetu je tento bit vynulován.

Řídicí bity		Formát přenosového rámce (znaku)				
M	PEN a PTY	Start bit	Datové bity	Parita	Stop bit	Délka znaku
0	0X	1	8	žádná	1	10 bitů
1	0X	1	9	žádná	1	11 bitů
0	10	1	7	sudá	1	10 bitů
0	11	1	7	lichá	1	10 bitů
1	10	1	8	sudá	1	11 bitů
1	11	1	8	lichá	1	11 bitů

Tabulka 5.1: Formát přenosového rámce

5.4.2 SU_TD (datový registr pro odesílaná data)

Do tohoto registru se zapisují data určená k odeslání. Devátý bit viz. řídicí registr SU_CR.



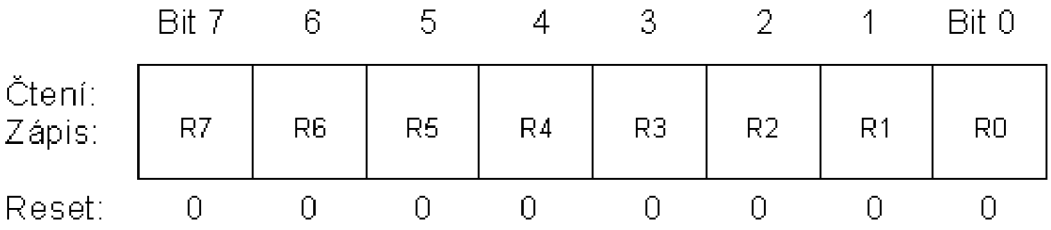
Obrázek 5.11: Datový registr odesílaných dat SU_TD

T7-T0 – Datové bity k odeslání

Pro bezchybnou fukнci sUART do registru zapisovat pouze při nastaveném bitu TE v SU_SR1. Devátý bit pro devítibitový přenos se nachází v registru SU_CR. Při nastaveném bitu PEN se při odesílání nebere ohled na hodnotu bitu T7 při osmibitovém režimu, resp. bitu T8 při devítibitovém režimu. Po zápisu dat je třeba vynulovat bit TE v SU_SR1. Po resetu jsou bity T7-T0 vynulovány.

5.4.3 SU_RD (datový registr pro přijímaná data)

Do tohoto registru zapisuje přijímač přijatá data. Devátý bit viz. řídicí registr SU_CR.



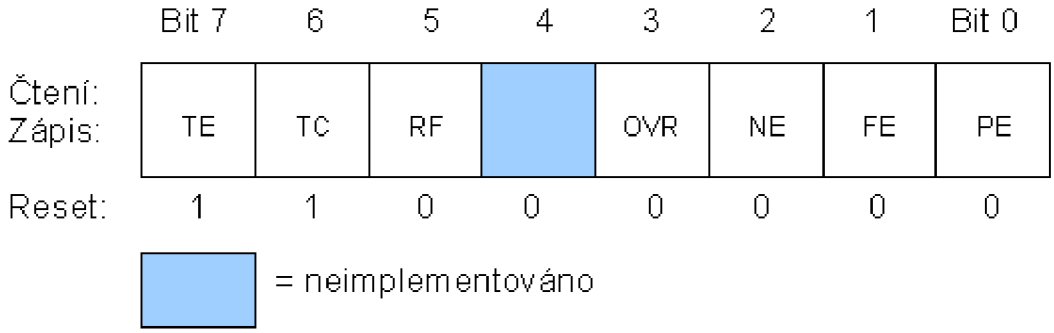
Obrázek 5.12: Datový registr pro přijatá data SU_RD

R7-R0 – Přijaté datové bity

Přijetí nových dat je indikováno bitem RF registru SU_SR1. Devátý bit devítibitového přenosu R8 se nachází v registru SU_CR. Při osmibitovém režimu je R8 kopií bitu R7. Pro umožnění přijetí dalších dat je třeba po přečtení nulovat bit RF v registru SU_SR1. Po resetu jsou bity R7-R0 vynulovány.

5.4.4 SU_SR1 (stavový registr 1)

Stavový registr 1 podává informace o připravenosti vysílače a přijímače a o případných chybách vzniklých během přijímání znaku – chyba parity, chyba časování, chyba zašumění a chyba přetečení přijímače.



