

UNIVERZITA PALACKÉHO V OLOMOUCI

PŘÍRODOVĚDECKÁ FAKULTA
KATEDRA OPTIKY A LABORATOŘ KVANTOVÉ OPTIKY



**Konstrukce zařízení pro čítání impulsů z jednofotonového detektoru
prostřednictvím počítače**

**Realization of the Device for Counting Impulses from Single – photon Detectors
Using a Computer**

Diplomová práce

Jiří Hobza

2009

UNIVERZITA PALACKÉHO V OLOMOUCI

PŘÍRODOVĚDECKÁ FAKULTA

KATEDRA OPTIKY A LABORATOŘ KVANTOVÉ OPTIKY



**Konstrukce zařízení pro čítání impulsů z jednofotonového detektoru
prostřednictvím počítače**

Diplomová práce

Vypracoval:

Bc. Jiří Hobza

Vedoucí diplomové práce:

Prof. RNDr. Miloslav Dušek, Dr.

Studijní obor:

Optika a optoelektronika

Forma studia:

Prezenční

Datum odevzdání práce:

Poděkování:

Tímto bych chtěl poděkovat především vedoucímu práce Prof. RNDr. Miloslavu Duškovi, Dr. za vynikající vedení, bez kterého by tato diplomová práce nikdy nevznikla. Dále za ochotu, s jakou mi vyšel vstříc při výběru tématu práce a za čas, který mi obětavě věnoval při řešení souvisejících problémů.

Také děkuji RNDr. Pavlu Krchňákovi, Ph.D. za zapůjčení programátoru a za možnost pracovat v Laboratoři pro elektřinu a magnetismus.

V neposlední řadě také děkuji mé rodině za veškerou podporu během studia.

Prohlášení:

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně za pomoci vedoucího práce a uvedené literatury. Souhlasím s použitím této práce pro potřeby Katedry Optiky a Laboratoře Kvantové optiky.

V Olomouci, dne _____

Podpis _____

Bibliografická identifikace

Autor: Bc. Jiří Hobza

Název práce: Konstrukce zařízení pro čítání impulsů z jednofotonového detektoru prostřednictvím počítače

Typ práce: diplomová práce

Katedra: Katedra optiky a Laboratoř kvantové optiky

Vedoucí práce: Prof. RNDr. Miloslav Dušek, Dr.

Rok obhajoby: 2010

Abstrakt

Práce se věnuje konstrukci zařízení pro čítání impulsů z jednofotonového detektoru prostřednictvím počítače. V teoretické části nejprve zmiňuje motivaci na konstrukci čítače impulsů a také parametry, které by měl splňovat. Na základě těchto parametrů pak práce navrhuje blokové schéma a konkrétní součástky. V dalších kapitolách se věnuje shromáždění údajů z katalogových listů a další literatury, potřebných ke konstrukci čítače.

V praktické části již navrhuje a popisuje konkrétní schéma zapojení. Dále čtenáře seznamuje s vývojovým prostředím ATMEL AVR Studio a se strukturou souboru formátu Intel HEX. Popisuje hlavní myšlenku programu pro mikrokontrolér a také program samotný. V dalších kapitolách odůvodňuje použití integrovaného vývojového prostředí Visual Basic 6.0 pro tvorbu doprovodných počítačových programů. Následně je popsána funkce těchto programů, spolu s nejdůležitějšími částmi zdrojového kódu.

Klíčová slova: konstrukce, čítač impulsů, mikrokontrolér, program.

Bibliographical identification

Author: Bc. Jiří Hobza

Title: Realization of the Device for Counting Impulses from Single – photon Detectors Using a Computer

Type of thesis: Diploma thesis

Department: Department of Optics and Laboratory for Quantum Optics

Supervisor: Prof. RNDr. Miloslav Dušek, Dr.

Year of presentation: 2010

Abstract

The thesis deals with construction of a device for counting impulses from single– photon detectors using a computer. The theoretical part of the thesis partially discusses the benefits of constructing such a device and partially its parameters. The thesis suggests block diagrams and specific electronic parts for the construction based on the finds. The following chapters present data from data sheets and other technical literature needed for the project.

The practical part of the thesis deals with the design and description of the actual circuit diagram of the device. The thesis also include a closer description of the development of the AVR Studio environment by ATMEL and the structure of the Intel HEX file. The main concept of the microcontroller program and the program itself are described. In the following chapter the use of Visual Basic 6.0 for developing supporting software is presented. Finally, the thesis deals with the most important source code parts and how to use these programs.

Keywords: construction, impuls counting, microcontroller, program.

Obsah

1. Obsah	7
2. Úvod	9
3. Teoretická část	10
3.1. Motivace a požadavky na zařízení	10
3.2. Blokové schéma čítače impulsů	10
3.3. Výběr konkrétních součástí	11
3.4. Způsob programování MCU	12
3.5. Sériový port a RS 232	14
3.6. Port USB	17
3.7. Mikrokontrolér ATtiny2313	19
3.7.1. Architektura ATtiny2313	19
3.7.2. Organizace paměti ATtiny2313	21
3.7.3. Hodinový signál ATtiny2313, jeho zdroje a nastavení	21
3.7.4. Jednotka USART	23
3.7.4.1. Blokové schéma USART	24
3.7.4.2. Registr UCSRA.....	25
3.7.4.3. Registr UCSRB.....	25
3.7.4.4. Registr UCSRC.....	26
3.7.4.5. Registr UBRR.....	28
3.7.5. Programování ATtiny2313.....	28
3.8. Integrovaný obvod FT232BM.....	31
3.8.1. Základní vlastnosti FT232BM.....	31
3.8.2. Příklady zapojení FT232BM.....	32
3.8.2.1. Připojení krystalu.....	33
3.8.2.2. Signalizační LED.....	33
3.8.2.3. Paměť EEPROM.....	34
3.8.2.4. Zapojení FT232BM pro napájení z USB.....	34
3.9. Klopný obvod 74F74.....	35
4. Praktická část	38
4.1. Schéma zapojení, návrh a výroba DPS.....	38
4.1.1. Schéma zapojení čítače impulsů.....	38
4.1.2. Návrh DPS.....	44
4.1.3. Výroba, osazení a oživení DPS.....	45
4.2. Připojení zařízení s FT232BM k počítači.....	47
4.2.1. Instalace ovladačů.....	47

4.2.2.	Programování EEPROM.....	50
4.2.3.	Odstalování ovladačů.....	52
4.3.	AVR Studio, program pro mikrokontrolér, formát Intel HEX.....	53
4.3.1.	Vývojové prostředí AVR Studio.....	53
4.3.2.	Soubory formátu Intel HEX.....	55
4.3.3.	Popis programu pro mikrokontroler.....	56
4.4.	Doprovodné programy pro počítač.....	62
4.4.1.	Integrované vývojové prostředí Visual Basic.....	63
4.4.1.1.	Zdrojový kód Counter-Signal.....	63
4.4.1.2.	Použití programu Counter-Signal.....	65
4.4.1.3.	Zdrojový kód Counter-Comm.....	67
4.4.1.4.	Použití programu Counter-Comm.....	69
4.4.1.5.	Zdrojový kód Counter-Prog.....	70
4.4.1.6.	Použití programu Counter-Prog.....	74
4.5.	Použití čítače impulsů.....	76
5.	Závěr.....	78
6.	Literatura.....	79

2 Úvod

Cílem diplomové práce je navrhnout a vyrobit zařízení, schopné čítat impulsy z jednofotonového detektoru. Navržený čítač by měl být konstrukčně jednoduchý, aby jej bylo možné snadno reprodukovat. Z tohoto důvodu by měla být práce také jakýmsi návodem na výrobu čítače a zároveň manuálem pro jeho použití. Pokud uvážíme cenu podobných komerčních přístrojů, která se pohybuje v řádu několika tisíc korun i více, byla by přijatelná pořizovací cena výhodou. Nicméně jednoduchá konstrukce a nízká cena by neměla bránit dosažení požadovaných parametrů.

Práce je rozdělena na teoretickou a praktickou část. V úvodu teoretické části zmíníme motivaci, která vedla ke vzniku práce. Rovněž uvedeme parametry, které by měl čítač impulsů splňovat. Dále se na základě požadavků kladených na čítač impulsů, pokusíme sestavit blokové schéma a vybrat konkrétní součástky. Následující kapitoly shromáždí podrobnější informace o těchto součástkách, protože je budeme potřebovat v dalších částech práci.

Praktická část začíná popisem konkrétního schématu zapojení. Následuje představení vývojového prostředí AVR Studio spolu s popisem struktury souboru formátu Intel HEX. V neposlední řadě se seznámíme s hlavní myšlenkou programu pro mikrokontrolér, včetně popisu jeho nejdůležitějších částí. Pro vytvoření programů na straně počítače, sloužilo integrované vývojové prostředí Visual Basic 6.0. Nebudeme se podrobněji zabývat jeho popisem, ale zmíníme důvody, které nás vedly k jeho použití, včetně některých aspektů programování v tomto prostředí. Poslední kapitoly práce jsou věnovány popisu doprovodných programů, včetně jejich zdrojového kódu.

3 Teoretická část

3.1 Motivace a požadavky na zařízení

V žádném optickém experimentu se neobejdeme bez detektoru. V případě experimentů s kvantovým zpracováním informace, kvantovou kryptografií, slabými laserovými pulsy či jednofotonovou interferencí v optických vláknech pracují experimentátoři s jednofotonovými detektory. Při jejich použití je obvykle třeba tyto impulsy dále zpracovávat. Jednou z těchto potřeb může být znalost jejich počtu v určitém časovém okně. Zařízení na čítání impulsů je potřebné i v laboratoři Katedry optiky PřF UP.

Z charakteru budoucího použití jsou kladeny na toto zařízení určité nároky. Vyjmenujme nejdůležitější z nich:

- schopnost reagovat na TTL impulsy od délky 5 ns při minimální frekvenci 500kHz
- připojení k sériovému portu (RS 232) nebo k portu USB

Zařízení by také mělo mít několik vstupů vybavených TTL (Transistor-Transistor Logic) logickými obvody schopných pracovat s řádově nanosekundovými impulsy a zároveň být schopné z těchto vstupů číst požadovanou rychlostí. Složitější experimenty jsou dnes většinou obsluhovány pomocí počítače, takže připojení přes sériový port je namístě. Lepší možností by představovalo rozhraní USB (Universal Serial Bus), protože má výhodu v podobě možnosti odběru napětí 5V a maximálního proudu 500mA. Tak by odpadla konstrukce zdroje nebo použití síťového napáječe.

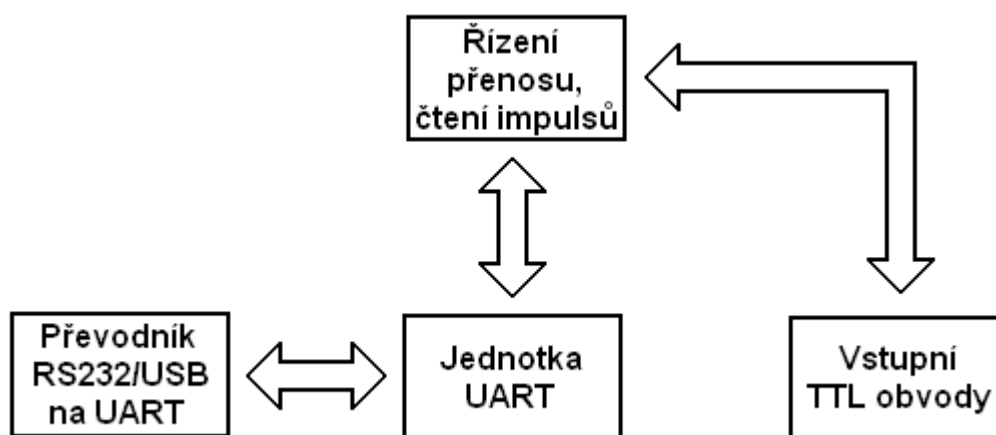
3.2 Blokové schéma čítače impulsů

Nyní tedy známe některé parametry budoucího čítače impulsů spolu s požadovaným připojením k počítači. Na základě těchto informací se teď pokusíme sestavit blokové schéma. Protože čítač bude pracovat s TTL impulsy a zároveň komunikovat s počítačem prostřednictvím sériového resp. USB portu, nevyhneme se použití příslušného převodníku úrovní. Logické úrovně pro sériovou nebo USB linku jsou totiž odlišné od úrovní pro TTL obvody. Konstrukčně výhodnější by byla varianta s USB a to z několika důvodů. Jedním z nich je skutečnost, že z portu USB lze odebírat napětí 5V a proud až 500mA. To jsou hodnoty, které jsou pro TTL logické obvody zcela dostačující. Nebylo by tedy nutné počítat se zdrojem napájení integrovaným na DPS čítače impulsů nebo s vnějším síťovým napáječem. USB je navíc univerzálnější a rychlejší než klasický sériový port.

Typické využití čítače v laboratoři může být následující. Čítač bude součástí skupiny přístrojů řízených počítačem a vykonávajících měření. Z tohoto důvodu musí být čítač schopen zpracovávat

příkazy definované obsluhou. Obsluha například vyšle požadavek na čítání impulsů po dobu jedné sekundy. Čítač požadavek přijme, vykoná jej a nakonec získaná data odešle zpět obsluze. Z toho mimo jiné vyplývá, že následující blok v našem schématu musí obsahovat univerzální přijímač a vysílač UART (Universal Asynchronous Receiver and Transmitter).

Další člen schématu by měl být přímo propojen s vysílací a přijímací jednotkou a zároveň se vstupními TTL obvody (ty musí splňovat parametry uvedené v kapitole 3.1). Tento blok by měl rozeznávat příkazy přicházející po sériové lince a čítat samotné impulsy. Všechny tyto úvahy nás vedou k obecnému schématu, jako je například to na následujícím obrázku.



Obr. 1: Obecné blokové schéma čítače impulsů.

3.3 Výběr konkrétních součástí

Známe-li obecné schéma čítače, můžeme na jeho základě vybrat konkrétní typy součástí tak, aby vyhovovaly zadaným parametrům. Nejprve je třeba najít vhodný převodník z úrovně RS-232/USB na UART. Firma FTDI Chips má dlouholetou zkušenost s různými konvertory pro sériovou linku a zároveň je prověřena českými uživateli. Pro účel práce se zdá být vhodný typ FT232BM. Jedná se o převodník USB-UART. Samotný čip umožňuje také napájení aplikace z USB a je plně kompatibilní se standardy USB 1.1 a USB 2.0. K dispozici jsou přitom všechny linky UART. FTDI Chips nabízí na svých webových stránkách volně ke stažení software pro programování rozšiřující paměti EEPROM a ovladače pro operační systémy Mac OS , Linux a Windows.

Další blok tvoří jednotka univerzálního přijímače a vysílače. Ačkoli existují integrované obvody s touto funkcí, většinou je jednotka UART zakomponována do složitějších obvodů, jako je například jednočipový mikrokontrolér (dále jen MCU). Jak napovídá název, na jednom čipu je spolu s mikroprocesorem umístěna také paměť pro program a data a vstupně/výstupní obvody.

MCU vykonává příkazy programu uloženého v programové paměti. **Použitím MCU bychom tedy dosáhli značné variability výsledné konstrukce. Pouhou změnou programu MCU lze čítači impulsů přidat další funkce nebo z něj výměnou vstupních obvodů učinit zcela jiné zařízení!** Proto je také většina praktické části věnována tvorbě doprovodného software, který slouží k programování MCU a k vývoji jiných aplikací. Velmi dobře dostupné a zároveň výkonné jsou například mikrokontroléry od firmy ATMEL. Protože základní myšlenkou řídicího programu pro MCU je opakující se čtecí smyčka (více v kapitole 2) je důležitý co největší výpočetní výkon. Ten nám poskytuje například ATtiny2313 dosahující až 20Mips při hodinovém taktu 20MHz (viz. [6]). Kromě možnosti paralelního programování podporuje ATtiny také sériové stahování dat skrze sběrnici SPI (Serial Peripheral Interface). Tuto sběrnici je navíc možné ovládat přímo linkami UART, což by jen podtrhovalo flexibilitu případné konstrukce.

Posledním blokem jsou vstupní TTL obvody. Ideálním kandidátem na tuto funkci by mohl být klopný obvod typu R-S. Ten může pracovat ve dvou režimech. V jednom lze logickou hodnotu na výstupu měnit pomocí nulovacího nebo nastavovacího vstupu. Ve druhém se logická hodnota na datovém vstupu přepíše na výstup s náběžnou, případně sestupnou hranou hodinového signálu. Tato hodnota zde zůstane do té doby, než ji vynulujeme. R-S klopné obvody 74F74 jsou kompatibilní s TTL a dokáží reagovat na impulsy od délky 4ns s opakovací frekvencí až 100MHz.

3.4 Způsob programování MCU

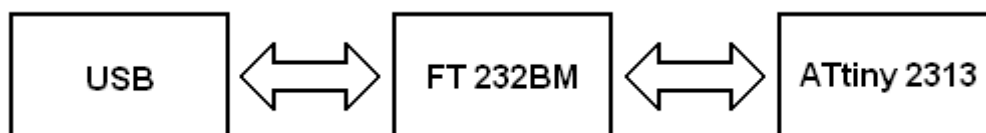
V úvahu připadá několik variant, jak naprogramovat MCU ATtiny. První varianta (viz. obr. 2) počítá s obousměrným propojením USB konvertoru a MCU. K programování touto variantou je dále možné přistupovat dvojím způsobem:

- přímým řízením linek UART,
- využitím režimu BitBang.

Při přímém řízení linek (viz. [4] a [5]) využíváme skutečnosti, že na port USB je možné se obracet jako na virtuální sériový port. Tudíž je možné linky portu USB ovládat stejně jako linky sériového portu. Sériový port patří mezi základní porty počítače a programovací jazyky jako C/C++, Delphi, Visual Basic, Visual C++ obsahují nástroje pro práci s ním. Nespornou výhodou tohoto přístupu je jeho univerzálnost a nevýhodou omezený počet vstupních a výstupních vývodů (viz. následující kapitola).

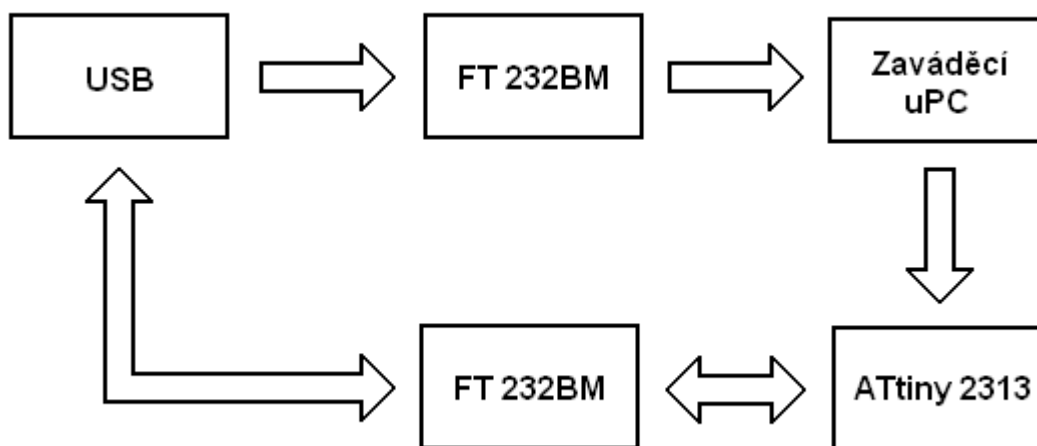
Druhou možností je využití speciálního režimu, do kterého lze obvod FT232BM přepnout (touto problematikou se podrobně zabývá [1]). Všechny linky UART se přemění na osm vstupně-výstupních vývodů a každý lze ovládat nezávisle na ostatních. Výhoda této metody spočívá ve

variabilitě, s jakou je možné určit, zda daná linka bude vstupní nebo výstupní. Ačkoli čip samotný podporuje širokou škálu přenosových rychlostí (až 3 MBaud), v režimu BitBang je programování značně pomalejší (viz. [1] - autor neuvádí přesnou hodnotu, ale zmiňuje značné zpomalení přenosu dat při programování ATMEL AT90S2313 touto metodou). Také je třeba podotknout, že v tomto módu přicházíme o sériový port.



Obr. 2: Schéma využívající přímého řízení linek a režimu BitBang

Další variantou jak ATtiny naprogramovat je použití zaváděcího MCU (viz. [1] nebo [5]). Ten by zde sloužil jako převodník signálů UART na signály pro sériové programování (viz. obr. 3). Tím získáme vyšší rychlost přenosu dat, ale pouze jedním směrem - do cílového MCU. Ze zadání této práce však plyne důležitost obousměrné komunikace, a proto by bylo nutné přidat další obvod FT232BM. Tím by však vzrostla složitost zapojení a zároveň cena. Navíc bychom potřebovali zároveň programátor pro zaváděcí MCU. To by znamenalo další konstrukci nebo koupi zařízení.

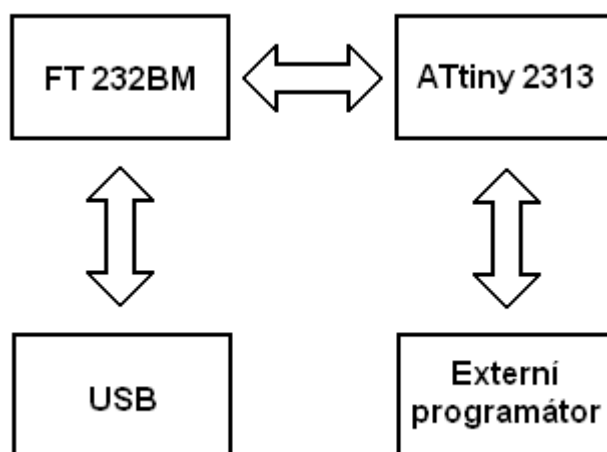


Obr. 3: Využití zaváděcího MCU jako převodníku UART na SPI

Jistou možností je také použití programového zavaděče (tzv. bootloaderu) (zmínka v [5] a [6]). Je to program pro mikrokontrolér, který se uloží na začátek programové paměti. Úkolem zavaděče je testovat přijímač sériové linky. Pokud rozezná požadavek na programování (definovaný uživatelem), začne číst příchozí data a zapisovat je do paměti. Výhodou tohoto způsobu je, že

nepotřebujeme další zaváděcí MCU nebo paralelní programátor. Ovšem programový zavaděč je vázaný na daný typ mikrokontroléru a pro nahrání zavaděče na začátek programové paměti bychom stejně potřebovali nějaký programátor.

Poslední variantou je programování externím zařízením (viz. obr. 4) a to buď přímo v aplikaci, což je výhodnější nebo způsobem, kdy se vyjme MCU z patice na desce a vsadí se do programátoru. To je ale poněkud nepraktické za předpokladu, že ve fázi testování bude třeba měnit program často. Výhodou je, že takové programátory pracují většinou v paralelním režimu, který je propracovanější než sériový. Mnohdy také podporují více typů mikrokontrolérů nebo další druhy čipů (např. paměti).



Obr. 4: Použití externího programátoru

Uvedené varianty mají své výhody i nevýhody. Při výběru řešení pro konstrukci čítače impulsů jsme se řídili několika zásadami. Vybraný způsob programování by měl být kompromisem mezi náklady na výrobu, složitostí konstrukce a náročností kódu aplikace pro OS Windows, která bude sloužit k demonstraci sériového programování MCU. Pokud by navíc byla konstrukce univerzální v tom smyslu, že by jí bylo možné snadno rozšířit na jiné typy MCU, znamenalo by to velké plus. Po pečlivém zvážení všech kladů a záporů jsme vybrali první variantu (viz. obr. 2), která bude využívat přímého řízení linek sériového nebo USB portu.

3.5 Sériový port a RS 232

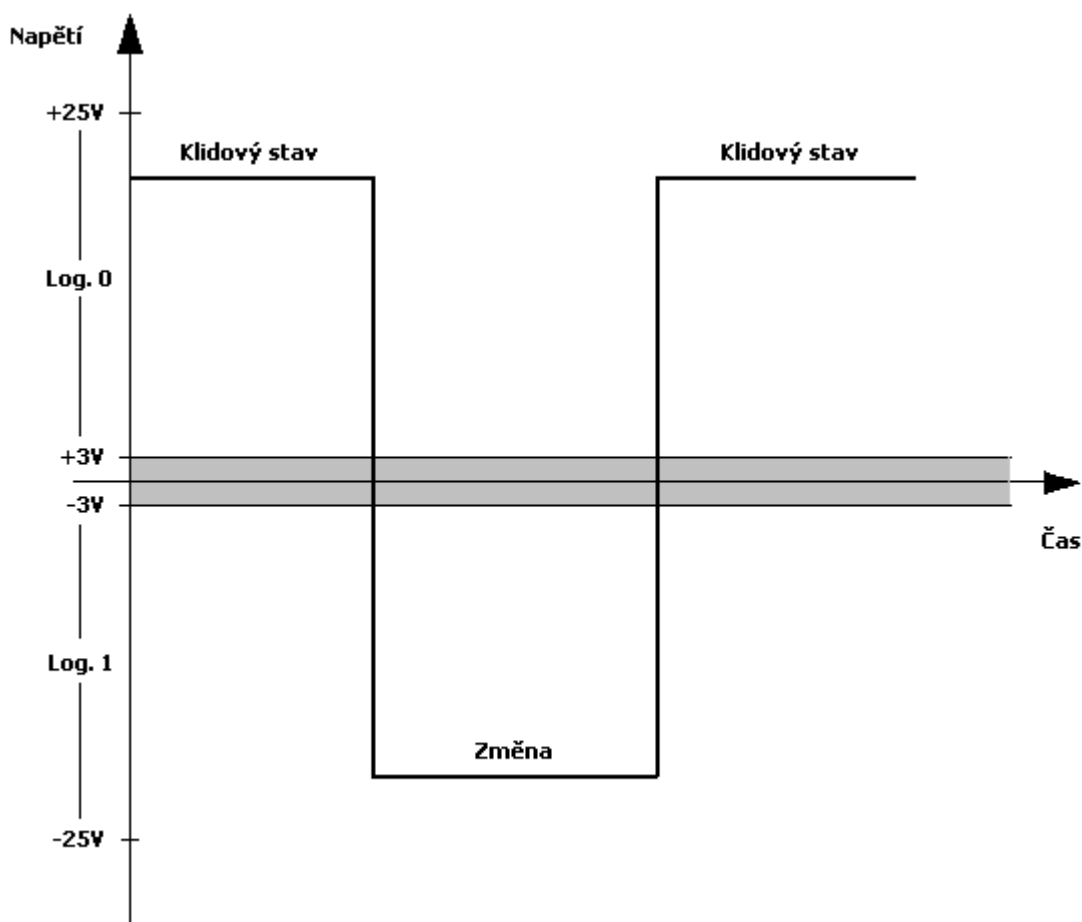
Sériový port (často označován jménem původního standardu RS 232 z počátku 60 let minulého století) je již řadu let součástí osobních počítačů. I když je postupně vytlačován modernějším USB, stále ještě je to zajímavé rozhraní pro spoustu uživatelských aplikací, a to zejména pro svou

jednoduchost, dostupnost a univerzálnost. Toto rozhraní bylo původně navrženo pro přenos informací mezi dvěma zařízeními (modemy) do vzdálenosti maximálně 20 m. Pro větší odolnost proti rušení je informace přenášena po propojovacích vodičích větším napětím než je standardních 5 V a pro fyzické připojení propojovacího kabelu se používá konektor Cannon 9.



Obr. 5: Schematická značka a ukázka v záslepce PC konektoru Cannon 9

Standart RS 232 používá dvě logické úrovně: logickou nulu a jedničku. Hodnotu logické nuly představuje kladná úroveň napětí na výstupních vodičích, zatímco logickou jedničku záporná. Jednotlivé úrovně mohou dosahovat hodnot napětí až 25V a to jak kladných tak záporných (viz. obr. 6). Tímto se docílilo menšího rušení přenášeného signálu.



Obr. 6: Změna napětí v závislosti na čase pro signály RS 232

K propojení s TTL logikou (používá jiné napěťové úrovně) je však třeba použít převodník. Nejznámějším z nich je Maxim MAX232N (příklad použití například ve [4]). V tabulce 1 jsou rozepsána napětí pro logické úrovně datových signálů RS 232.

Datové signály		
Úroveň	Přijímač	Vysílač
Log. 0	+3V až +25V	+5V až +15V
Log. 1	-3V až -25V	-5V až -15V
Nedefinované	-3V až +3V	

Tab. 1: Hodnoty napětí pro jednotlivé logické úrovně signálů RS 232

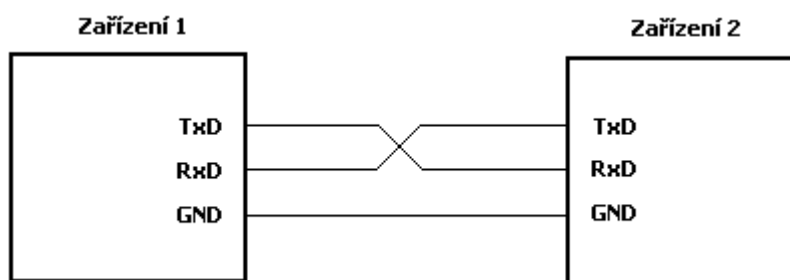
Jednotlivé signály sériového portu mají samozřejmě také svůj název, který je odvozen z jejich účelu. Následující tabulka obsahuje jejich přehled.

Vývod Cannon 9	Název	Směr	Popis
1	DCD	vstup	Data Carrier Detect
2	RXD	vstup	Receive Data
3	TXD	výstup	Transmit Data
4	DTR	výstup	Data Terminal Ready
5	GND	-	Ground
6	DSR	vstup	Data Set Ready
7	RTS	výstup	Request to Send
8	CTS	vstup	Clear to Send
9	RI	vstup	Ring Indicator

Tab. 2: Popis signálů RS 232

Při základní komunikaci si často vystačíme pouze se třemi signály a to RXD, TXD a GND (viz. obr. 7). Ostatní signály se používají pro kontrolu přenosu dat mezi jednotlivými zařízeními (modemy). To bývá také označováno jako řízení toku dat neboli handshaking. Při dnešních plně duplexních systémech částečně ztrácejí svůj smysl. Nicméně jich lze využít v uživatelských aplikacích. Na základě zkušeností získaných při vývoji čítače impulsů, však doporučujeme vybírat

tyto signály s opatrností. Některé se totiž vzájemně ovlivňují (z důvodu jejich primárního účelu, tj. řízení přenosu informací mezi modemy) a to může způsobit nežádoucí chování aplikace!

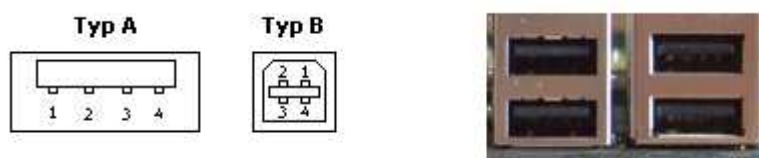


Obr. 7: Propojení dvou zařízení pomocí třech vodičů

3.6 Port USB

Port USB (Universal Serial Bus) je stejně jako port RS 232 sériové rozhraní počítače. Jeho výhodami oproti sériovému portu jsou vysoké přenosové rychlosti (až 480Mb/s), možnost napájet zařízení přímo z konektoru (běžně napětí +5V a proud až 500mA) a velký počet připojitelných zařízení (až 127 při použití rozbočovače). Obvykle se také setkáváme s označením USB 1.1 nebo USB 2.0 (dnes dokonce i USB 3.0). Jedná se o jednotlivé standardy, které se liší právě podporovanými přenosovými rychlostmi.

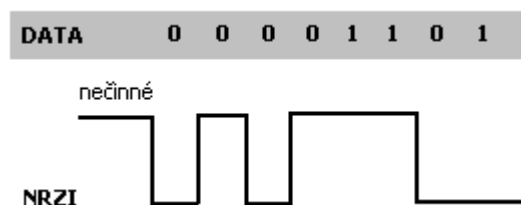
K připojení zařízení se používají USB konektory (viz. obr. 8) pouze se čtyřmi vodiči. Pořadí vodičů je patrné z následujícího obrázku a je závazné při čelním pohledu na konektor. První vodič je zdrojem napětí +5V. Druhý je vyhrazen pro přímá data (Data+) a třetí vodič pro negovaná data (Data-). Poslední vodič je zemnicí (GND).



Obr. 8: Konektory USB typu A/B a ukázka konektorů typu A na základní desce počítače

Co se týče přenosu dat, je USB jednomasterová sběrnice. To znamená, že všechny aktivity přicházejí z počítače. Data se vysílají v paketech délky 8 až 64 (1024 pro izochronní přenos) bajtů a jejich přenos se uskutečňuje v rámcích o délce 1ms. Pokud je k počítači připojeno více zařízení obsahují rámce pakety pro několik zařízení, která mohou pracovat i s různými přenosovými rychlostmi. Rozdělení paketů pak zajistí rozbočovač. Podřízené zařízení se pak synchronizuje s datovým tokem a získat z něj hodinový signál. Ten není přenášen po zvláštním vodiči, ale je za-

kódován přímo do datového toku. K tomu se používá metoda NRZI (Non Return To Zero - logická nula vede k změně úrovně a logická jednička ponechává úroveň beze změny). Kódování a dekódování je pak čistě záležitostí přijímače resp. vysílače. K zamezení ztrátě synchronizace se používá metoda vsouvání bitů (pokud se v datovém proudu vyskytne šest po sobě jdoucích logických jedniček, přidá vysílač logickou nulu, aby vynutil změnu úrovně) a synchronizační bajt na konci každého paketu.



Obr. 9: Kódování metodou NRZI

Protože USB také podporuje Plug&Play, musí být každé zařízení připojené k počítači automaticky rozpoznáno operačním systémem. Aby se tak stalo, musí si počítač od zařízení vyžádat jeho parametry. Souhrn těchto parametrů nazýváme deskriptorem zařízení a procesu jejich vyžádání enumerací. Rozbočovač pozná připojení nového zařízení tak, že dojde ke zdvihnutí linky Data+ nebo Data-. Enumerace pak proběhne v následujících krocích:

- 1) Hub informuje hostitelský počítač (host), že je připojeno nové zařízení.
- 2) Host se dotáže hubu, na který port je zařízení připojeno a vydá příkaz tento port aktivovat a provést reset USB sběrnice.
- 3) Rozbočovač vyvolá reset USB o délce 10 ms a uvolní pro zařízení proud 100 mA. Zároveň proběhne reset zařízení a tím je připraveno.
- 4) Než zařízení obdrží vlastní sběrniceovou adresu, je možno se na něj obracet přes implicitní adresu 0. Aby host stanovil délku datových paketů, čte první bajty deskriptoru zařízení.
- 5) Pak host zařízení přiřadí novou sběrniceovou adresu a s její pomocí načte všechny informace obsažené v deskriptoru zařízení.
- 6) Nakonec host přiřadí zařízení jednu z možných konfigurací. Zařízení pak může odebírat tolik proudu, kolik je stanoveno v aktivovaném konfiguračním deskriptoru a tím je tedy připraveno k použití.

Enumerace je děj, který provádí OS zcela samostatně. Uživatel pouze vybere příslušné ovladače, pokud je OS neobsahuje. Přehled nejdůležitějších parametrů zařízení uvádí následující tabulka.

Položka	Význam
VID	číselný identifikátor prodejce (16bitové číslo přidělované organizac USB)
PID	číselný identifikátor výrobku (16bitové číslo určené výrobcem)
Manufacturer ID	řetězec identifikující výrobce
Manufacturer	řetězec popisující výrobce
Product	řetězec popisující výrobek
Serial Number	řetězec sériového čísla (umožní připojit několik stejných výrobků)
Počet konfigurací	počet konfiguračních deskriptorů, které se například liší odběrem

Tab. 3: Přehled nejdůležitější deskriptorů zařízení

3.7 Mikrokontrolér ATtiny 2313

V této kapitole se budeme podrobně zabývat popisem, ale především nastavením a programováním mikrokontroléru ATtiny2313.

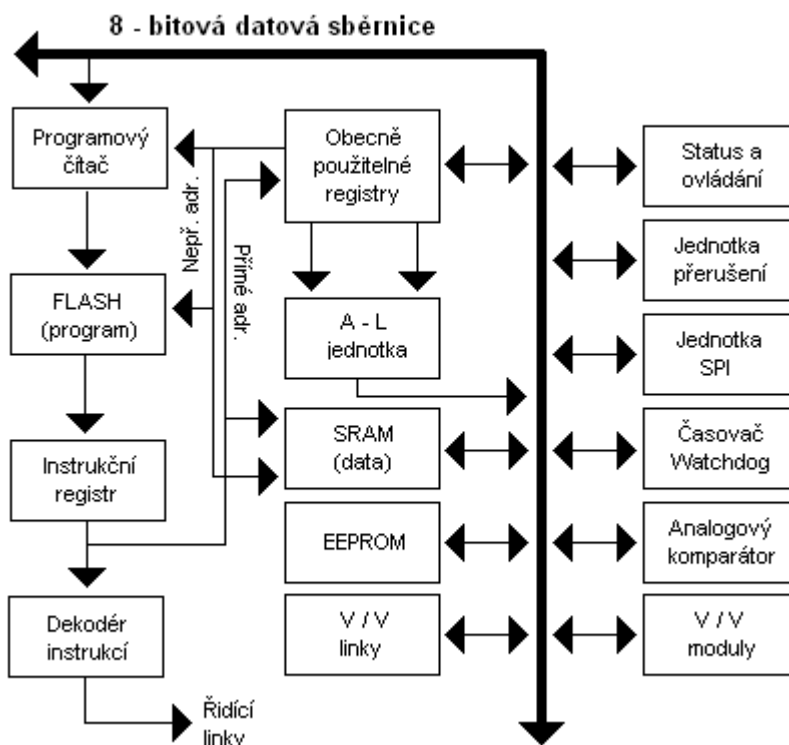
3.7.1 Architektura ATtiny 2313

Existují dvě základní architektury CPU (Central Processor Unit):

- von Neumannova
- Harvardská

První dostala své jméno po maďarském matematikovi žijícím v USA Johnu von Neumannovi. Tuto koncepci navrhl pro počítačový stroj s uloženým programem již v roce 1945. Její hlavní charakteristikou je to, že paměť je společná pro data i program. Princip společné paměti je slabinou této architektury, protože rychlost komunikace s pamětí je tak omezená. Tento neduh odstraňuje

koncepte Harvardská. Zde je paměť datová a programová oddělena a mohou být dokonce různého typu a délky slova (viz. [2]). Mikroprocesor, paměti a vstupně - výstupní obvody jsou integrované na jednom čipu. Pokud má takový čip vyvedou vnitřní sběrnici na vývody (umožňuje např. připojit další paměť), mluvíme o tzv. jednočipových mikropočítačích. V řadě aplikací však požadujeme co nejnižší cenu a vystačíme si s jednočipovým mikrořadičem bez rozšiřující paměti. Tímto způsobem se sníží počet vývodů a pak mluvíme o jednočipových řadičích nebo častěji o mikrokontrolerách – zkráceně MCU. Více o vývoji těchto architektur a o jádrech mikrokontrolerů řady ATMEL AVR například v [2].

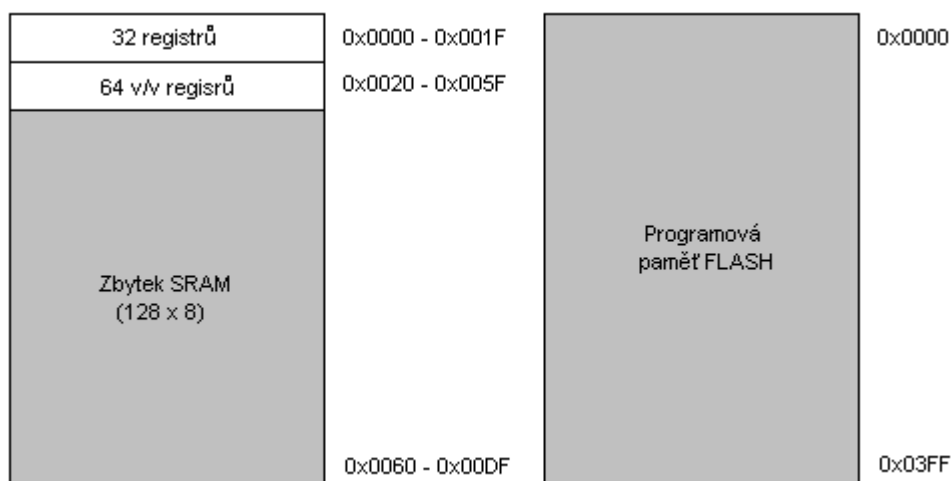


Obr. 10: Jádro mikroprocesoru ATtiny2313 (přeloženo a převzato [6])

Základem osmibitových mikrořadičů ATMEL ATtiny je vylepšená verze Harvardské architektury a nese název RISC (Reduced Instruction Set Computer). Přednostmi tohoto řešení jsou jednocyklové instrukce, vyšší taktovací frekvence spojené s vyšším výpočetním výkonem, stejně jako efektivní optimalizace překladu instrukcí do strojového kódu. Pro účely této práce není nutné popisovat jednotlivé bloky z nichž se jádro ATtiny2313 skládá. Pro bližší představu o jeho uspořádání tak poslouží pouze obrázek. Více informací o jednotlivých blocích jádra mikroprocesoru ATtiny2313 lze nalézt především v [6].

3.7.2 Organizace paměti ATtiny 2313

Jak již bylo řečeno mikrokontrolér ATtiny2313 disponuje oddělenou pamětí pro program a data. Programová paměť FLASH má velikost 2 kB. Datová paměť je typu SRAM a velikost její části vyhrazené pro data uživatele je 128 B. Prvních 32 bajtů je totiž vyhrazeno pro registry R0 až R31. Tyto registry je možné až na několik výjimek používat libovolně. Výjimky tvoří instrukce andi, cpi, ldi, ori, sbci a subi, které lze používat pouze s registry R16 až R31. Následuje 64 vstupně výstupních registrů a nakonec 128 bajtů volné datové paměti (viz. obr. 11). Kromě toho disponuje ATtiny2313 také pamětí typu EEPROM. Celková velikost této paměti je 128 bajtů a je organizována jako oddělená datová paměť. Je možných 100 000 zápisů nebo čtení a tyto operace lze provádět s každým jednotlivým bajtem. Pro přístup k ní se používají speciální registry adresové (EEAR), datové (EEDR) a řídicí (EECR). O významu jednotlivých bitů více například v [6].