Obrázek 5.13: Stavový registr SU_SR1

TE – vysílač prázdný

Bit je nastavován automaticky vysílačem, uživatel jej nuluje. Je-li bit nastaven, je možné do datového registru pro odesílání zapsat nová data. Po zapsání dat je třeba tento bit vynulovat, tím se dá vysílači najevo, že jsou připravena nová data k odeslání. Po zkopírování dat do vnitřních registrů vysílače je bit nastaven, aby uživatel mohl zapsat další data např. při odesílání souvislého bloku dat. Po resetu je tento bit nastaven.

TC – vysílání dokončeno

Tento bit indikuje dokončené odesílání, je-li nastaven. Je-li vynulován, probíhá vysílání. Je třeba mít na zřeteli, že bit TC je vysílačem nulován až při zahájení vysílání. Má-li být tento bit užit pro testování dokončení odesílání, je třeba jej předtím vynulovat, aby nedošlo kvůli zmíněnému zpoždění k chybnému rozhodnutí. Po resetu je tento bit nastaven

RF – přijámač zaplněn

Je-li bit nastaven, jsou v registru SU_RD přijata nová data. Bit je automaticky nastavován přijímačem, pro bezchybné přijetí následujícího znaku jej musí uživatel po přečtení dat vynulovat, aby nedošlo k chybě OVR - přetečení přijímače. Po resteu je tento bit vynulován.

OVR – přetečení přijímače

Bit je automaticky nastavován přijímačem, pokud je přijat nový znak a starý ještě nebyl z registru SU_RD vyzvednut, resp. Nebyl vynulován příznak RF. Nastane-li situace, že je přijat nový znak a současně je nastaven příznak RF, je nově přijatý znak ztracen a dojde k nastavení příznaku OVR. V takovém případě přijetí nového znaku nemá žádný vliv na datový registr SU_RD a bit T8 v registru SU_CR. Je-li bit nastaven, je třeba jej uživatelsky nulovat. Po resetu je bit vynulován.

NE – chyba zašumění

Bit je automaticky nastaven, liší-li se vzorky úrovně na lince přijímače v rámci jednoho bitového intervalu. Je-li bit nastaven, je třeba jej uživatelsky nulovat. Po resetu je bit vynulován.

FE – chyba časování

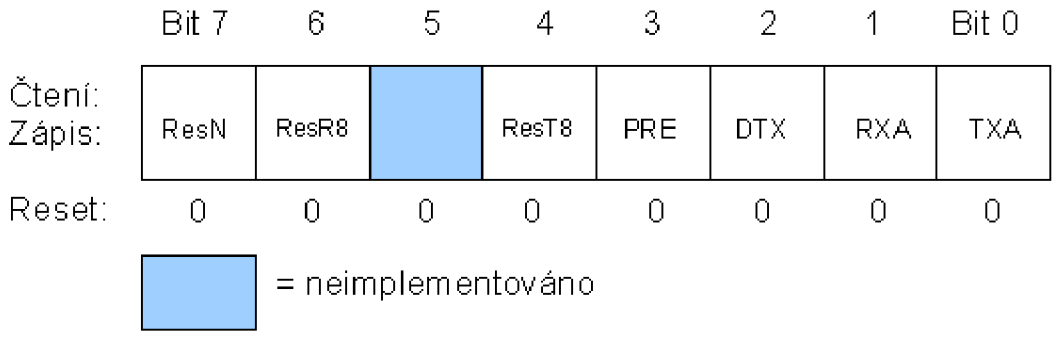
Bit je automaticky nastaven, pokud je úroveň linky v rámci bitového intervalu pro stopbit vyhodnocena jako úroveň 0. Je-li bit nastaven, je třeba jej uživatelsky nulovat. Po resetu je bit vynulován.

PE – chyba parity

Bit je automaticky nastaven, neodpovídá-li paritní bit přijatým datům a nastavené paritě. Je-li bit nastaven, je třeba jej uživatelsky nulovat. Po resetu je bit vynulován.

5.4.5 SU_SR2 (stavový registr 2)

Stavový registr 2 nemá z programovacího hlediska význam, slouží pouze indikaci příznaků, zejména pro vnitřní potřeby sUART. Je zde popsán pouze pro úplnost. Zápis do tohoto registru může způsobit nesprávnou funkci sUART.



Obrázek 5.14: Stavový registr SU_SR2

ResN – záložní bit zašumění

Bit slouží pro dočasné uložení příznaku zašumění během přijímání znaku do okamžiku, kdy se kopírují nově přijatá data do registru SU_RD, pak se kopíruje bit ResN do bitu NE v registru SU_SR1. Pokud dojde k přetečení přijímače (příznak OVR), původní hodnota bitu NE v registru SU_SR1 zůstane zachována, aby se zabránilo případnému zneplatnění bezchybně přijatých předchozích, dosud nevyzvednutých, dat.