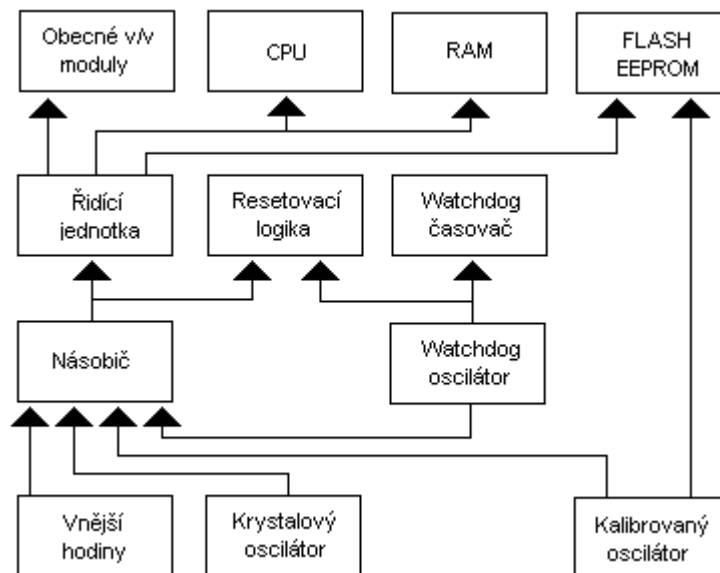


Obr. 11: Organizace paměti mikroprocesoru ATtiny2313 (přeloženo a převzato z [6])

3.7.3 Hodinový signál ATtiny2313, jeho zdroje a nastavení

Ještě než přistoupíme k samotnému popisu nastavení jednotky USART, je důležité nejprve objasnit zdroje a možnosti nastavení hodinového signálu (dále jen hodin). Důležitost hodin je zřejmá z obrázku 12. Ten ukazuje, jak je signál ze zdroje hodin veden do násobiče, který připojí vybraný zdroj k řídicí jednotce. Ta jej přerozdělí do pamětí, jádra (tzn. obecně použitelné registry, ukazatel zásobníku spolu s datovou pamětí, kterou používá a stavový registr) a periferních obvodů jako jsou např. vstupně - výstupní obvody a jednotka USART.

U ATtiny existují celkem tři možné zdroje hodinového signálu. Vnější zdroj hodin, krystalový oscilátor a do třetice vnitřní kalibrovaný RC oscilátor. Protože zdrojů hodin je několik, má uživatel



Obr. 12: Distribuce hodin mikrokontrolérem ATtiny2313 (přeloženo a převzato z [6])

možnost výběru nastavením programovatelných pojistek CKSEL 0 až CKSEL 3 (viz. tab. 4) v paměti Flash. Navíc lze pojistkami SUT 0, SUT 1 definovat startovací a čekací interval. V prvním případě je to čas, za který se ustálí kmity oscilátoru před vykonáním první instrukce. Ve druhém představuje dobu potřebnou pro stabilizaci spotřeby před vstupem do normálního režimu (viz. tab. 5). Tyto doby mají význam tehdy, pokud se MCU probouzí z režimu snížené spotřeby nebo resetu.

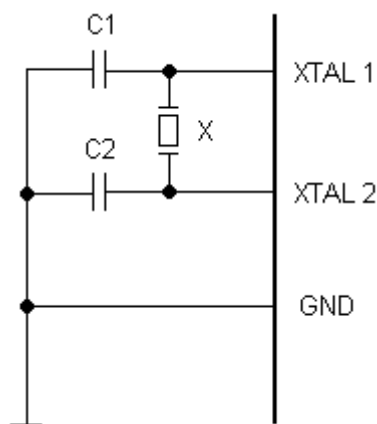
Zdroj hodinového signálu	CKSEL 3 . . 0
Vnější zdroj	0 0 0 0
Vnitřní kalibrovaný RC oscilátor 4MHz	0 0 1 0
Vnitřní kalibrovaný RC oscilátor 8MHz	0 1 0 0
Oscilátor Watchdog 128 kHz	0 1 1 0
Krystal 0,4 - 0,9 MHz	1 0 0 1
Krystal 0,9 - 3 MHz	1 0 1 1
Krystal 3 - 8 MHz	1 1 0 1
Krystal 8 a více MHz	1 1 1 1
Rezervováno (nepoužívat!)	0 0 0 1 / 0 0 1 1 / 0 1 0 1 / 0 1 1 1
Pozn.: "1" - nenaprogramováno ; "0" - naprogramováno	

Tab. 4: Nastavení pojistek CKSEL (převzato z [6])

SUT 1 0	Startovací čas	Čekací doba	Použití
0 1	16384 CK	14 CK	Povolení detekce napětí
1 0	16384 CK	14 CK + 4,1ms	Rychlý náběh odběru
1 1	16384 CK	14 CK + 65 ms	Pomalý náběh odběru
Pozn.: CK značí periodu hodinového impulsu			

Tab. 5: Volba startovací a čekací doby (převzato z [6])

V praktických aplikacích je nejrozšířenějším zdrojem hodin krystalový oscilátor nebo zkráceně krystal. Jejich výhodou je vysoká frekvenční stabilita, nízká cena a široké spektrum pracovních kmitočtů. Krystal připojujeme k vývodům podle obrázku 13 mezi vývody XTAL 1 a XTAL 2. Kondenzátory C1 a C2 by měli být keramické s kapacitou 15 - 30 pF. Dále je třeba si uvědomit, že výchozí nastavení pojistek CKSEL je vnitřní RC oscilátor 8MHz. Navíc je aktivní propojka CKDIV8, která ještě dělí kmitočet RC oscilátoru číslem 8. Pokud bychom použili jako zdroj hodin vnější krystal s kmitočtem jiným než 1MHz, vznikala by při komunikaci značná chyba. Toto může být velmi častý důvod nefunkční sériové komunikace.



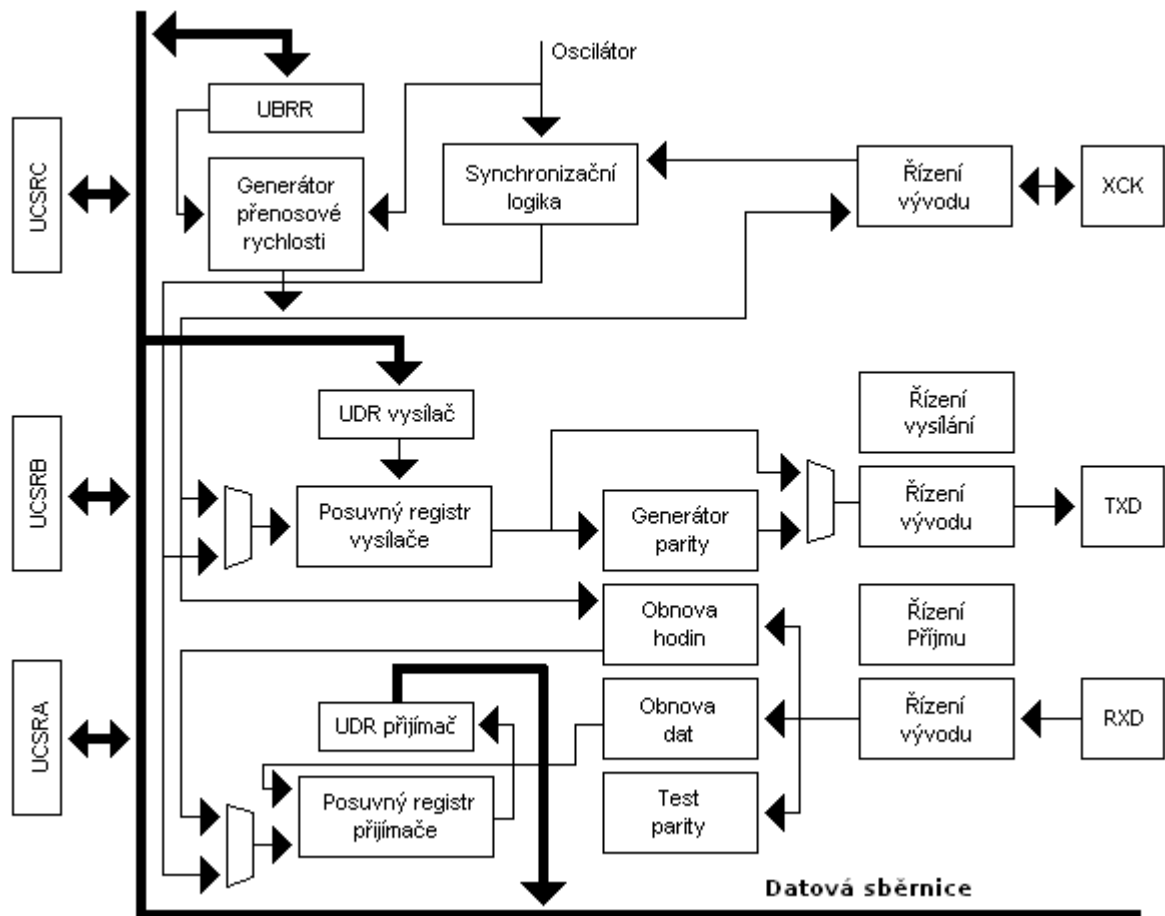
Obr. 13: Připojení krystalu k vývodům XTAL (převzato z [6])

3.7.4 Jednotka USART

Z hlediska použití čítače impulsů je nejdůležitější periferií MCU jednotka USART (Universal Synchronous and Asynchronous serial Receiver and Transmitter), jak ostatně plyne ze zadání práce. Proto se následujícím textu budeme věnovat jejímu popisu a především nastavení, které je nutné ke správnému chodu sériové komunikace.

3.7.4.1 Blokové schéma USART

Jednotka USART je plně duplexní (příjem i vysílání najednou) přijímač - vysílač schopný pracovat v asynchronním nebo synchronním režimu. Uživateli dovoluje nastavit přenosovou rychlost v rozmezí, které podporuje standard RS 232 a samozřejmě je také volba délky přenosového rámce (5 - 9 datových bitů + 1 až 2 stop bity). Podporovány jsou také paritní bity, detekce chyby rámce nebo detekce přetečení dat. Blokové schéma jednotky (obr. 14), lze rozdělit na tři hlavní části. Je to generátor přenosové rychlosti, vysílač a přijímač. Hodinový signál putuje ze zdroje přímo do gene-



Obr. 14: Blokové schéma jednotky vysílače a přijímače (přeloženo a převzato z [6])

rátoru přenosové rychlosti. Ten jej dále upravuje a rozděluje pro přijímač i vysílač. K dispozici je zde pouze vývod XCK. Ten se v režimu sériového programování stane vstupem hodin z master zařízení (viz. text dále).

Pro příjem a odeslání dat po sériové lince slouží osmibitový registr UDR (Usart Data Register). Tento registr je společný pro přijímač i vysílač. To znamená, že právě prováděná operace určuje, zda bude vstupní nebo výstupní. Pokud tedy chceme odeslat data z MCU, musíme je vložit do

registru UDR. Tím naše práce jako uživatele skončila. USART nyní přesune bajt z UDR do posuvného registru vysílače. Ten zajistí vysílání rámců bez časové prodlevy. V tomto okamžiku se k němu podle volby uživatele může připojit také paritní bit. Nakonec je rámec bit po bitu vystaven na vývod TXD. Při příjmu rámce je situace opačná. Rámec přichází po jednotlivých bitech na vývod RXD. Projde testem parity, který zjistí, zda byl datový rámec přijat bez chyby. Data jsou poté přesunuta do posuvného registru přijímače, kde zůstanou, dokud nejsou načtena do UDR. Opět je tak zajištěna kontinuita přenosu bez zbytečných prodlev.

Pokud by však došlo k situaci, kdy by byl registr UDR plný (totéž platí i pro jeho protějšek ve vysílací části) a hrozila by tak ztráta dat způsobená nahrazením již uložených nově příchozími, budou nastaveny příslušné bity v registru UCSRA. Proto je doporučeno tyto bity při vysílání nebo příjmu testovat. Více o popisu a funkci jednotlivých bloků v [5] nebo v [6].

3.7.4.2 Registr UCSRA

V popisu jednotky USART jsme zmiňovali bity, které jsou nastaveny při naplnění posuvných registrů přijímače resp. vysílače. Aby se předešlo nechtěné ztrátě dat, je doporučeno tyto bity testovat v podprogramech obsluhujících příjem resp. vysílání dat (podrobná ukázka v teoretické části). Jedná se o bity RXC, TXC a UDRE z řídicího a stavového registru UCSRA (Usart Control and Status Register A).

Je-li bit RXC nastaven, znamená to, že v přijímacím bufferu (příjímací část UDR) jsou nepřečtená data a lze je tak přečíst. Pokud je vyrovnávací paměť prázdná, příznak se vynuluje. Příznak TXC indikuje odeslání právě vysílaného rámce a vyprázdnění vysílacího bufferu (vysílací část UDR). Je-li vysílací buffer prázdný a tím pádem připraven přijmout nová data, nastaví se bit UDRE. Do vysílacího bufferu lze zapisovat jen tehdy, je-li tento příznak nastaven. V opačném případě budou zapisovaná data ignorována!

7	6	5	4	3	2	1	0	
RXC	TXC	UDRE	FE	DOR	UPE	U2X	MPCM	UCSRA
R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
0	0	1	0	0	0	0	0	

Obr. 15: Registr UCSRA

3.7.4.3 Registr UCSRB

V registru UCSRB (Usart Control and Status Register B) jsou nejdůležitější bity RXEN a TXEN. Pokud jsou tyto bity nastaveny je překryta standardní funkce vývodů PD0 a PD1 funkcí přijímače

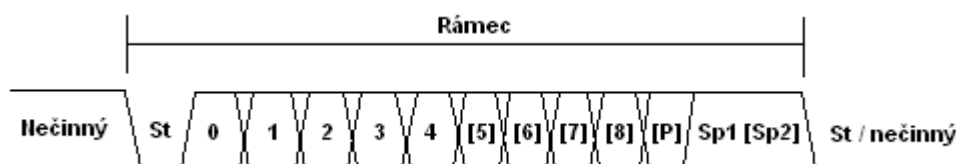
(RXD) a vysílače (TXD). Vynulováním těchto bitů dojde k odstavení přijímače a vysílače, tím však budou odstraněna data z registru UDR. Dále poznamenejme, že před každým novým přenosem je třeba jednotku USART zinicilizovat. To představuje nastavení registru SPL, nastavení rámce přenosu, přenosové rychlosti a povolení přijímače a vysílače. Ve formě podprogramu pak lze inicializaci provádět pouhým skokem na návěští (viz. teoretická část práce).

7	6	5	4	3	2	1	0	
RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	UCSRB
R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

Obr. 16: Registr UCSRB

3.7.4.4 Registr UCSRC

Existuje několik kombinací nastavení přenosového rámce. Volit lze počet datových bitů, paritu a počet stop bitů (viz. obr. 17).



Obr. 17: Obecný tvar přenosového rámce

Obvykle se však volí počet datových bitů 8 spolu s jedním start a stop bitem. Ty slouží pro synchronizaci příjmu, resp. vysílání. Všechna tato nastavení se provádí prostřednictvím registru UCSRC (Usart Control and Status Register C).

7	6	5	4	3	2	1	0	
-	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	UCSRC
R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
0	0	0	0	0	1	1	0	

Obr. 18: Registr UCSRC

Buňky spolu s popisky představují bity registru, číslo nad nimi pak pořadí bitu v registru. Značky R (čtení) nebo W (zápis), popřípadě jejich kombinace označuje možný druh práce s daným bitem a

konečně hodnota pod nimi, výchozí hodnotu bitu. Jednotlivé bity registru UCSRC mají následující význam:

- UMSEL – nastaví typ přenosu ; 0 = asynchronní, 1 = synchronní
- UPM0, UPM1 – nastaví paritu (viz. tab. 6)
- USBS – výběr stop bitu pro vysílač, přijímač jej ignoruje ; 0 = 1 stop bit , 1 = 2 stop bity
- UCSZ0, UCSZ1 – v kombinaci s bitem UCSZ2 z registru UCSRB nastaví počet datových bitů (viz. tab. 7)
- UCPOL – využíván pouze při synchronním přenosu

UPM0	UPM1	Parita
0	0	zakázána
0	1	sudá parita
1	0	vyhrazená kombinace
1	1	lichá parita

Tab. 6: Nastavení parity pomocí bitů UPM0 a UPM1 z registru UCSRC

UCSZ2	UCSZ1	UCSZ0	Počet datových bitů
0	0	0	5
0	0	1	6
0	1	0	7
0	1	1	8
1	0	0	rezervováno
1	0	1	rezervováno
1	1	0	rezervováno
1	1	1	9

Tab. 7: Nastavení počtu datových bitů v rámci bity UCSZ0 až UCSZ2

3.7.4.5 Registr UBRR

UBRR (Usart Baud Rate Register) je 16-bitový registr sloužící pro nastavení přenosové rychlosti. Hodnota v něm uložená určuje hodnotu předděličky přenosové rychlosti. Velikost této hodnoty pro asynchronní přenos vypočteme následovně:

$$UBRR = \frac{f_{osc}}{16 \cdot \text{PřenosováRychlost}} - 1,$$

kde f_{osc} je kmitočet krystalového oscilátoru a *PřenosováRychlost* je požadovaná rychlost přenosu v bitech za sekundu. Je důležité znát jaká rychlost bude nakonec jednotkou generována, protože pokud se bude rychlost příchozích dat výrazně lišit od rychlosti generované MCU, přijímač nebude schopen synchronizovat start bity a data nebudou rozeznána. K tomu slouží následující vzorec:

$$\text{GenerovanáRychlost} = \frac{f_{osc}}{16 \cdot (UBRR + 1)} \text{ [bit/s]}.$$

Celkovou chybu generované rychlosti v jednotkách procent pak stanovíme jednoduše podle:

$$\text{ChybaPřenosu} = \left(\frac{\text{GenerovanáRychlost}}{\text{PřenosováRychlost}} - 1 \right) \cdot 100 \text{ [\%]}.$$

Výrobce doporučuje maximální chybu 1,5 – 2%. Tuto chybu lze minimalizovat buď kompromisem mezi hodnotou registru UBRR a přenosové rychlosti nebo speciální hodnotou frekvence krystalu (např. 11,0592MHz). Taková hodnota zaručí, že dělení ve výše uvedených rovnicích (pouze pro standardní přenosové rychlosti - typicky 9600 bit/s) bude beze zbytku a tak bude vygenerovaná rychlost maximálně přesná.

3.7.5 Programování ATtiny2313

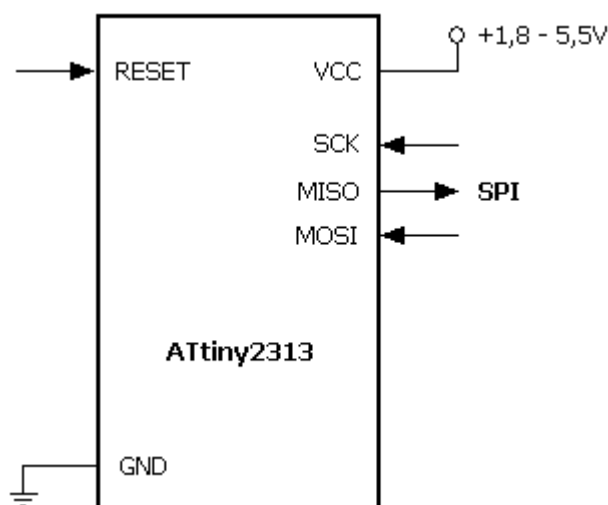
Jak již bylo řečeno, MCU ATtiny2313 umožňuje uživateli programovat paměť FLASH i EEPROM. K tomuto účelu jsme se rozhodli používat sériové rozhraní SPI řízené přímo linkami

UART. Toto rozhraní sestává ze tří vývodů (viz. obr. 19) a to SCK (vstup hodin), MOSI (vstup) a MISO (výstup). MCU je v režimu sériového programování vždy podřízené zařízení! Proto veškerá iniciativa přichází z nadřízeného zařízení (v našem případě PC). Perioda hodinového signálu na SCK však musí být při programování vždy o něco větší, než perioda zdroje (viz. tab. 8). S tímto požadavkem také musí korelovat správné nastavení propojek CKSEL. Budeme-li například použi-

Frekvence zdroje hodin f_{ck} [MHz]	Záporný puls SCK [$1/f_{ck}$]	Kladný puls SCK [$1/f_{ck}$]
< 12	2x	2x
> 12	3x	3x

Tab. 8: Závislost periody SCK na frekvenci zdroje hodin

vat krystal 20MHz, bude perioda tohoto signálu 50ns. Perioda hodin na SCK tedy musí být větší než 300ns. Tato hranice je více než dostačující. Během vývoje čítače jsme totiž nenalezli způsob, jak programově pod OS Windows realizovat časové prodlevy kratší než několik mikrosekund.



Obr. 19: vývody ATtiny2313 potřebné pro sériové programování

Při zapisování dat do ATtiny je třeba pamatovat také na to, že data jsou časována nástupnou hranou SCK a při čtení pak sestupnou hranou SCK. Jako první je třeba vystavit nejvýše významný bit (tzv. MSB) prvního bajtu, což je poměrně důležitá informace. Formát instrukcí je čtyřbajtový. První bajt zpravidla definuje charakter instrukce a používanou paměť. Druhý a třetí bajt pak obsahují informaci o adrese, ke které se přistupuje. Čtvrtý bajt pak obsahuje data.

Pro uvedení MCU ATtiny2313 do režimu sériového programování doporučuje ATMEL dodržet následující posloupnost úkonů:

- MCU musí být připojen na napájení zatímco jsou vývody SCK a RESET nastaveny na hodnotu logické nuly. Pokud tuto podmínku nelze splnit (nejprve se připojí napájení), nastavíme SCK na log. 0 a pošleme na RESET kladný puls s délkou trvání alepoň 2 hodinových cyklů MCU.
- Následuje čekání nejméně 20ms a pak povolení seriového programování vysláním povolovací instrukce na vývod MOSI (viz. tabulka 8).
- Instrukce seriového programování nebudou fungovat pokud nebude komunikace synchronizována. Bude-li synchronizace provedena, obdržíme ve třetím přijatém bajtu hodnotu 83 (hexadecimálně 0x53) jako echo druhého bajtu. Ať už je echo přijato nebo ne, všechny čtyři bajty musí být odeslány. Pokud hodnota 83 není poslána zpět, pošle se na RESET kladný puls a celá procedura se opakuje.
- Paměť Flash je programována po stránkách (1 stránka = 16 slov). Nejprve je tedy třeba načíst data do stránky pomocí instrukce Načtení stránky programové paměti a čtyř bitů adresy B. Nejprve je třeba načíst nižší bajt slova a pak vyšší a to za použití stejné adresy B. Po naplnění stránky pak s pomocí šesti bitů adresy a instrukce Zápis stránky programové paměti, stránku zapíšeme do paměti. Před načítáním další stránky je třeba čekat alespoň 4,5ms.
- Na konci sériového programování nastavíme RESET na log. 1, popřípadě vypneme napájení.

Během vývoje programátoru pro čítač impulsů se ukázalo, že v prvním kroku není krátký kladný puls na RESET nutný. Dále doporučujeme před načtením další stránky dat čekat déle než 4,5ms (například 10ms). Zkušenost nám ukázala, že hodnoty doporučené výrobcem (obzvláště ty časové) v praxi často nefungují. Proto je dobré ponechat si vždy nějakou rezervu. Na tomto místě také považujeme za důležité upozornit čtenáře na jeden fakt, který v dokumentaci ATtiny2313 nenajde. Jejím pořadí obsluhy vývodů sběrnice SPI. Nejprve musíme přečíst bit na vývodu MISO a pak zapsat bit na MOSI. Nakonec vyšleme kladný puls na SCK a pokud je to nutné, celou proceduru opakujeme. Ačkoli by to mělo být irelevantní ukázalo se, že právě toto pořadí je nejrychlejší a správné.

V následující tabulce jsou uvedeny instrukce nezbytné pro potřeby této práce. Pro další informace o sériovém programování a o ostatních instrukcích doporučujeme [6].

Instrukce	Formát instrukce				Stručný popis
	Bajt 1	Bajt 2	Bajt 3	Bajt 4	
Povolení programování	1010 1100	0101 0011	XXXX XXXX	XXXX XXXX	Povolí sériové programování
Smazání čipu	1010 1100	100X XXXX	XXXX XXXX	XXXX XXXX	Smaže paměti Flash i EEPROM
Načtení stránky programové paměti	0100 H000	000X XXXX	XXXX BBBB	I I I I I I I I	Zapíše bajt H dat I do stránky programové paměti na adresu slova B
Zápis stránky programové paměti	0100 1100	0000 00AA	BBBB XXXX	XXXX XXXX	Zapíše stránku do programové paměti na adresu A:B
Čtení programové paměti	0010 H000	0000 00AA	BBBB BBBB	O O O O O O O O	Čte bajt H dat O z programové paměti na adrese A:B
Čte zamykacích bitů	0101 1000	111X XXXX	XXXX XXXX	XXXX XXOO	Čte zamykací bity. 0 - povolen, 1 - nepovolen
Zápis zam. bitů	1010 1100	111X XXXX	XXXX XXXX	XXXX XX I I	Zapíše zamykací bity
Čte pojistkové bity	0101 0000	0000 0000	XXXX XXXX	O O O O O O O O	Přečte pojistkové bity. 0 - povolen, 1 - nepovolen
Zapíše pojist. bity	1010 1100	1010 0000	XXXX XXXX	I I I I I I I I	Zapíše pojistkové bity
Vysvětlivky:					
<ul style="list-style-type: none"> • A - vyšší bajt adresy , B - nižší bajt adresy • H = 1 - vyšší bajt slova , H = 0 - nižší bajt slova • O - výstupní data (ve smyslu MCU) , I - vstupní data (ve smyslu MCU) • X – na hodnotě nezáleží 					

Tab. 9: Vybrané instrukce pro sériové programování

3.8 Integrovaný obvod FT232BM

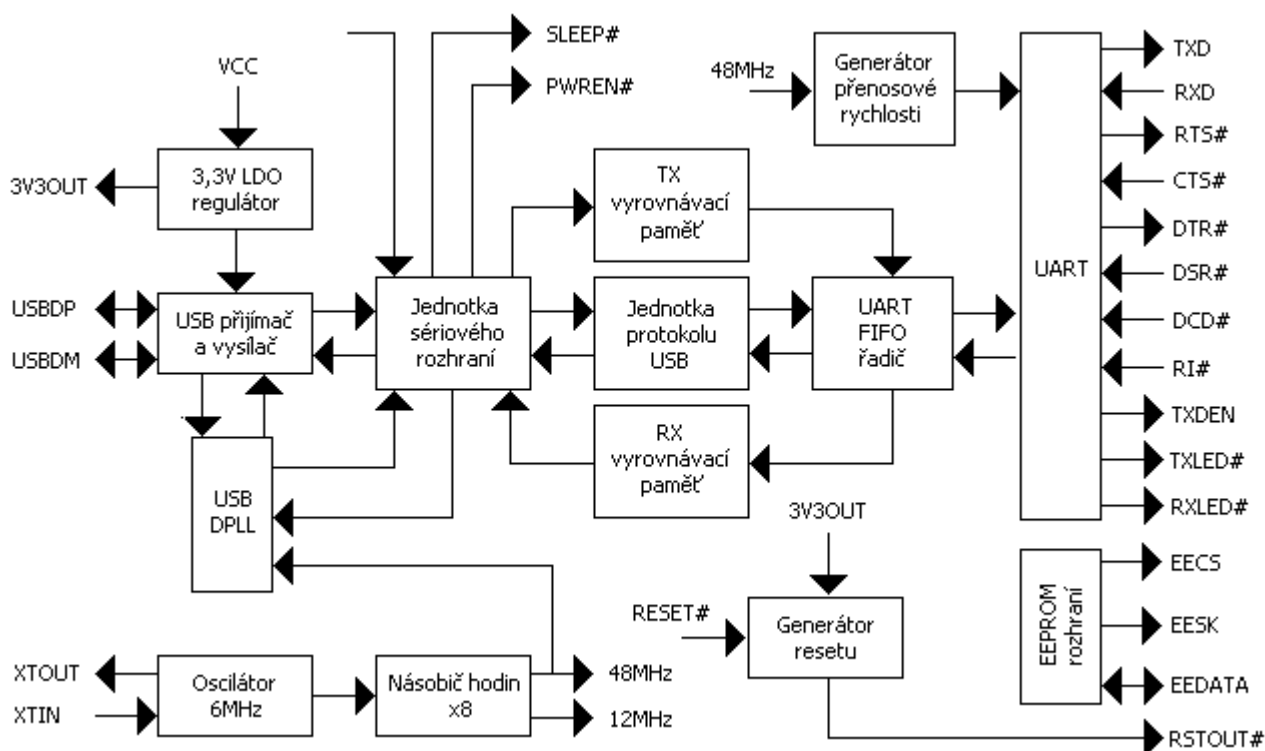
Pro převod signálů USB na UART jsme vybrali obvod FT232BM firmy FTDI. V následujících kapitolách uvedeme jeho nejdůležitější vlastnosti a ukážeme si jak k němu připojit periferní součástky.

3.8.1 Základní vlastnosti FT232BM

Jedná se o jednočipový převodník USB <=> RS-232. To s sebou přináší výhodu minimálního počtu externích součástek. Přitom podporuje plné řízení toku dat (handshaking) a všechny linky modemu. Podporováno je také široké rozmezí přenosových rychlostí od 300 Baudů do 1M Baud pro RS-232 a až 3M Baud pro TTL a RS-422/RS-485. Z dalších HW vlastností jmenujme například přijímací a vysílací vyrovnávací paměť, které zajistí vysokou kontinuitu přenosu dat. Obvod také

obsahuje integrovaný převodník úrovní pro 3,3V a 5V TTL logiku. To je výhodné pro účely této práce, protože chceme-li využít linek UART k řízení sběrnice SPI, odpadá nutnost použití dalšího konvertoru úrovně. Mezi další výhody patří skutečnost, že pokud není vyžadováno napájecí napětí větší jak 5V, může FT232BM sloužit zároveň jako zdroj napětí s odběrem proudu až 500mA (tyto hodnoty pro čítač impulsů plně postačují). Na stránkách výrobce jsou ke stažení také potřebné ovladače pro OS Windows, Mac OS a Linux 2.40 a vyšší. Jedná se o klasický ovladač VCP (Virtual Communication Port) a pro využití pokročilých funkcí je tu také přímý USB ovladač D2XX včetně DLL knihoven. Ke stažení je zde rovněž programová aplikace, pomocí níž lze do externí EEPROM ukládat VID, PID a sériová čísla spolu s popisem připojeného zařízení.

Nakonec zde ještě uveďme blokové schéma FT232BM. Pro účely této práce není nutné znát jeho funkční popis a tak postačí pouze obrázek. Pokud by některého čtenáře zajímaly detaily odkazujeme ho na [1] nebo [8].



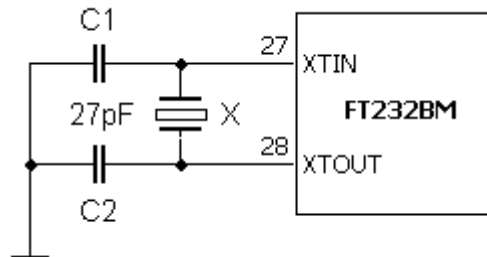
Obr. 20: Blokové schéma obvodu FT232BM (přeloženo a převzato z [8])

3.8.2 Příklady zapojení

Ke správné činnosti potřebuje obvod FT 232 BM připojit k určitým svým vývodům několik dalších součástek. Následující řádky se proto zabývají připojením periferních součástek. Uveden bude také příklad zapojení za účelem napájení přímo z portu USB. Celkové schéma zapojení čítače impulsů spolu s popisem a seznamem součástek bude uvedeno až v praktické části práce.

3.8.2.1 Připojení krystalu

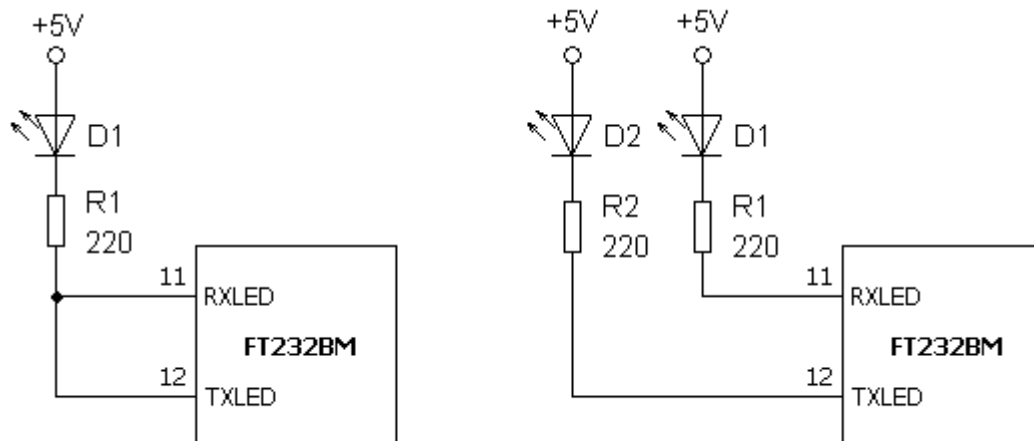
Základním předpokladem pro správnou funkčnost čipu FTDI je připojení krystalu nebo keramického rezonátoru s rezonančním kmitočtem 6MHz k vývodům XTIN a XTOUT. My jsme pro účely této práce zvolili krystal. Výrobce doporučuje v tomto případě připojení dvou zatěžovacích kondenzátorů kapacity 27pF.



Obr. 21: Připojení krystalu k obvodu FT232BM

3.8.2.2 Signalizační LED

FT232BM má také dva vývody určené pro připojení LED indikujících příjem nebo vysílání dat. Zvláště při vývoji oceníte jejich funkci. Pokud si totiž nejste jisti funkční komunikační linkou, je dobré vědět, zda jsou data skutečně přenášena. Při příjmu resp. vysílání totiž přejde vývod RXLED



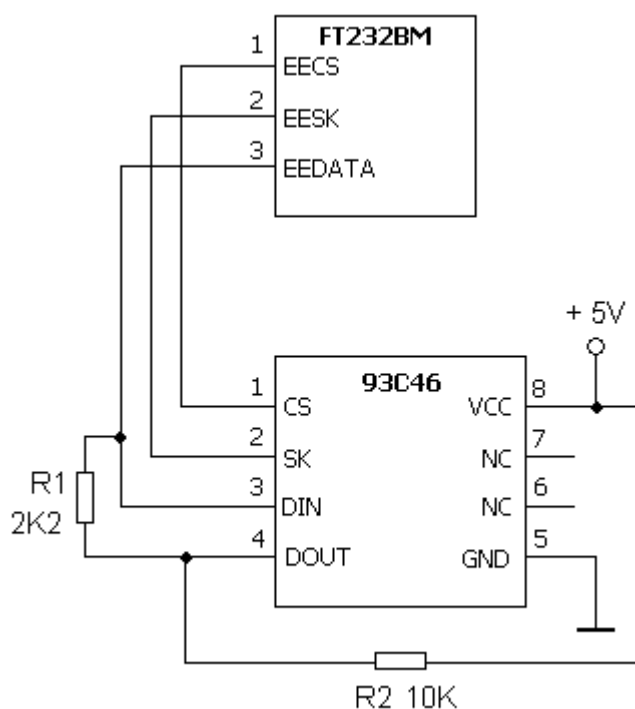
Obr. 22: Zapojení LED pro indikaci přenosu dat

resp. TXLED ze stavu vysoké impedance do logické nuly (jedná se o výstup typu otevřený kolektor), takže lze připojenou LED indikovat přenos dat. Je jasné, že při vyšších přenosových rychlostech by uživatel nebyl schopen takový děj prostřednictvím LED sledovat. Proto je pro prodloužení impulsu použit monostabilní klopný obvod. Ke sledování přenosu dat lze použít jednu

nebo dvě LED (viz. obr. 22).

3.8.2.3 Paměť EEPROM

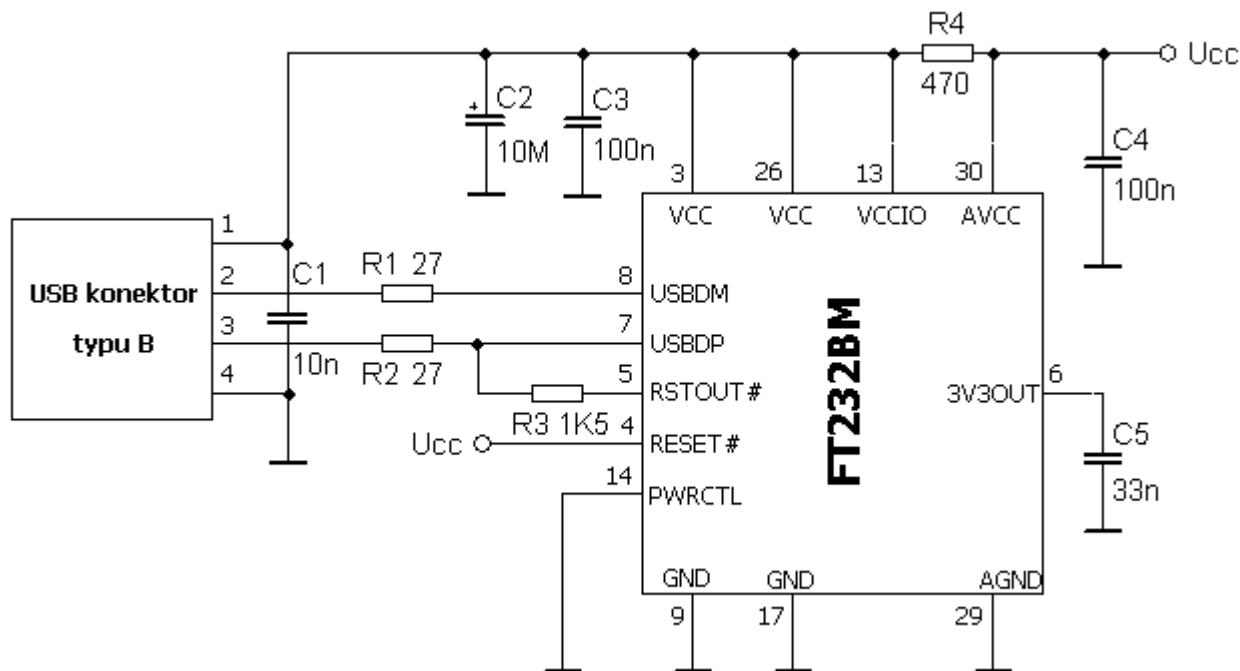
Při restartování zařízení nebo při připojení k portu USB se testuje, je-li připojena externí paměť, a pokud ano, tak jestli obsahuje platná data. Pokud je skutečně obsahuje, je použita pro definici deskriptorů USB. Paměť musí být typu EEPROM s délkou slova 16 bitů. Doporučen je Microchip 93LC46C nebo ekvivalentní. Při použití ekvivalentu je třeba zkontrolovat především délku slova, rychlost čtení (1Mb/s) a napájecí napětí (+4,4V až +5,25V). Na následujícím obrázku je patrné propojení příslušných vývodu paměti s FT232BM.



Obr. 23: Připojení paměti EEPROM k čipu FT 232 BM

3.8.2.4 Zapojení FT232BM pro napájení z USB

Hlavní předností FT232BM, která byla využita také při návrhu čítače impulsů, je možnost napájet aplikaci přímo z portu USB. To je výhodné zvláště tehdy, pokud pracujete s TTL obvody (postačuje napětí +5V) a odběr proudu nepřesáhne 500mA. Pro aplikace, které tyto požadavky nesplňují je tu samozřejmě také možnost napájení z vlastního zdroje. Nicméně toto je skutečně maximální hodnota odebíraného proudu a k připojení nesmí být použit rozbočovač. Zároveň je to maximální hodnota pro všechny porty USB.



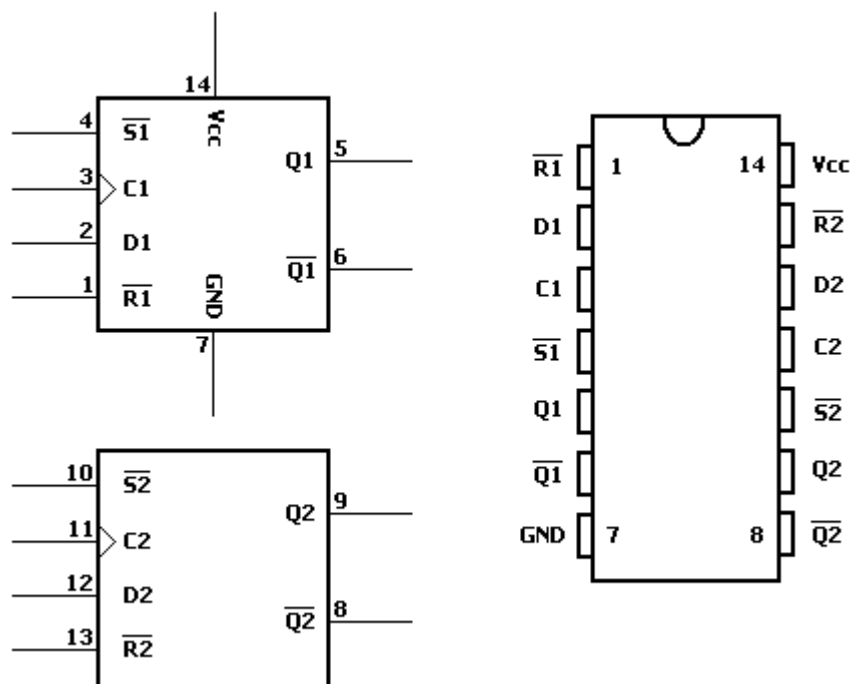
Obr. 24: Napájení aplikace z portu USB pomocí FT232BM

3.9 Klopný obvod 74F74

Integrovaný obvod 74F74 byl vybrán, aby pracoval jako vstupní člen čítače impulsů. Jedná se o dvojnásobný klopný obvod typu D, reagující na vzestupnou hranu hodinového impulsu s možností nastavení nebo nulování výstupu nezávisle na hodinovém a datovém vstupu. Přesněji byl zvolen typ Texas Instruments SN74F74N, jehož schematická značka a pouzdro DIL 14 je zobrazeno na obrázku 25. Tento obvod byl vybrán pro konstrukci čítače impulsů zejména proto, že je to logický obvod typu TTL. Je tedy zajištěna kompatibilita s jednofotonovým detektorem, který slouží jako zdroj čítaných impulsů. Dále je schopen detekovat impulsy minimální délky 4ns a přicházejících s frekvencí až 100MHz. Tyto dynamické parametry tedy zcela splňují požadavky na čítač impulsů vyjmenované v úvodu diplomové práce.

Funkci klopného obvodu popisuje tabulka 10. Jak je patrné, obvod může pracovat ve třech režimech. Prvním z nich je asynchronní režim. V tomto režimu pracuje jako klasický klopný obvod R-S. To znamená, že logickou hodnotu na výstupu Q lze nastavit pomocí dvojice vstupů R a S (negované) nezávisle na datovém nebo hodinovém vstupu. Další mód je čistě synchronní. Logická hodnota na datovém vstupu D se přepíše na výstup Q s náběžnou hranou hodinového signálu na vstupu C. Poslední režim využívá kombinace předchozích dvou. Právě kombinace synchronního a asynchronního režimu byla vybrána pro konstrukci čítače impulsů. Pokud má čítač detekovat sled impulsů z jednofotonového detektoru, je výhodné aby právě náběžná hrana těchto impulsů přepisovala logickou hodnotu na datovém vstupu (pevně nastavena na log. 1) na výstup, zatímco

vstupy R a S (negované) budou nastaveny do log. 1. Přitom nevadí, že frekvence příchozích impulsů se bude neustále měnit. Jediným omezením je, aby doba mezi dvěma impulsy nebyla být kratší než 10ns (1/100MHz). Pro více informací doporučujeme katalogový list [9].