ResR8 – záložní datový bit R8

Bit slouží pro dočasné uložení devátého bitu přijímaných dat do okamžiku, kdy se kopírují nově přijatá data do registru SU_RD, pak se kopíruje do bitu R8 v registru SU_CR. Je-li nastaven bit OVR, zůstane hodnota bitu R8 zachována.

ResT8 – záložní datový bit T8

Bit slouží pro dočasné uložení devátého bitu odesílaných dat do okamžiku jeho odeslání.

PRE – bit „předmluvy“

Tento bit slouží k indikaci právě probíhajícího odesílání bloku samých log. 1 před odesláním vlastního znaku. Předmluva se odesílá vždy po přerušení činnosti vysílače přijímačem a vždy na začátku přenosu. Probíhá-li vysílání souvislého bloku dat, předmluva se mezi jednotlivými odesílanými znaky nevysílá.

DTX – odložené vysílání

Bit slouží k indikaci přerušeného vysílání přijímačem. V takovém případě se po skončení činnosti přijímače opakuje vysílání posledně vysílaného znaku.

RXA – přijímač aktivní

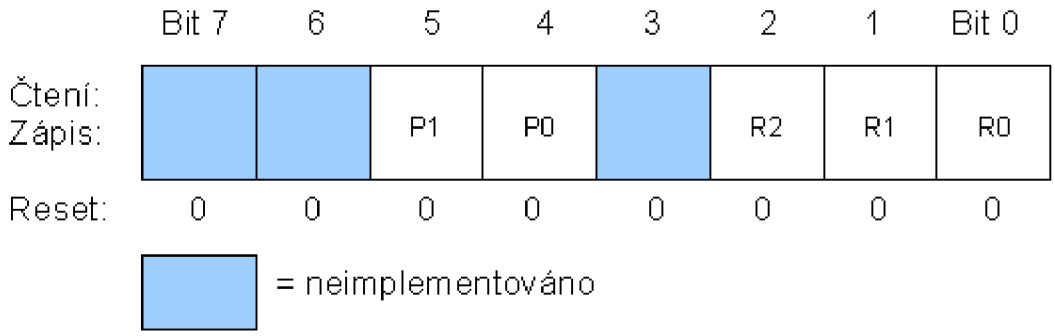
Je-li bit v 1, je přijímač aktivní a probíhá příjem znaku. Je-li bit v 0, přijámač aktivní není.

TXA – vysílač aktivní

Je-li bit v 1, je vysílač aktivní a probíhá vysílání znaku nebo vysílač čeká na dokončení činnosti přijímače. Je-li bit v 0, vysílač aktivní není.

5.4.6 SU_BR (registr přenosové rychlosti)

Pomocí tohoto registru je možné nastavovat přenosovou rychlost pomocí volby hodnot děličky a předděličky.



Obrázek 5.15: Registr přenosové rychlostu SU_BR

P1 a P0 – bity předděličky přenosové rychlosti

Tyto bity nastavují hodnotu virtuální předděličky tak, jak je ukázáno v tabulce 5.2. Reset nuluje P1 a P0.

P1 a P0	Hodnota předděličky (PD)
00	2
01	3
10	neimplementováno
11	neimplementováno

Tabulka 5.2: Bity P1 a P0 registru SU_BR

R2-R0 - bity výběru přenosové rychlosti

Těmito bity se volí dělicí hodnota přenosové rychlosti tak, jak je ukázáno v tabulce 5.3. Reset nuluje bity R2-R0.

R2, R1 a R0	Hodnota děličky (BD)
000	2
001	4
010	8
011	16
100	32
101	64
110	128
111	256

Tabulka 5.3: Bity R2, R1 a R0 registru SU_BR

K výpočtu přenosové rychlosti slouží tento vzorec:

$$přenosová\ rychlost = \frac{3 \times f_{bus}}{125 \times PD \times BD}$$

kde:

f_{bus} = frekvence sběrnice - v případě mikrokontrolérů řady QT, QY je to 3,2MHz*

PD = hodnota předděličky

BD = hodnota děličky

*bez použití vnějšího oscilátoru

Tento registr je navenek zcela shodný s registrem SCBR u modulu SCI. Rozdílné jsou konstanty děličky a předděličky. Odlišný vzorec pro výpočet přenosové rychlosti je třeba použít k zachování obvyklých přenosových rychlostí užívaných nejen modulem SCI při jiné frekvenci systémové sběrnice.