Obr. 25: Schematická značka klopného obvodu 74F74 a popis vývodů u pouzdra DIL14

Vstupy				Výstupy	
S (neg.)	R (neg.)	D	C	Q	Q (neg.)
L	H	X	X	H	L
H	L	X	X	L	H
H	H	H	↑	H	L
H	H	L	↑	L	H
H	H	X	↓	předchozí stav	
L	L	X	X	hazardní stav	
Pozn.: X – na logické hodnotě nezáleží, L – logická nula, H – logická jedna ↑ - vzestupná hrana hodinového impulsu, ↓ - sestupná hrana hodinového impulsu					

Tab. 10: Funkční tabulka klopného obvodu 74F74

Při konstrukci čítače impulsů byly použity ještě další integrované obvody. Jedná se především o IO 74HCT14, 74HCT32 a 74HCT4066. Nebudeme je zde však popisovat. Jednak proto, že nejsou klíčové a také proto, že jejich použití je velmi jednoduché. V seznamu literatury však uvádíme odkaz na internetové stránky, kde čtenář nalezne jejich katalogové listy.

4 Praktická část

4.1 Schéma zapojení, návrh a výroba DPS

V této části práce se budeme věnovat popisu zapojení čítače impulsů a návrhu desky plošného spoje. Schéma bylo kvůli praktickým důvodům rozděleno do tří částí. Tyto části jsou ve skutečnosti propojeny a bude z nich vytvořena jedna DPS (Deska Plošného Spoje).

4.1.1 Schéma zapojení čítače impulsů

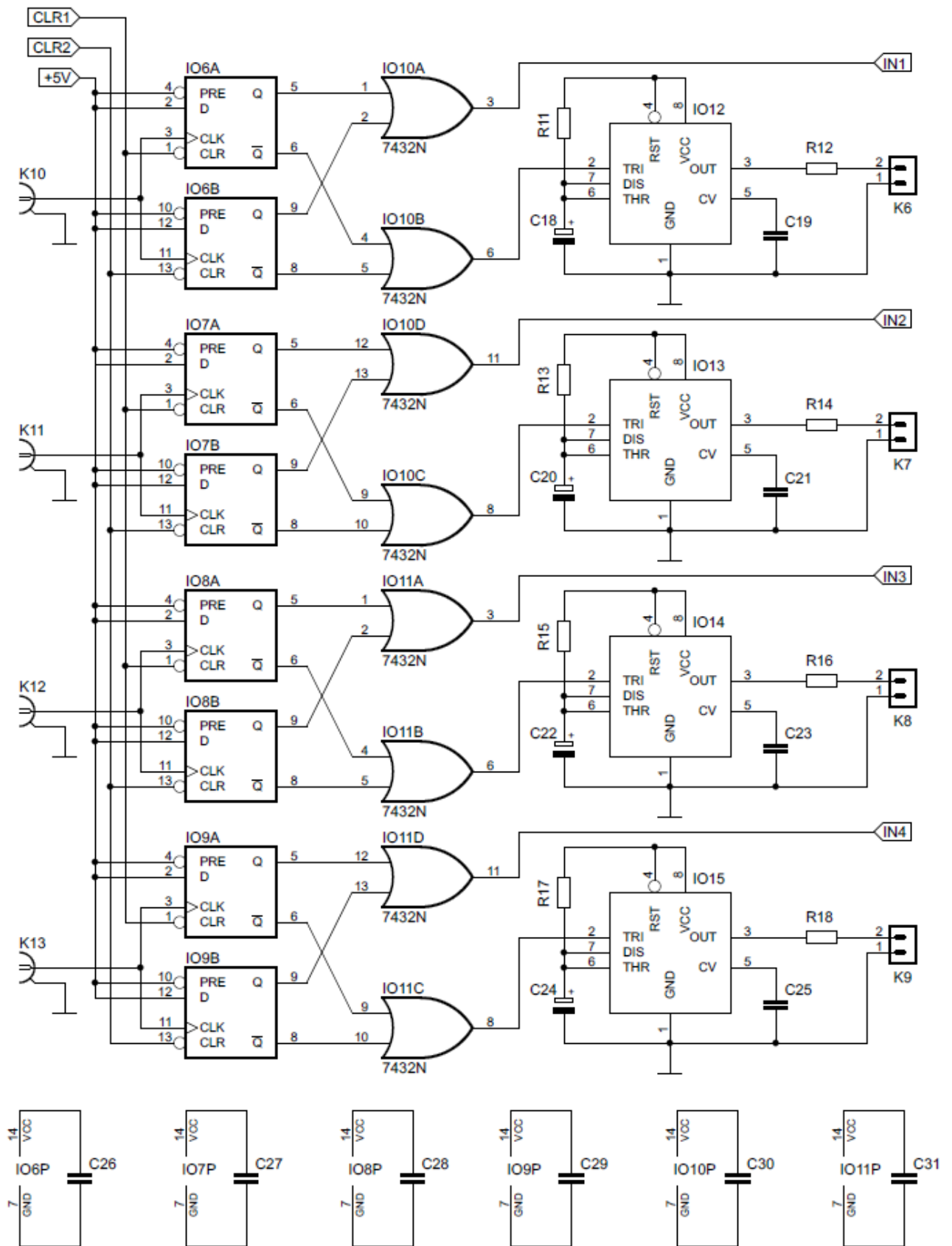
Na obrázku 26 je zobrazena první část schématu čítače impulsů. To představuje v souladu se schématem v kapitole 3.2. blok Vstupní TTL obvody. Hlavními součástkami jsou zde klopné obvody 74F74. Ty zpracovávají vstupní, který je z detektoru přiveden na BNC konektory K až K. Pro snazší montáž jsou zvoleny typy do DPS. Odsud je signál přiveden na hodinové vstupy dvojice IO 74F74. To proto, aby se minimalizovala chyba při čítání (viz. dále). Vstupy PRE a D jsou permanentně připojeny na +5 V. Signály CLR1 a CLR2 propojují střídavě části A a B klopných obvodů. Kromě toho jsou propojeny s MCU na obrázku 27.

Protože na vstupně výstupní bráně PB mikrokontroléru ATtiny2313 máme volné jen čtyři vstupy a dvojice klopných obvodů na každém vstupu počet výstupů Q zdvojnásobuje, museli jsme použít IO jako sčítací členy. Je zřejmé, že se jedná o hradla OR (viz. []). Tato hradla sečtou logickou hodnotu výstupů Q a nonQ. Součet výstupů Q je pak vždy vyveden k MCU, který bude logické hodnoty na těchto vodičích číst. Z praktických důvodů by bylo výhodné, aby mohla obsluha čítače rozpoznat příchozí impulsy.

To lze například pomocí indikační LED. Nicméně doba, po kterou bude na výstupu sčítacího hradla žádaná logická hodnota je příliš krátká (asi 1μs) na to, aby vybudila LED. Proto je součet výstupů nonQ přiveden na vstupy TRI integrovaných obvodů NE555 (viz. [10]). Každý „časovač“ 555 je v zapojení monostabilního klopného obvodu. To znamená, že příchozí záporný puls na triggerovací vstup, překlopí výstup do stavu logické jedničky. Tato doba je dána hodnotami rezistoru R a kondenzátoru C:

$$T = 1,1 \cdot R \cdot C .$$

Postupujeme tak, že pro požadovanou dobu nejprve zvolíme hodnotu kondenzátoru. Pak dopočítáme rezistor, ovšem tak, aby velikost jeho odporu byla v rozmezí 1kΩ až 10MΩ. Námi zvolená doba překlopení je asi 4,5ms. To je čas dostatečně dlouhý na vybudění LED a zároveň dostatečně krátký na to, aby při sérii příchozích pulsů LED blikala. Protože se předpokládá montáž



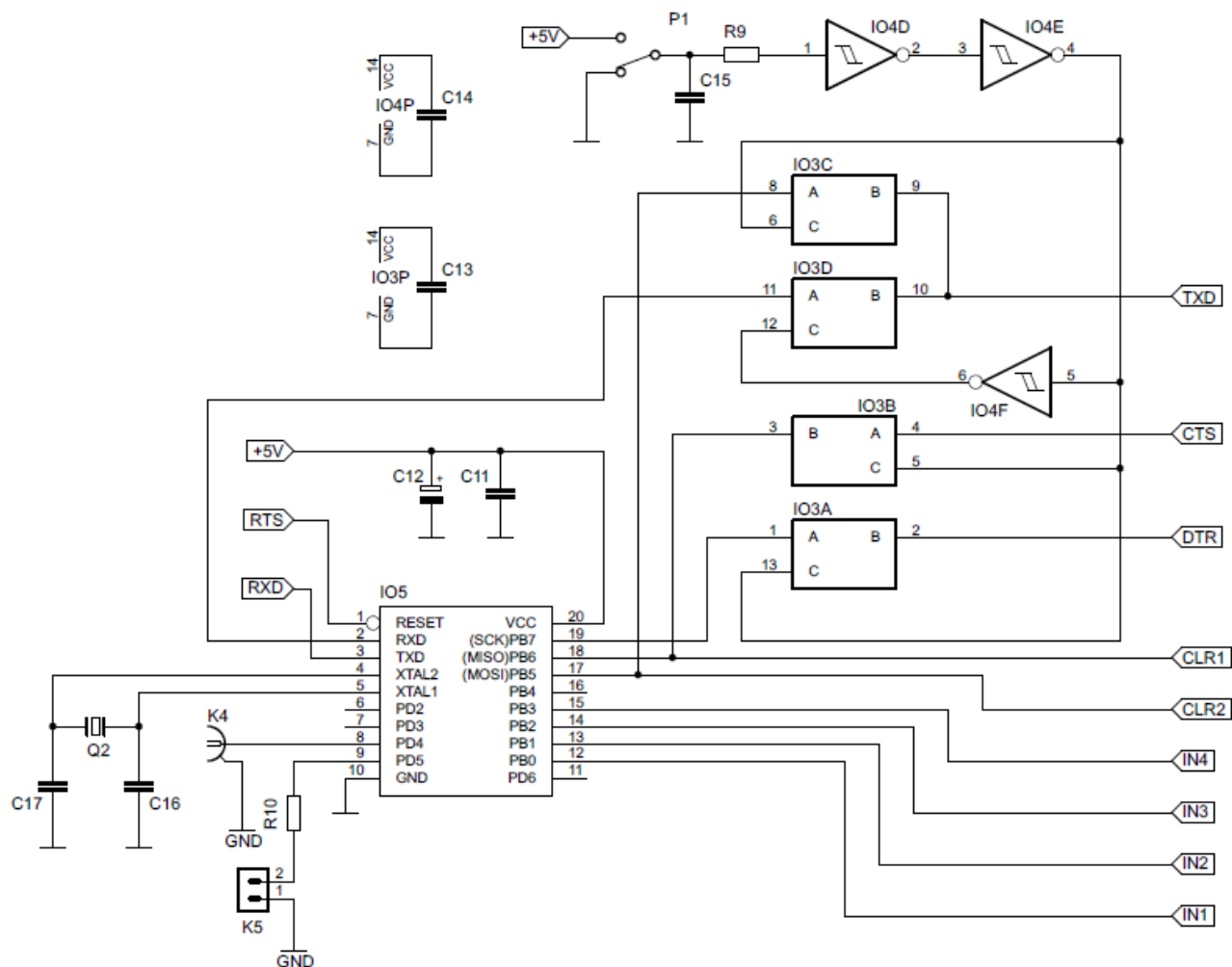
Obr. 26: Schéma čítače impulzů: vstupní část

čítače impulsů do nějaké krabičky, počítáme s připojením LED ke konektorům K až K. Tak je bude možné v případě demontáže čítače snadno odpojit. Kondenzátory C až C blokují napájení integrovaných obvodů IO až IO. V návrhu se počítá s typem pro povrchovou montáž (tzv. SMD), aby byly co nejlíže napájecím vývodům.

Seznam součástek pro vstupní část čítače impulsů:

K6 až K9.....	PFH02-02P
K10 až K13.....	BNC-Z 50RW
R11, R13, R15, R17.....	4K5, 7,5mm
R12, R14, R16, R18.....	1K8, 7,5mm
C19, C21, C23, C25.....	10nF keramické
C18, C20, C22, C24.....	1μF/50V, elektrolytický
C26 až C31.....	100nF keramické , SMD 1206
IO6 až IO9.....	SN74F74N
IO10, IO11.....	SN74HCT32N
IO12 až IO15.....	NE555P

Další část schématu čítače impulsů (viz. obr. 27) je poměrně jednoduchá. Tuto jednoduchost umožnilo použití mikrokontroléru ATtiny2313 jako hlavní součástky. MCU v sobě skrývá jak jednotku UART zajišťující komunikaci s počítačem, tak množství vývodů, kterými lze číst vstupní TTL obvody 74F74. Je tedy již zřejmé, že se jedná o tu část, která odpovídá blokům Řízení přenosu, čtení impulsů a Jednotka UART na obrázku 1 v kapitole 3.2. Hodinový kmitočet je odvozen od krystalu připojeného k vývodům XTAL1 a XTAL2. Kondenzátory C11 a C12 blokují napájecí napětí a měli by být co nejlíže napájecímu vývodu. Použití elektrolytického tantalového kondenzátoru C12 zajistí pokrytí špičkových odběrů proudu mikrokontrolérem. Vývod PD4 je přímo napojen na konektor K4 a slouží jako triggerovací výstup, který se aktivuje při čítání. LED připojená ke konektoru K5 pak indikuje, že je aktivní. První čtyři vývody brány PB slouží jako vstupy pro čtení klopných obvodů 74F74. Protože čítač bude zároveň sloužit také jako programátor pro MCU ATtiny2313, je nutné tuto funkci odlišit od funkce čítání. Už z toho důvodu, že signál TXD se používá jak při programování tak při sériové komunikaci. Proto tu je obvod IO3. Je to čtyřnásobný analogový přepínač, který je možné ovládat logickou hodnotou na povolovacím vstupu. Tento vstup je ovládán páčkovým přepínačem P1. Ten je však zdrojem zákmitů, které je nutno odstranit. Proto je za přepínač zapojen integrační článek, který tyto zákmity odfiltruje.

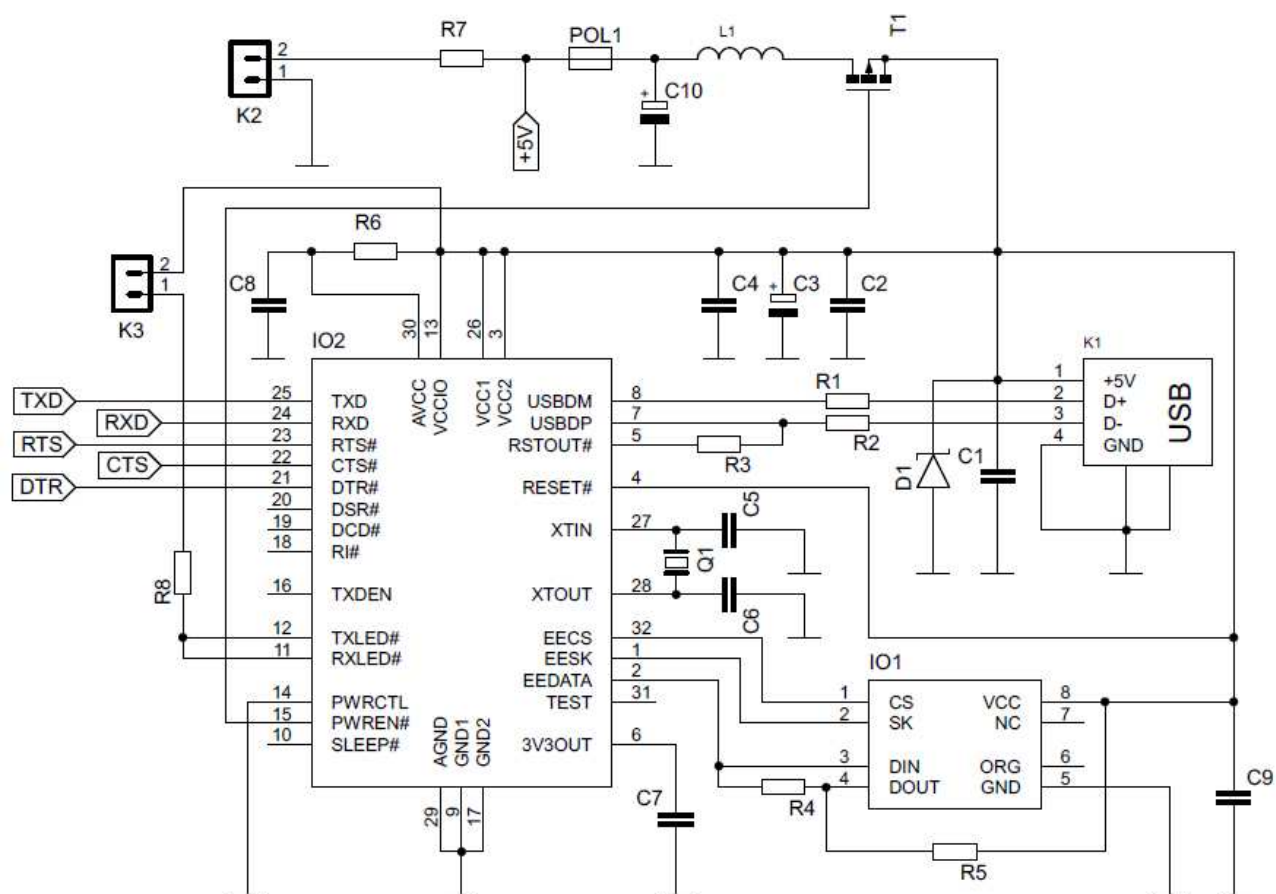


Obr. 27: Schéma čítače impulzů: část s MCU a přepínacím obvodem

Seznam součástek pro přepínací část a část s MCU čítače impulzů:

P1.....	KNX-1
K4.....	BNC-Z 50RW
K5.....	PFH02-02P
Q2.....	20MHz, 10mm
R9	100K, 7,5mm
R10.....	1K8, 7,5mm
C11, C13, C14	100nF keramické , SMD 1206
C12.....	10µF/25V, tantalový, SMD D
C15.....	100nF keramický, 5mm
C16, C17.....	27pF keramické, SMD 1206
IO3.....	CD74HCT4066E
IO4.....	MC74HCT14AN
IO5.....	ATTINY2313-20PU

Bohužel zmenšuje strmost hran číslicového signálu, ale IO a IO se Schmittovými klopnými obvody tento jev kompenzují. Bohužel každý člen zvlášť neobsahuje dva vstupy, mezi kterými by bylo možné přepínat. Proto linku TXD přivádíme na jednu dvojici z nich. Pomocí IO4 jako invertoru logických úrovní zajistíme, že dvojice přepínačů bude pracovat jako jeden typu 1x vstup – 2x výstup. Linky CTS a DTR se používají jen při programování a tak postačí jejich připojení resp. odpojení pouze jedním přepínačem. Poznamenejme ještě, že obvody přepínače a invertoru jsou vyráběny technologií CMOS (Complementary Metal Oxide Semiconductor), která je standardně nekompatibilní s TTL. Naštěstí se dnes vyrábí řada HCT (High speed Cmos compatible with Ttl), která tento neduh odstraňuje.



Obr. 28: Schéma čítače impulsů: část převodníku

Seznam součástek pro část převodníku:

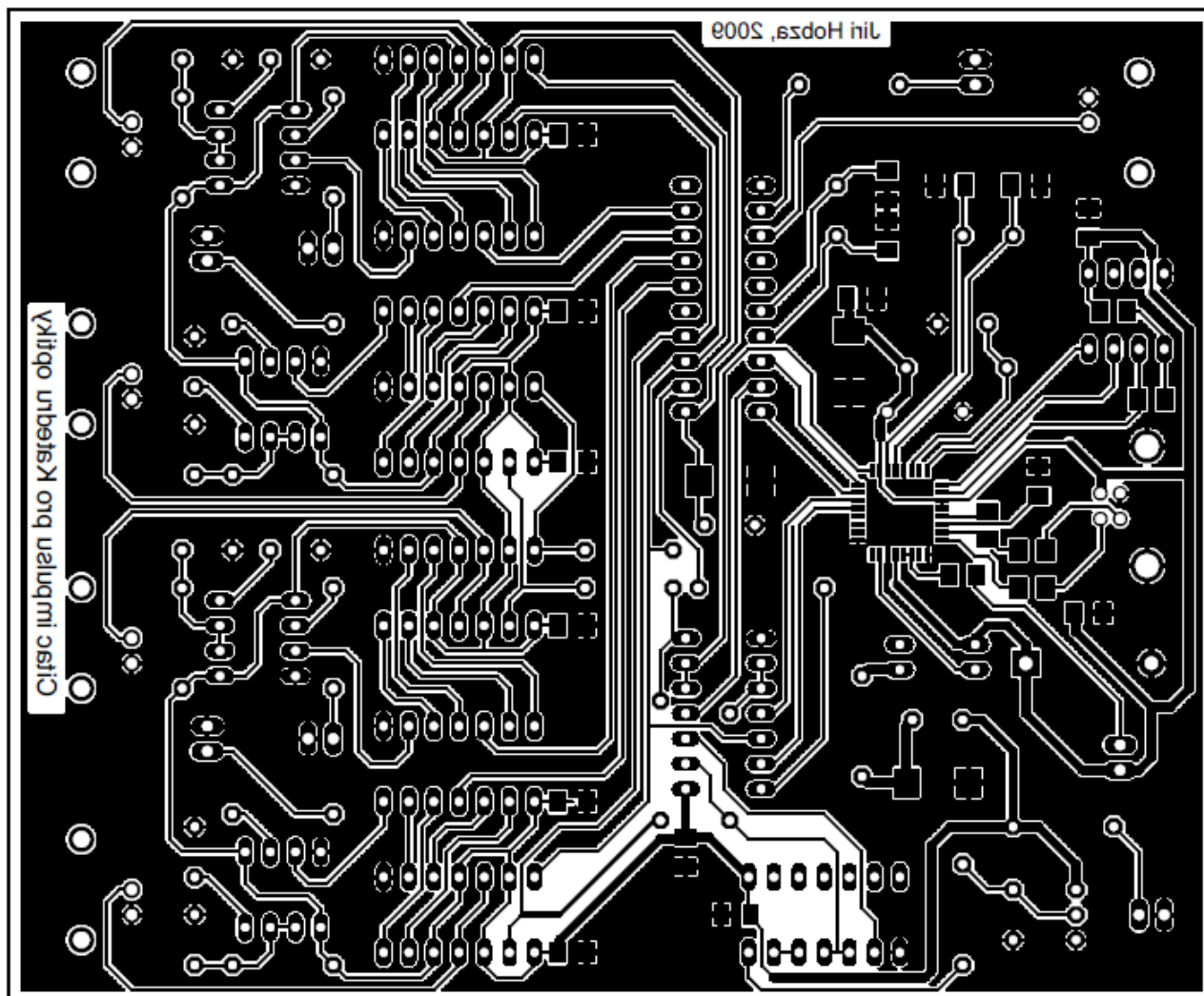
- K1.....USB B, DPS
- K2, K3.....PFH02-02P
- Q1.....6MHz, 10mm
- R1, R2.....27R, SMD 1206

R3.....	1K5, SMD 1206
R4.....	2K2, SMD 1206
R5.....	10K, SMD 1206
R6.....	470R, 7,5mm
R7.....	1K8, 7,5mm
R8.....	220R, 7,5mm
C1.....	10nF, SMD 1206
C2, C4, C9	100nF keramické , SMD 1206
C5, C6.....	27pF, SMD 1206
C7.....	33nF, SMD 1206
C8.....	100nF, keramický
C3, C10.....	10µF/25V, tantalový, SMD D
L1.....	33µH
POL1.....	RXE50
D1.....	BZX51
T1.....	IRFD9120
IO1.....	93LC46B
IO2.....	FT232BM

Posledním blokem ze schématu na obrázku 1, který jsme ještě nepopsali konkrétním zapojením je Převodník RS232/USB na UART. Při sestavování schématu na obrázku 28 jsme využili znalostí z předchozích kapitol. Zapojení součástek kolem IO1 je více méně předepsané výrobcem. Zenerova dioda D1 funguje jako napěťová ochrana. Trojice kondenzátorů C2 až C4 blokuje napájecí napětí a měli by být co nejbližší obvodu. Ke konektoru K2 se připojuje LED signalizující obousměrnou komunikaci po lince USB. Tranzistor T1 je typu MOSFET a zajišťuje napájení pro další části obvodu. Tlumivka L1 a kondenzátor C10 jej vyhlazují. Vratná pojistka POL1 omezuje proud nad 500mA. LED připojovaná ke K3 indikuje přítomnost napájecího napětí a tím zapnutí čítače.

4.1.2 Návrh DPS

Návrh DPS jsme provedli v programu Eagle (Easily Applicable Graphical Layout Editor) od vývojářské firmy Cadsoft. Instalační soubor tohoto programu čtenář nalezne na doprovodném CD.



Obr. 29: DPS čítače impulsů

Není účelem této práce popisovat práci v tomto programu. Domníváme se, že to není ani nutné. Jeho ovládání je do značné míry intuitivní. Pro všechny případy zde existuje nápověda programu. Kromě toho na internetových stránkách českého prodejce Eagle (viz. [10]) existuje průvodce vytvářením schémat a pokládáním spojů.

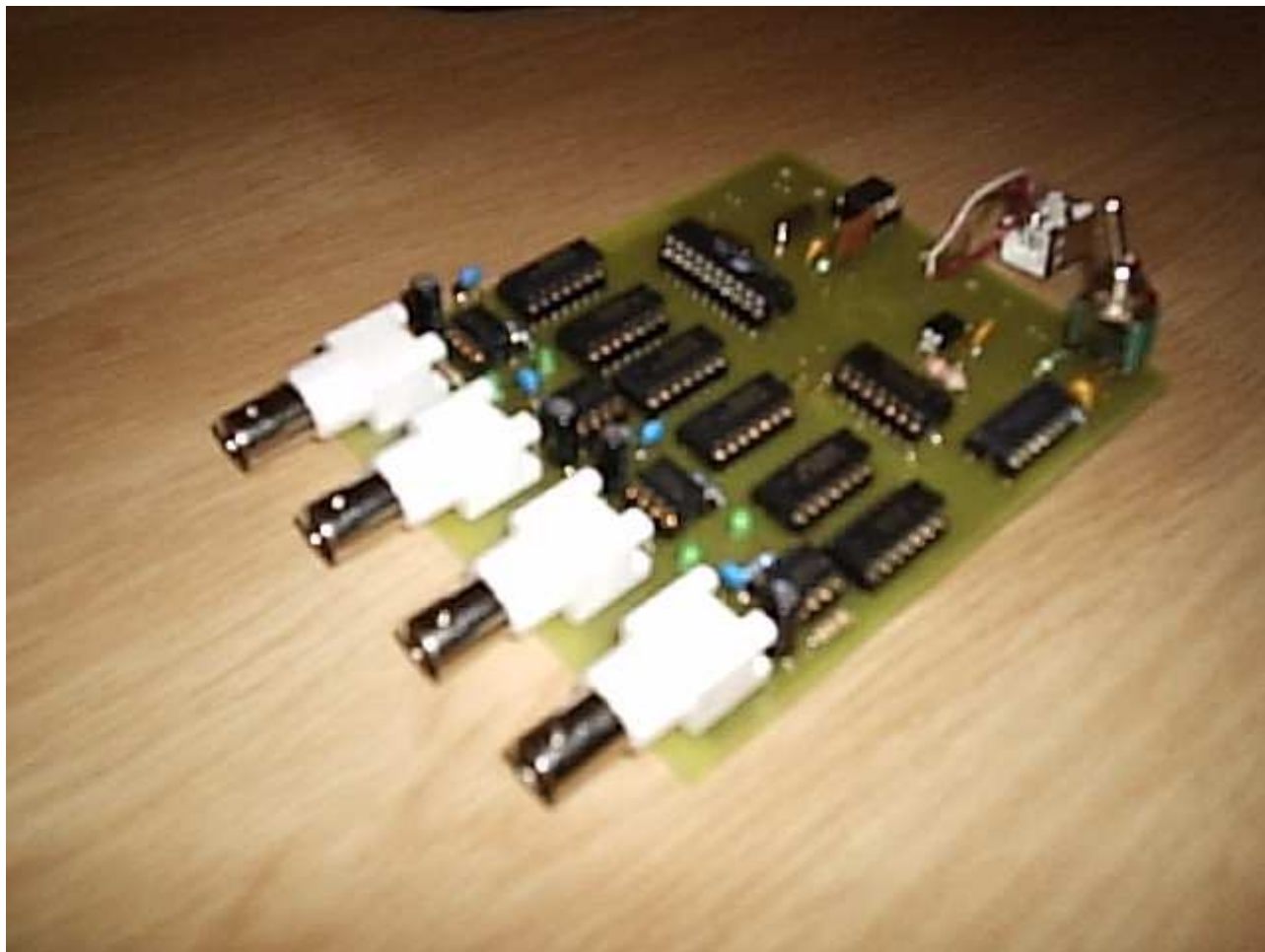
Návrh DPS je do značné míry otázkou zkušeností. Pokusíme se zde však shrnout postřehy, o kterých se domníváme, že by mohly začátečníkovi návrh zjednodušit. Je dobré postupovat ve dvou krocích a to hlavně u složitějších projektů. Nejprve nakreslíme schéma v editoru Schematic. K dispozici máme celou řadu knihoven s nejrůznějšími typy součástek a můžeme vytvářet i vlastní.

Ze Schematic se lze pak pouhým kliknutím přepnout do editoru Board. Eagle automaticky převede schematické značky na pájecí body nebo patice a propojí je tzv. vzdušnými čarami podle schématu. Pak už můžeme kreslit jednotlivé vrstvy. V případě jednostranné desky je to pouze jedna a to strana spojů. Přitom se snažíme, aby vzdálenosti jednotlivých spojů byly co nejkratší. Tomu přizpůsobujeme také rozmístění součástek. Pokud navrhujeme jednostrannou DPS, nevyhneme se zřejmě ani drátovým propojkám. To jsou spoje, které musíme vést stranou součástek. Nicméně jejich počet lze minimalizovat kombinací klasických součástek a součástek pro povrchovou montáž. Pokud budeme i pak nuceni nějaké použít, snažíme se je vhodně umístit a to tak, aby byly co nejkratší. Na obrázku 29 je uveden jako příklad strana spojů DPS čítače impulsů.

4.1.3 Výroba, osazení a oživení DPS

Výroba DPS je předposledním krokem v konstrukci nového zařízení. V této fázi jde o především o to, přenést motiv spojů na měděnou stranu desky kuprextitu. Tuto fázi lze přenechat odborné firmě nebo se pokusit o vlastní výrobu. První variantu doporučujeme jen v případě, že jsme si jisti bezchybností návrhu. Důvodem jsou vyšší ceny takových desek. Výhodou je vysoká kvalita a možnost dalších povrchových úprav takových desek. Někteří výrobci také umožňují osazení součástek. Oproti tomu vlastní výroba poskytuje dostatečnou kvalitu za mnohem nižší cenu. Existuje mnoho internetových stránek a diskusních fór zabývajících se výrobou DPS v domácím prostředí (např. [11]). Uvědomujeme si, že se nejedná o seriózní zdroj informací, ale právě zde najdeme to nejdůležitější – praktické zkušenosti elektroniků.

Ať už se rozhodneme svěřit výrobu odborné firmě nebo si pomoci vlastními silami, měli bychom ji před osazením (pokud jej provádíme sami) zkontrolovat. Nejprve vizuálně a pak proměřit multimetrem. Při osazování postupujeme tak, že nejprve osadíme a zapájíme konektory a pasivní součástky. Takto osazený čítač připojíme k USB. Dioda na konektoru K3 zatím svítit nebude. Při této příležitosti zkontrolujeme všechny vývody, na kterých má být napájecí napětí. Je-li vše v pořádku, odpojíme od USB, osadíme a zapájíme také ostatní součástky ze schématu na obrázku 28. Zvláště při pájení FT232BM jsme opatrní, abychom nechtěně cínem nespojili sousední vývody. Poté vizuálně zkontrolujeme lupou a připojíme opět k USB. Je-li vše v pořádku, OS rozpozná připojení nového zařízení a začne s instalací ovladačů (viz. kapitola dále). Po instalaci se rozsvítí LED připojená ke konektoru K3. Pokud ne, opět zkontrolujeme zapojení okolo IO a jeho zapájení. Nakonec ověříme, je-li za vratnou pojistkou POL1 skutečně +5 V. Ujistíme se také, že je také přivedeno tam kam má. Pokud ano, osadíme zbytek součástek a tím by mělo být oživení u konce.

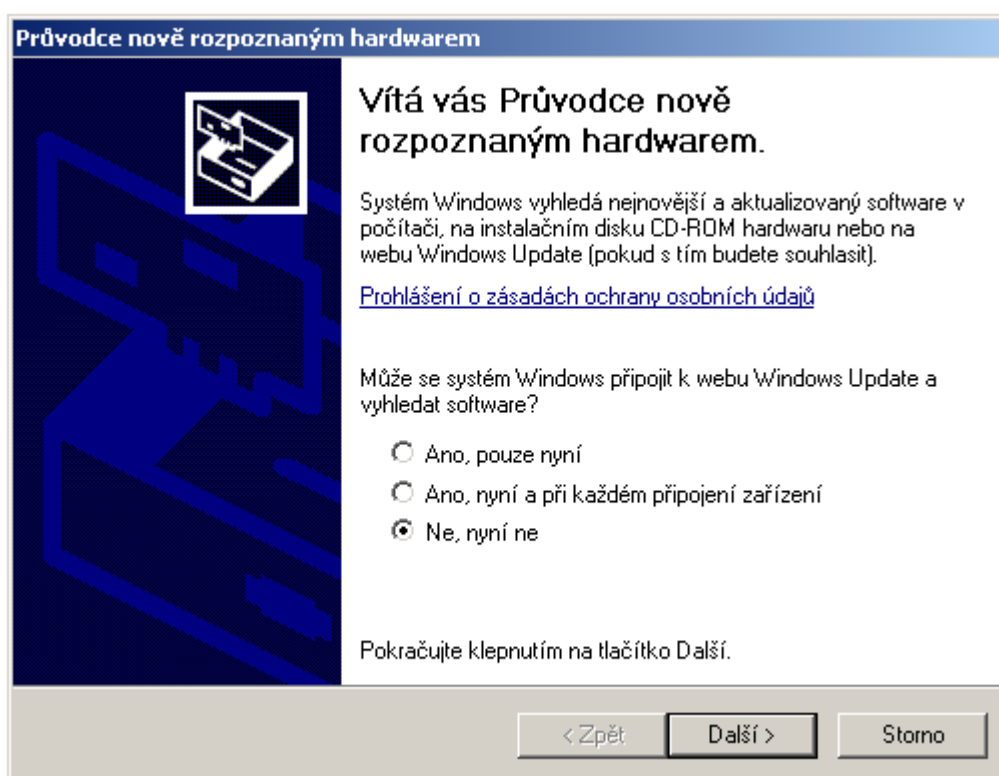


Obr. 30: Prototyp čítače impulsů

4.2 Připojení zařízení s FT232BM k počítači

4.2.1 Instalace ovladačů

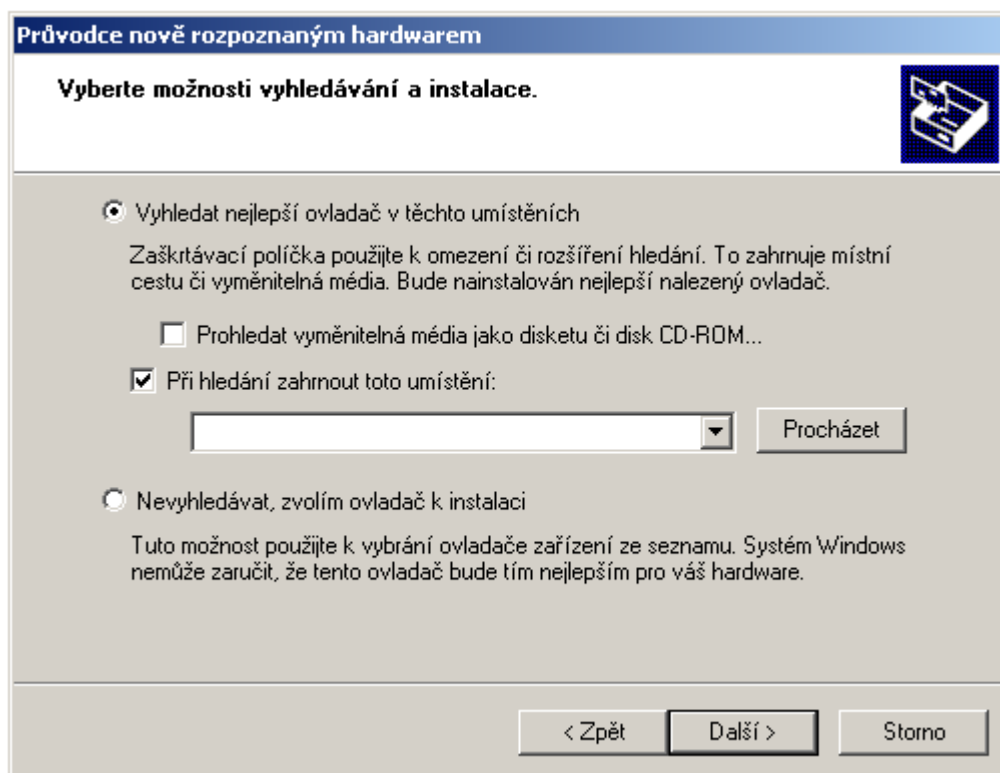
Integrovaný obvod FT232BM podporuje Plug&Play, to znamená, že po připojení k portu USB jej operační systém automaticky rozpozná. Poznamenejme, že před prvním připojením je třeba mít staženy a připraveny ovladače pro daný OS. Ovladače najdeme na webových stránkách FTDI. Máme-li vše připraveno, můžeme připojit čítač k USB. V případě OS Windows XP se ihned spustí Průvodce instalací nově rozpoznaného hardwaru. Průvodce se dotáže, zda chceme najít ovladače pro zařízení pomocí služby Windows Update (obr. 31). Zvolíme *Ne, nyní ne* a stiskneme tlačítko *Další*.



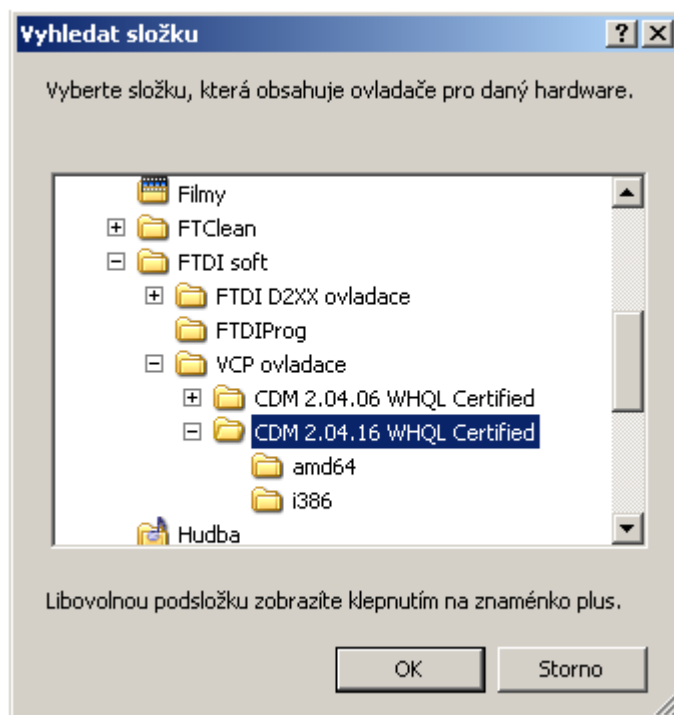
Obr. 31: Ukázka z instalace čítače impulsů (první krok)

Protože OS Windows XP neobsahuje ovladače zařízení FTDI, je třeba v dalším kroku zvolit *Instalovat ze seznamu* či *daného umístění* a kliknout na tlačítko *Další*. Poté jsme vyzváni k vyhledání umístění ovladače. K tomuto účelu zvolíme *Vyhledat nejlepší ovladač v těchto umístěních* a dále *Při hledání zahrnout toto umístění*. Stiskem tlačítka *Procházet* vyvoláme další nabídku, pomocí které vyhledáme složku s ovladači pro čítač impulsů. V našem případě jsou to ovladače virtuálního sériového portu k FT232BM. Jsou-li ve složce *.inf soubory obsahující informace o instalaci zařízení, tlačítko *OK* přejde do aktivního stavu a můžeme tak postoupit dále. Máme-li

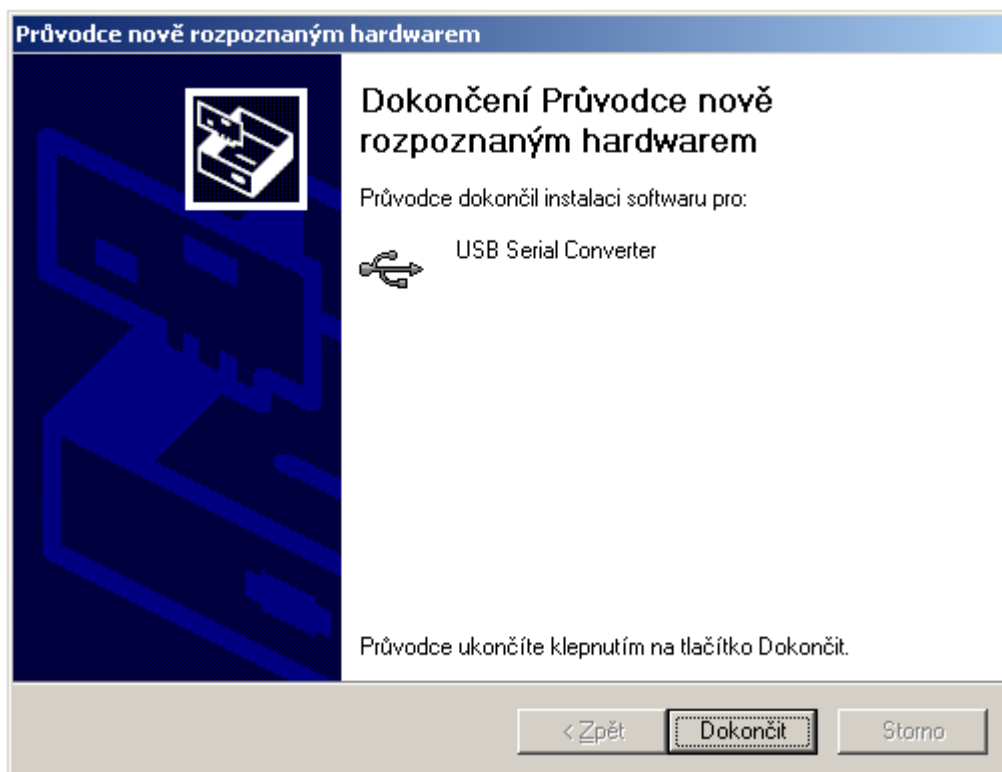
složku vybranou, začnou se automaticky instalovat ovladače zařízení. Proběhne-li vše v pořádku, zobrazí se nám okno na obrázku 34. Stiskneme tlačítko *Dokončit*. Tím máme dokončenu jednu část



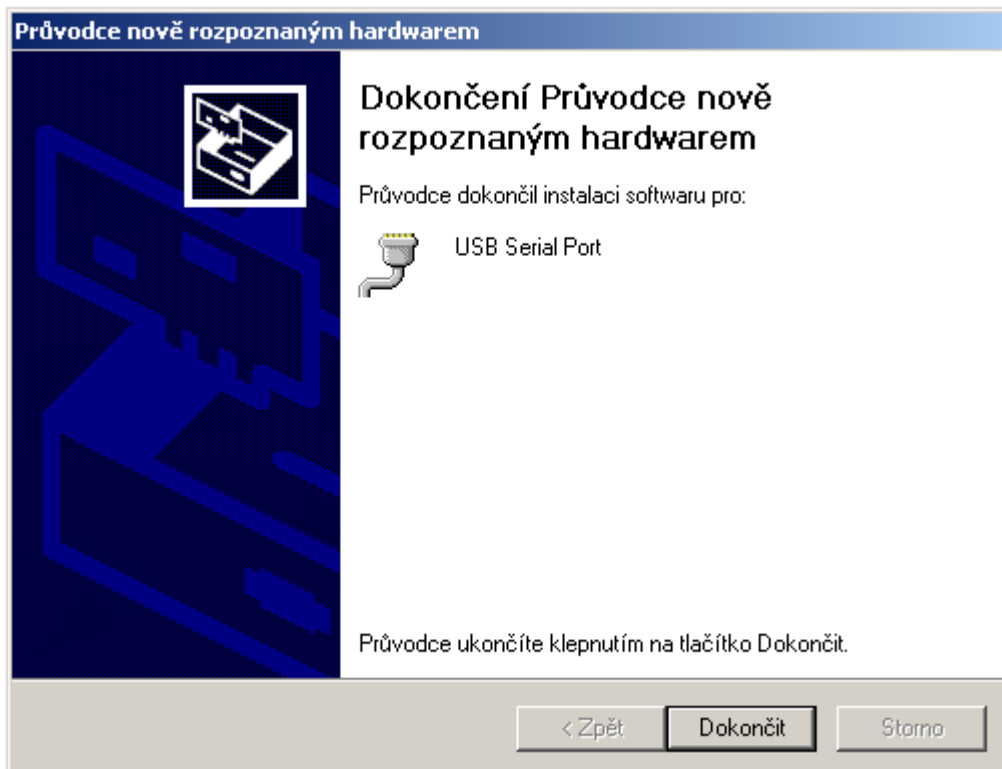
Obr. 32: Ukázka z instalace čítače impulsů (třetí krok)



Obr. 33: Ukázka z instalace čítače impulsů (čtvrtý krok)



Obr. 34: Ukázka z instalace čítače impulsů (pátý krok)



Obr. 35: Ukázka z instalace čítače impulsů (šestý krok)

instalace. Ještě je totiž třeba instalovat USB sériový port.