V tabulce 5.4 je možno vyčíst výsledné přenosové rychlosti při frekvenci hodin sběrnice 3,2MHz.

P1 a P0	Hodnota předděličky	R2, R1 a R0	Hodnota děličky	Přenosová rychlost (baud)
00	2	000	2	19200
00	2	001	4	9600
00	2	010	8	4800
00	2	011	16	2400
00	2	100	32	1200
00	2	101	64	600
00	2	110	128	300
00	2	111	256	150
01	3	000	2	12800
01	3	001	4	6400
01	3	010	8	3200
01	3	011	16	1600
01	3	100	32	800
01	3	101	64	400
01	3	110	128	200
01	3	111	256	100

Tabulka 5.4: Přenosové rychlosti

5.5 Omezení

Při realizaci a testování jsem narazil na jedno omezení, které se mi nepodařilo žádným způsobem odstranit. Vzhledem k časové náročnosti vykonávání obslužné rutiny po dokončení příjmu znaku vzniklo omezení příjmu bloku dat při rychlosti 19200bd. Řešení jak překonat toto omezení, kromě snížení přenosové rychlosti, je několik. První možností je nastavení vysílací aplikace, aby vysílala znaky po jednom, případně čekala na potvrzení nebo ozvěnu mikrokontroléru, který by tím potvrdil připravenost k přijetí dalšího znaku. Druhou možností je nastavení přenosu se dvěma stopbity. Další možností je připojení vnějšího oscilátoru. Díky vyššímu kmitočtu by bylo možné během stejného časového úseku vykonat větší počet instrukcí. Toto však považuji za krajní a nepravděpodobné řešení, protože by v takovém případě zůstalo málo volných pinů mikrokontroléru, u modelu QT pouze dva.

6 Závěr

Na základě analýzy současného stavu jsem navrhl a implementoval rozhraní sUART. Za vlastní přínos považuji skutečnost, že vytvořené řešení umožňuje pohodlné ovládání pomocí registrů, podobně, jako tomu je u rozhraní SCI u vyšších řad mikrokontrolérů rodiny HC08. Vlastní řešení odstraňuje nedostatky již existujících řešení. Do rychlosti 9600bd rozhraní umožňuje bezproblémovou komunikaci i při kontinuálním přenosu, při rychlosti 19200bd se vyskytuje omezení popsané v kapitole 5.5.

Uplatnění sUART je možné v aplikacích, kde jsou kladeny velké nároky na rozměry výsledného výrobku, tedy v aplikacích, kde lze s výhodou využít malých rozměrů mikrokontrolérů řady NITRON. Možnými příklady takové aplikace jsou např. automatická čidla s vlastní logikou, která by pomocí sériové linky předávala získané hodnoty nadřazenému prvku, nebo akční členy, které by se chovaly podle hodnot vstupů sériové linky. Výhodou těchto prvků je nízký počet přívodních vodičů. Např. u automatických čidel bez zpětné vazby využitých v automobilu by stačily pouze dva vodiče – jeden napájecí a jeden datový, zem je společná pro všechna zařízení vedená celým skeletem automobilu. Tím se dostávám k možnému pokračování vývoje a to využití kontrolérů NITRON jako programovatelné řídící jednotky jednoduchých zařízení v dopravních prostředcích. Například řídící jednotka zapalování motocyklu s programovatelnou (pomocí sUART) křivkou průběhu předstihu zážehu nebo jako jádro elektronického tachometru s programovatelnou hodnotou obvodu kola a podobně.

Literatura

[1] *MC68HC908GP20/D, Advance information.* Ver. Rev 1.0. Motorola, Inc. 9/1998. 406 s.

[2] VÁŇA, Vladimír: *Začínáme pracovat s mikrokontroléry MOTOROLA HC08 NITRON.* 1. vyd. Praha: BEN – technická literatura, 2003. 96 s. + CD ROM. ISBN 80-7300-124-1.

[3] *MC68HC908QY4/D, Data sheet*, Ver. Initial release, Motorola, Inc. 9/2002. 230 s.

[4] ROZEHNAL, Zdeněk: *Mikrokontroléry Motorola HC11.* 1. vyd. Praha: BEN – technická literatura, 2002. 192 s. ISBN 80-86056-77-5.

[5] AXELSON, Jan: *Serial Port Complete: Programming and Circuits for RS-232 and RS-485 Links and Networks*, Lakeview Research, 1998. 306 s. + FDD. ISBN 9780965081924.

Seznam příloh

Příloha 1. CD s kompletními zdrojovými texty, elektronickou verzí bakalářské práce a terminálovým programem.