Jeho instalace probíhá zcela totožně. OS tedy najde další zařízení a spustí průvodce z obrázku 31. Dále postupujeme shodně dle obrázků 32 a 33. Nakonec jsme vyzváni vybrat příslušnou složku s ovladači. Všimněte si, že nejprve bylo instalováno zařízení USB Serial Converter a potom USB Serial Port. Stiskem tlačítka *Dokončit* ukončíme průvodce a tím také instalaci čítače impulsů. OS nám tuto skutečnost oznámí zprávou o úspěšné instalaci a správné funkci právě instalovaného zařízení.

4.2.2 Programování EEPROM

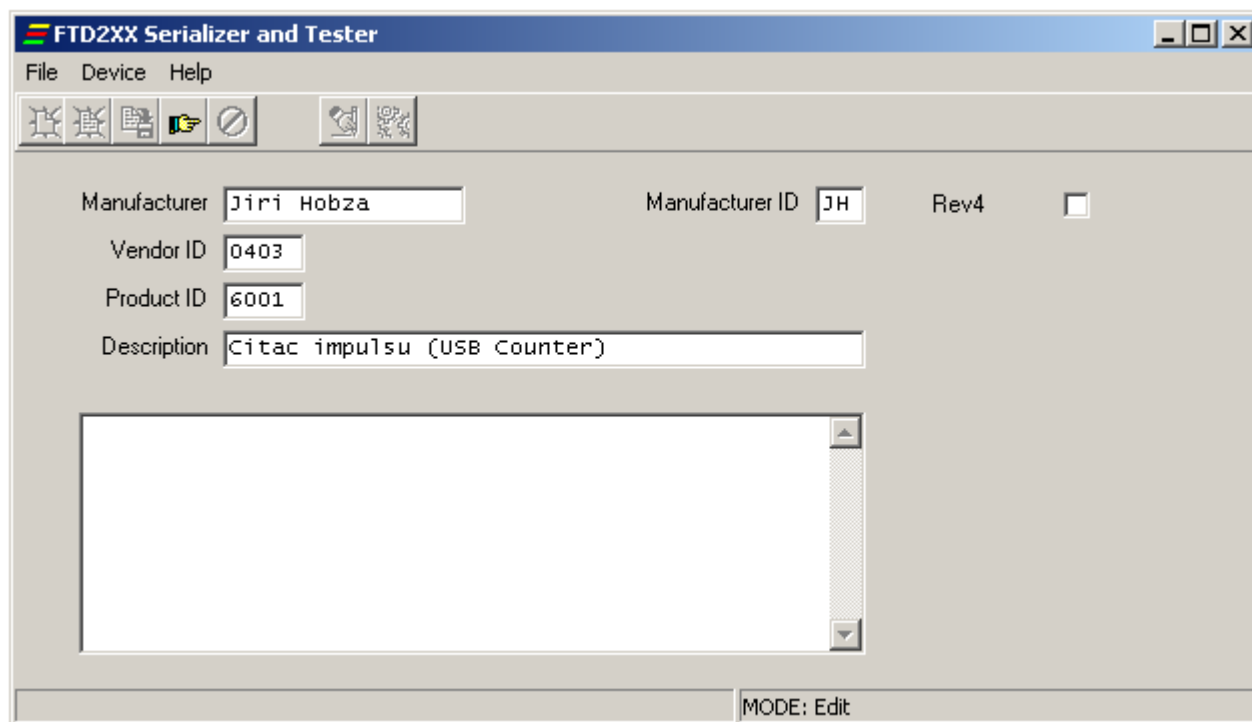
Jak již bylo uvedeno v teoretické části práce, je z mnoha důvodů výhodné používat spolu s obvodem FTDI také paměť EEPROM. Výrobce čipu FT232BM ji umožňuje naprogramovat přímo v aplikaci pomocí dvou nástrojů. Jsou to

- Ftd2xxst - jednoduchý nástroj pro programování EEPROM, není nutná instalace
- MProg - nahrazuje Ftd2xxst, podporuje nejnovější čipy, není nutná instalace

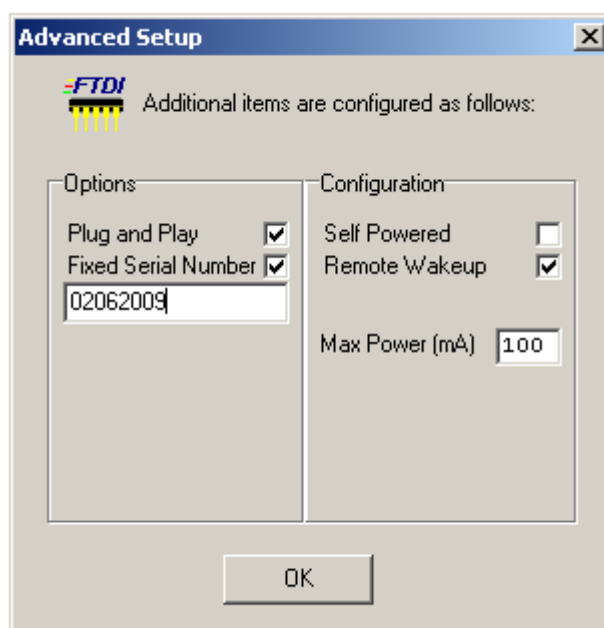
Oba dva programy jsou zdarma ke stažení na stránkách FTDI Chip. Soubor si stáhneme jako .zip archiv. Po rozbalení ve složce objevíme samotný spouštěcí soubor, soubor dynamicky linkované knihovny FTD2XX.dll a soubor registrační položky ftdi. Ovšem dle našich zkušeností nejsou tyto soubory ke správné činnosti třeba, nicméně přidáme-li registrační položku ftdi do registru, bude první spuštění programu doprovázeno rozpoznáním připojeného zařízení a jednotlivé položky v okně budou patřičně vyplněny. Pokud tak neučiníme, nevádí ale budeme muset postupovat následujícím způsobem.

Program se otevře v editačním módu, přičemž položky Manufacturer, Manufacturer ID, VID, PID a Description budou nevyplněné. Vyplníme je tedy podle obrázku níže. Pole popisující výrobce a výrobek lze vyplnit libovolně. Nicméně pro VID a PID musíme použít údaje firmy FTDI Chip. Pokud bychom chtěli použít vlastní, musíme složit poplatek organizaci USB (viz. [1]). Po vyplnění všech údajů musíme ještě kliknout do informačního pole níže. Tím se aktivuje ikona pro pokročilá nastavení . Zde jsou nejdůležitější políčka v nabídce *Configuration*. Pokud budeme aplikaci napájet z vlastního zdroje pak zaškrtneme *Self Powered*. V opačném případě ponecháme nezaškrtnuto. V tomto případě musíme zvolit maximální odebíraný proud v miliampérech. Maximální hodnota je 500mA, ale výrobce doporučuje vyplnit jen 490mA. Políčko *Remote Wakeup* slouží pro aktivaci čipu z režimu nízkého odběru proudu, pomocí vývodu *RI*. Lze ponechat zaškrtnuto, což je výchozí nastavení. V nabídce *Options* zaškrtneme políčko *Plug&Play*. Je-li



zaškrtnuto, je povolen tento typ enumerace také pod OS Windows 98/ME. Zaškrtnutím políčka *Fixed Serial Number* lze nastavit pevné sériové číslo připojeného zařízení. Číslo může být například aktuální datum (viz. obr. níže). Stiskem tlačítka *OK* uložíme nastavení a vrátíme se do předchozího



Obr. 36: Vyplnění enumeračních údajů v programu Ftd2xxst



Obr. 37: Pokročilá nastavení enumeračních údajů

okna. Tím se aktivuje ikona uložení . Jejím stisknutím uložíme naše nastavení a nyní už zbývá jen zapsat je do paměti. To provedeme ikonou . Všechny tyto úkony lze také provést přes příslušné nabídky v menu programu.

Velmi podobné a intuitivní ovládání má také program MProg. Veškeré volby jsou přitom obsaženy v jednom okně a při ukládání nastavení můžeme zvolit jejich název a umístění. Soubor je uložen ve formátu *.ept (eEPROM pROGRAM tEMPLATES). Výhodou MProg je podpora nejnovějších integrovaných obvodů FTDI.

4.2.3 Odinstalování ovladačů FTDI

Pokud je třeba odinstalovat ovladače FTDI ze systému, doporučujeme nečinit tak ručně, tj. například pomocí *Správce zařízení*. Tento způsob totiž neodstraní klíče z registru a nevymaže některé další soubory (především *.pnf a *.inf) ze složek Windows/System32 a Windows/System32/Drivers. K tomuto účelu jsou vhodnější pomůcky, které lze volně stáhnout z webových stránek firmy FTDI Chip. Jedná se o FTClean a FTDIUNIN. Použití těchto pomůcek je snadné a přitom dokonale odstraní všechny soubory vzniklé instalací ovladačů včetně klíčů registru.



Obr. 38: Pomůcka FTClean pro odinstalování ovladačů FTDI Chip

Při spuštění FTClean se nám zobrazí okno z obrázku 38. Zde můžeme zvolit, která zařízení FTDI mají být odinstalována a to na základě VID a PID. Je-li jich připojeno k PC několik, je nutné tyto

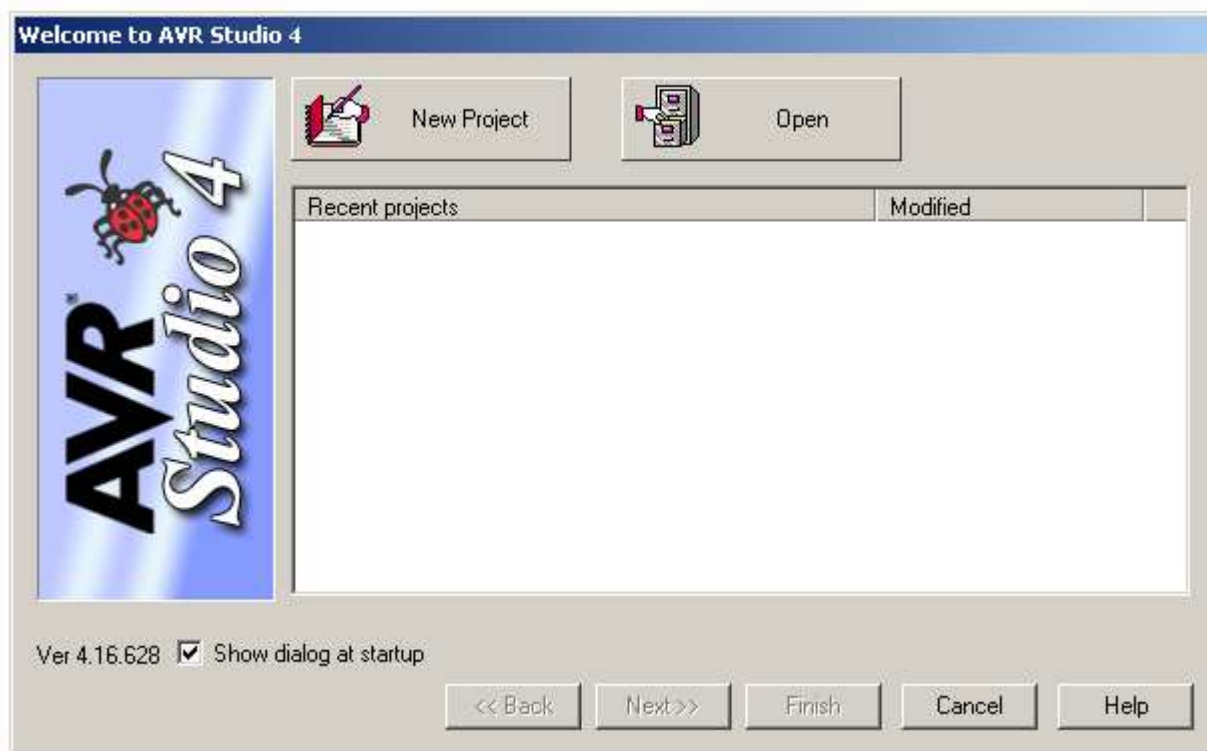
identifikátory specifikovat. Pokud je připojeno pouze jedno, lze ponechat výchozí nastavení (zařízení s VID firmy FTDI). Nyní můžeme stisknout tlačítko *Clean System*. Tímto se spustí další pomůcka a to FTUNIN, která se již samotná postará o odstranění všech souborů. Pomůcku ukončíme tlačítkem *Exit*.



Obr. 39: Okno pomůcky FTUNIN

4.3 AVR Studio, program pro mikrokontrolér, formát Intel HEX

4.3.1 Vývojové prostředí AVR Studio

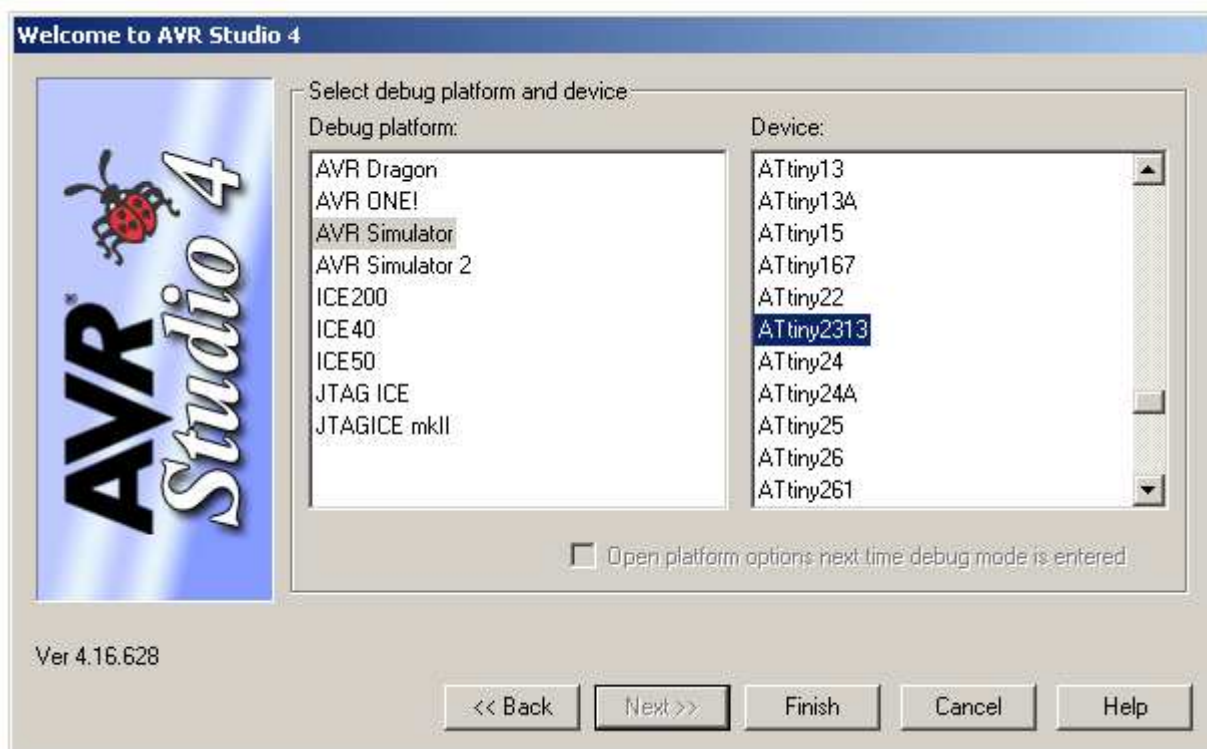


Obr. 40: Založení nového projektu ve vývojovém prostředí Atmel AVR Studio 4

Program ATMEL AVR Studio je vývojové prostředí, které výrobce mikrokontrolérů dodává ke svým produktům. Na internetových stránkách ATMEL je po zaregistrování ke stažení zcela zdarma.



Obr. 41:



Obr. 42:

Čtenář však nalezne instalační soubor na doprovodném CD. Termín vývojové prostředí znamená, že AVR Studio v sobě sdružuje editor, překladač a simulátor. Zvláště simulátor ocení každý uživatel při vývoji svých aplikací. Nebudeme se zde zabývat podrobným popisem prostředí AVR Studia, práce s ním je totiž velmi intuitivní. Pouze si ukážeme založení nového projektu, které je poměrně snadné.

Spustíme AVR Studio, otevře se nám dialogové okno jako na obrázku 40. Stiskneme tlačítko *New project*. V následujícím okně (obrázek 41) si můžeme zvolit, zda chceme psát kód v assembleru (Atmel AVR Assembler) nebo v jazyce C (AVR GCC). Dále vyplníme jméno projektu a souboru *.asm a pokračujeme kliknutím na tlačítko *Next*. V dalším okně (viz. obr. 42) zvolíme vývojovou platformu (pro naše účely AVR Simulator) a nakonec typ zařízení (ATtiny2313). Kliknutím tlačítka *Finish* je založení nového projektu hotové a můžeme začít psát zdrojový kód.

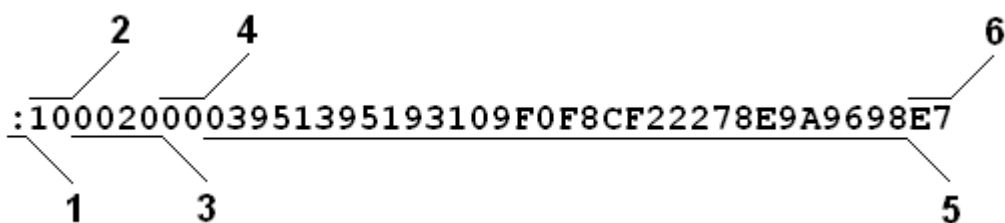
4.3.2 Soubory formátu Intel HEX

K programování mikrokontroléru je nutná znalost souborů formátu Intel HEX, tedy formátu do kterého AVR Studio kompiluje zdrojový kód assembleru nebo C. Tento formát je určený k přenosu binární informace do mikroprocesorů, paměti EEPROM a jiných čipů. Jedná se o jeden z nejstarších formátů, protože se používá už od 70. let dvacátého století. Existují tři typy těchto formátů:

- Intel HEX 8 – bitový
- Intel HEX 16 – bitový
- Intel HEX 32 – bitový

Každý řádek formátu se skládá ze šesti částí, z nichž každá má svůj význam (viz. tabulka 9). Struktura formátu v jakém jsou data uložena je zřejmá z obrázku 43.

Při programování je třeba vytvořit algoritmus, který tato data ze souboru *.hex extrahuje a bude s nimi dále pracovat. Každý programovací jazyk obsahuje techniky pro čtení ze souboru. V případě



Obr. 43: Struktura řádku v souboru formátu Intel HEX

Visual Basic jde o stejnou techniku, která se používá pro čtení souborů binárního typu. Samozřejmě samotné načtení dat nestačí. Protože zkompileovaný soubor je v hexadecimálním tvaru, musí být data před použitím ještě převedena do tvaru decimálního. V části práce týkající se popisu zdrojových kódů ukázkových programů, bude také vysvětleno načtení takového souboru za účelem programování mikrokontroléru. Tuto ukázkou lze považovat za předlohu při vytváření podobně zaměřeného kódu, ale v jiném programovacím jazyce.

Pozice v řádku	Význam	Popis
1	začátek řádku	jeden znak uvozující začátek řádku
2	počet bajtů	hexadecimální číslo udávající počet datových bajtů v řádku
3	adresa	čtyři číslice určující absolutní adresu v paměti
4	typ zápisu	dvě číslice definující typ zápisu <ul style="list-style-type: none"> • 00 – datové bajty • 01 – konec záznamu v souboru • 02 – rozšířená část adresy • 03 – začátek rozšířené adresy • 04 – rozšířená přímá adresa • 05 – začátek přímé adresy
5	data	datové bajty, jejich počet je daný číslem ve druhé pozici
6	kontrolní součet	dvě číslice udávající sumu všech číslic z pozic 2 až 5

Tab. 11: Význam jednotlivých částí řádku v souboru formátu Intel HEX

4.3.3 Popis programu pro mikrokontrolér

V průběhu práce jsme již několikrát naznačili, že čtení impulsů detekovaných vstupními TTL obvody, bude neustále opakující se smyčka. Tuto smyčku bude podle programu uloženého v paměti vykonávat MCU. Nyní si podrobněji přiblížíme podstatu této smyčky. Je mou milou povinností zde zmínit, že spoluautorem velké části tohoto programu je vedoucí práce Prof. Miloslav Dušek.

Nejdůležitější částí řídicího programu pro mikrokontrolér je čítací smyčka, která zajišťuje čtení stavů výstupů na klopných obvodech. Jedna smyčku tvoří čtyři cykly. To proto aby se mezi jednotlivé cykly postupně rozložila dekrementace čítače počtu smyček programu (tj. měření celkového času), ale také proto, aby celkový čas reprezentovaný počtem smyček programu (pro který jsou vymezeny 3B) mohl být rozumně dlouhý. Jeden čtecí cyklus zabere 20 hodinových impulsů, tedy při hodinovém kmitočtu 20MHz trvá 1 μ s (to odpovídá vzorkovací frekvenci 1MHz). Jedna smyčka (4 čtecí cykly) trvá 4 μ s. tj celkový čas čítání může být až $16777215 * 4\mu s = 67,1s$. Ve

skutečnosti je však doba čítání o asi 2 μ s delší než doba nastavena. Než dojde k zastavení programu, proběhnou ještě dva čtecí cykly (protože dekrementace a test počtu smyček jsou rozloženy do několika čtecích cyklů).

Přetečení dané vysokým počtem příchozích impulsů nelze, vzhledem k nedostatku času uvnitř jednoho čtecího cyklu, úplně ohlídat. Proto je třeba dodržet následující omezení. Součin doby čítání a střední frekvence signálu by měl být menší než 2²⁴. Předpokládá se, že v laboratoři bude vždy měřen signál se střední frekvencí menší než 250kHz. Tak bude podmínka splněna vždy.

Začínáme návěstím *read* (řádek 5). Na dalších řádcích vynulujeme registry *Rzero* a *Rone*. Do dvojitého registru *Y* (stačí pouze dolní bajt) načteme symbol *shift*, který představuje začátek vyhrazené paměti (řádek 10). Samotné smazání probíhá na řádcích 14 až 16. Nejprve načteme do pomocného 16-bitového registru obsah registru, který jsme před tím vynulovali a posuneme adresu o jednu nahoru. Poté dekrementujeme registr *R16*, který slouží jako počítadlo. Pokud není registr vynulován, vracíme se na návěstí *loopC*. Na dalších řádcích nahrajeme do pomocných registrů potřebné konstanty. Na řádce 22 nastavíme směr vývodů portu *B* a na dalším pak vynulujeme klopné obvody.

Nyní následuje první ze čtyř čítacích smyček. Na jejím začátku načteme do 16-bitového registru stav celého portu *B* (řádek 3). Dále resetujeme klopné obvody a obnovíme normální režim. Prázdné instrukce na řádcích jsou nutné k zachování délky smyčky (20 hodinových cyklů). Pro uložení počtu načítaných impulsů je vyhrazeno místo v paměti o velikosti třech bajtů. Proto je na dalších řádcích postupně do příslušných registrů načten stav portu *B*. K těmto registrům se pak přičte obsah registrů *Rone* a *Rzero*. U posledního pak včetně příznaku *C*. Posledními instrukcemi přičtené hodnoty opět uložíme na své místo v paměti. Samozřejmě s příslušným posunutím adresy. Pro další vysvětlení postačí jen popis v kódu.

```
; Main loop for reading states of flip flop circuits during time t
; results at RAM from address "shift" (length: 16*3 bytes)
```

```
read:   CLR Rzero           ; zero to Rzero
        CLR Rone
        INC Rone           ; one to Rone

        LDI YL,shift       ; clear memory space
        LDI R16,48         ; 3 * 16 bytes
loopC:  ST Y+,Rzero         ; clear byte
        DEC R16
        BRNE loopC
```

```

LDI Rnorm,norm
LDI Rdirs,dirs
LDI Rshift,shift      ; set Rshift to memory offset

OUT DDRB,Rdirs      ; set directions of I/O port

OUT PORTB,Reras      ; start erasing flip-flop circuits
OUT PORTB,Rnorm      ; stop erasing flip-flop circuits
NOP                  ; necessary to set port pins

CLI                  ; forbid interrupts

;loop: (4*20 clock ticks)

loop:      ; 1st round

OUT PORTB,Rnorm      ; stop erasing flip-flop circuits [1]
OUT PORTB,Reras2     ; start erasing flip-flop circuits [1]
IN  YL,PINB         ; read data from flip-flop circuits [1]

LD  b1,Y            ; read 1st byte [2]
LDD b2,Y+shift2     ; read 2nd byte [2]
LDD b3,Y+shift3     ; read 3rd byte [2]

ADD b1,Rone         ; 1st byte + 1 [1]
ADC b2,Rzero        ; 2nd byte + C [1]
ADC b3,Rzero        ; 3rd byte + C [1]

ST  Y,b1            ; write 1st byte [2]
STD Y+shift2,b2     ; write 2nd byte [2]
STD Y+shift3,b3     ; write 3rd byte [2]

SUB cycles1,Rone    ; decrement 1st byte of cycles [1]
SBC cycles2,Rzero   ; decrement 2nd byte of cycles [1]

; 2nd round

OUT PORTB,Rnorm      ; stop erasing flip-flop circuits [1]
OUT PORTB,Reras1     ; start erasing flip-flop circuits [1]
IN  YL,PINB         ; read data from flip-flop circuits [1]

SBC cycles3,Rzero   ; decrement 3rd byte of cycles [1]

```

```

BRCS loopE          ; if zero then stop          [1]-cont [2]-jump

LD  b1,Y            ; read 1st byte             [2]
LDD b2,Y+shift2    ; read 2nd byte             [2]
LDD b3,Y+shift3    ; read 3rd byte             [2]

ADD b1,Rone        ; 1st byte + 1              [1]
ADC b2,Rzero       ; 2nd byte + C              [1]
ADC b3,Rzero       ; 3rd byte + C              [1]

ST  Y,b1           ; write 1st byte            [2]
STD Y+shift2,b2    ; write 2nd byte            [2]
STD Y+shift3,b3    ; write 3rd byte            [2]

; 3rd round

OUT PORTB,Rnorm    ; stop erasing flip-flop circuits [1]
OUT PORTB,Reras2   ; start erasing flip-flop circuits [1]
IN  YL,PINB       ; read data from flip-flop circuits [1]

NOP                ; for proper timing          [1]
NOP                ; for proper timing          [1]

LD  b1,Y            ; read 1st byte             [2]
LDD b2,Y+shift2    ; read 2nd byte             [2]
LDD b3,Y+shift3    ; read 3rd byte             [2]

ADD b1,Rone        ; 1st byte + 1              [1]
ADC b2,Rzero       ; 2nd byte + C              [1]
ADC b3,Rzero       ; 3rd byte + C              [1]

ST  Y,b1           ; write 1st byte            [2]
STD Y+shift2,b2    ; write 2nd byte            [2]
STD Y+shift3,b3    ; write 3rd byte            [2]

; 4th round (the last one)

OUT PORTB,Rnorm    ; stop erasing flip-flop circuits [1]
OUT PORTB,Reras1   ; start erasing flip-flop circuits [1]
IN  YL,PINB       ; read data from flip-flop circuits [1]

LD  b1,Y            ; read 1st byte             [2]

```

```

LDD b2,Y+shift2      ; read 2nd byte          [2]
LDD b3,Y+shift3      ; read 3rd byte          [2]

ADD b1,Rone          ; 1st byte + 1          [1]
ADC b2,Rzero         ; 2nd byte + C          [1]
ADC b3,Rzero         ; 3rd byte + C          [1]

ST Y,b1              ; write 1st byte        [2]
STD Y+shift2,b2      ; write 2nd byte        [2]
STD Y+shift3,b3      ; write 3rd byte        [2]

RJMP loop            ; repeat from "loop"     [2]

```

To byla čítací část ovládacího programu. Nyní si ukážeme jak naprogramovat MCU ke komunikaci s počítačem. Opět postačí popis programu přímo v kódu.

```

;=====Inicializace jednotky USART=====
;
Usart_Init:          ;nastavi prenosovou rychlost
LDI Reg, Baud_9600
OUT UBRR, Reg
;povoli prijimac a vysilac
SBI UCSRB, 3
SBI UCSRB, 4
;nastavi ramec prenosu {8 bitu,1 stop-bit}
SBI UCSRC, 1
SBI UCSRC, 2
;navrat z podprogramu
RET
;
;=====Obsluha vysilani=====
;
Usart_Vysilac:      ;ceka na vyprazdneni vysilaciho bufferu
SBIS UCSRA, UDRE
RJMP Usart_Vysilac
;pak posle data opet na UART
OUT UDR, Data
;navrat z podprogramu
RET
;
;=====Obsluha prijmu=====

```

```

;
Usart_Prijimac_B:      ;ceka na prijata data
SBIS UCSRA, RXC
RJMP Usart_Prijimac_B
;precte je a ulozi do registru Data
IN Data,UDR
;a potom do bufferu
ST Z+, Data
;testuje konec bufferu
CPI ZL, string + 5
;pokud neni konec, opakuje
BRNE Usart_Prijimac_B
;pokud ano nacte znovu zacatek bufferu
;LDI ZL, string
;navrat z podprogramu
RET
;
;=====Obsluha vysilani=====
;
Usart_Vysilac_B:     ;ceka na vyprazdneni vysilaciho bufferu
SBIS UCSRA, UDRE
RJMP Usart_Vysilac_B
;pak nacte data z bufferu
LD Data, Z+
;a odesle je na port
OUT UDR, Data
;testuje konec bufferu
CPI ZL, string + 32
;pokud neni konec, opakuje
BRNE Usart_Vysilac_B
;pokud ano nacte znovu zacatek bufferu
;LDI ZL, string
;navrat z podprogramu
RET
;
;=====Obsluha vysilani retezce=====
;
Usart_Retezec:      ;ceka na vyprazdneni vysilaciho bufferu
SBIS UCSRA, UDRE
RJMP Usart_Retezec
;pak nacte data z bufferu
LPM Data, Z+

```

```

;testuje konec retezce
CPI Data, '/'
;pokud je konec, skonci
BREQ Ret_
;pokud ne, odesle na port a opakuje
OUT UDR, Data
BRNE Usart_Retezec
;navrat z podprogramu
Ret_:          RET


```

4.4 Doprovodné programy pro počítač

4.4.1 Integrované vývojové prostředí Visual Basic 6.0

Visual Basic 6.0 není pouze programovací jazyk, ale také integrované vývojové prostředí (IDE). To znamená, že v něm můžeme vyvíjet, spouštět, testovat a odlaďovat svoje aplikace. Samotným vývojovým prostředím se nebudeme zvláště zabývat. K tomuto účelu bylo napsáno mnoho knih (např. [3]). Ty poskytnou zájemci o tento programovací jazyk dostatečnou průpravu. Upozorníme pouze na několik skutečností týkajících se tématu práce.

Jednou z nich je skutečnost, že vývojové prostředí Visual Basic 6.0 umožňuje uživateli přístup na sériový port prostřednictvím ovládacího prvku *MSComm*. Prvek *MSComm* není v nově založeném projektu obsažen a proto jej do něj musíme přidat. Učiníme tak pomocí nabídky *Project - Components*. Zde v záložce *Controls* vyhledáme a zaškrtneme *Microsoft Comm Control 6.0*.

V panelu nástrojů se přidání této komponenty projeví objevením ikony telefonu . Dále poznamenejme, že tento prvek nemůže současně pracovat s více porty. Pokud tedy potřebujeme obsluhovat dva nebo více portů najednou, musíme pracovat se stejným počtem prvků *MSComm*.

Následující programy byly vytvořeny jako součást diplomové práce. Jejich úkolem je demonstrovat funkčnost čítače impulsů a pomoci případným zájemcům při vývoji vlastních aplikací využívajících sériový port nebo USB. V dalším textu popíšeme jejich ovládání spolu s nejdůležitějšími částmi kódu, protože popis celého kódu by byl příliš dlouhý (kompletní zdrojové kódy jsou uvedeny na doprovodném CD). Díky těmto ukázkám (neobsahují složité programovací techniky), mohou být tyto programy snadno upraveny a přepsány do jiného programovacího jazyka.

4.4.1.1 Zdrojový kód Counter-Signal

První důležitou procedurou je obsluha události kliknutí na tlačítko *Command1*. Jedná se o tlačítko s popisem *Připojit na port*. Pokud by se uživatel pokusil připojit na port, ale žádný by před tím nevybral, bude to mít za následek chybu, která ukončí aplikaci. Taková situace nesmí nikdy nastat, proto tu jsou příkazy pro zachytávání chyb. Příkladem budiž řádek 3. Pokud program zachytí jakoukoli chybu, odskočí na dané návěští, kde vyvolá zprávu pro uživatele (řádek 31 až 34). Následuje blok řízení programu pomocí If ... Else ... End If. Pokud ještě není port otevřen, začne se provádět cyklus For ... Next (řádek 5 až 15). Jeho úkolem je zjistit, který port uživatel vybral (řádek 6), a podle toho příslušný port inicializovat (řádek 7). Příkazem na řádku 16 pak tento port otevře. Další příkazy změní popis tlačítka a aktualizují stavový řádek. Na řádcích 24 a 25 nastavíme a spustíme časovač. Časovač je objekt, jehož hlavní vlastností je interval. Udává se v milisekundách a je to doba od skončení procedury časovače do jejího dalšího začátku. Z toho plyne, že pokud je časovač aktivní, vykonávají se jeho příkazy znovu a znovu. V našem případě procedura časovače obsluhuje události vstupů každou milisekundu, aby byla zaručena okamžitá odezva. Událostmi myslíme přečtení jejich logické hodnoty na vstupech CTS, DSR a DCD a podle výsledku také změnu popisu štítků pod odpovídajícími tlačítky. Následuje druhá část bloku If ... Else ... End If. Ta odpovídá situaci, kdy je port otevřen a další kliknutí na tlačítko jej uzavře (*Odpojit od portu*).

```
1. Private Sub Command1_Click()  
2.     'v pripade chyby odskoci na navesti chyba_portu  
3.     On Error GoTo Chyba_portu  
4.     If MSComm1.PortOpen = False Then  
5.         For port = 1 To 20  
6.             If Combo1.Text = com(port) Then  
7.                 MSComm1.CommPort = port  
8.                 'upozorni, pokud je nektery port nahodou otevreny  
9.                 If hledej_port(port) Then  
10.                    List1.AddItem ">> vybrán port " + CStr(port) + "..."  
11.                Else  
12.                    List1.AddItem ">> port není dostupný!"  
13.                End If  
14.            End If  
15.        Next port  
16.        MSComm1.PortOpen = True  
17.        Command1.Caption = "Odpojit od portu"  
18.        List1.AddItem radek + "Připojeno..."  
19.        'vychozi hodnota pro vystupy
```



```

20.     Label1.Caption = "H"
21.     Label2.Caption = "H"
22.     Label3.Caption = "L"
23.     'nastavi casovac
24.     Timer1.Interval = 1
25.     Timer1.Enabled = True
26.     Else
27.         MSComm1.PortOpen = False
28.         Command1.Caption = "Připojit na port"
29.         List1.AddItem radek + "Port byl odpojen..."
30.     End If
31. Chyba_portu:
32.     If Combo1.Text = "Porty" Then
33.         msg = MsgBox("Nejprve vyber port!", vbExclamation + vbOKOnly)
34.     End If
35. End Sub

```

Procedura tlačítka *Hledat porty* obsahuje pouze jeden řádek a to volání procedury spuštění okna. Další důležitou procedurou je *Form_Load()*, která se provede ihned po spuštění programu. Řádky 4 až 6 obsahují cyklus For ... Next, který plní pole com(port) řetězci tvaru "COM + číslo portu". Pole com(port) figuruje také v proceduře Command1_Click() popsané výše. Příkaz na řádku 7 nastaví sériový port, přičemž význam je následující. "9600" označuje přenosovou rychlost, "n" bezparitní přenos, "8" počet datových bitů a "1" počet stop bitů. Další řádky obsluhují seznam portů pod tlačítkem *Hledat porty*. Cyklus využívá funkce *hledej_port()*. Při volání funkce musíme na rozdíl od procedury udat také všechny její argumenty.

```

1. Private Sub Form_Load()
2.     radek = ">> "
3.     'naplni pole com(i)
4.     For port = 1 To 20
5.         com(port) = "COM" & port
6.     Next port
7.     MSComm1.Settings = "9600,n,8,1"
8.     'prida port do combolist
9.     Combo1.Clear
10.    Combo1.Text = "Porty"
11.    For port = 1 To 20
12.        If hledej_port(port) Then
13.            Combo1.AddItem com(port)

```

```
14.      End If
15.      Next port
16.  End Sub
```

Při zápisu funkce musíme zadat argument funkce (v našem případě proměnná *port* typu *integer*) a její typ. V tomto případě *boolean*. Tato funkce vyhledává volné porty tím, že testuje jejich otevření. Nejprve tedy inicializujeme daný port (viz. řádek 4). Je-li port otevřen (řádek 5), nelze jej přidat do seznamu a tudíž návratová hodnota funkce musí být *false* (řádek 6) a funkce se ukončí. Naopak, je-li port uzavřen testuje se jeho existence otevřením a opětovným zavřením (řádky 9 a 10). Pokud se příkazy provedou, funkce nabude hodnoty *true* a ukončí se. Pokud ne, bude vyvolána chyba, oznamující neplatné otevření portu. V tomto případě je tu příkaz na řádku 3, který způsobí odskočení na návěští *Chyba*, kde funkci přiřadíme hodnotu *false*.

```
1.  Private Function hleddej_port(port As Integer) As Boolean
2.      'funkce hledající volne porty
3.      On Error GoTo Chyba
4.      MSComm1.CommPort = port
5.      If MSComm1.PortOpen = True Then
6.          hleddej_port = False
7.          Exit Function
8.      Else
9.          MSComm1.PortOpen = True
10.         MSComm1.PortOpen = False
11.         hleddej_port = True
12.         Exit Function
13.     End If
14.     Chyba:
15.         hleddej_port = False
16. End Function
```

4.4.1.2 Použití programu Counter-Signal



Obr. 44: okno programu RS - Signal

Program Counter-Signal byl vytvořen za účelem testu UART linek, které lze ovládat přímo z prostředí Visual Basic. Přímou ovladatelnými míníme situaci, kdy uživatel sám určuje logickou hodnotu na daném vývodu (logická 0 nebo 1). Účel je zřejmý - tímto způsobem je možné programovou cestou řídit sběrnici SPI a tím i programování mikrokontroléru ATtiny2313. Pomocí tohoto programu zjistíme průchodnost linek TxD, RTS a DTR jako výstupů a CTS, DSR a DCD jako vstupů. Využití této pomůcky najde také při vývoji podobně zaměřených aplikací a to nejen připojených k USB, ale i k sériovému portu.

V horní části vidíme rámečky s výstupními a vstupními tlačítky (viz. obr. 44). Vstupní tlačítka nejsou aktivní, protože hodnotu na těchto vývodech lze pouze číst. O stavu těchto vývodů jsme informováni popisnými štítky pod tlačítky. Udávají skutečnou logickou hodnotu na vývodu (H - logická 1, L - logická 0). Pod rámečkem *Výstupy* se nacházejí tlačítka pro práci s vybraným portem. Ty lze vybírat pomocí nabídky níže s nápisem *Porty*. Aktivováním této nabídky se nám ukáže seznam volných portů. K vyhledání nebo aktualizování seznamu portů slouží tlačítko *Hledat porty*. Vybraný port si vyhradíme pro další práci stiskneme tlačítka *Připojit na port*. Opětovným stiskem tlačítka se od portu odpojíme. O všech krocích jsme přitom informováni stavovým řádkem, který lze kdykoli smazat tlačítkem *Smaž*.

4.4.1.3 Zdrojový kód Counter-Comm

Připojení k portu u pomůcky Conter-Signal je programově řešeno obdobně. Pouze procedura tlačítka *Připojit na port* se částečně liší kvůli obsluze rámečku *Přenosová rychlost*. Protože je nutné pro správnou inicializaci portu nejprve vybrat přenosovou rychlost, začíná procedura blokem *If ... End If*. Ten testuje má-li proměnná *sign* hodnotu *false* (nastaví se v proceduře *Timer1_Timer*, viz. dále). Pokud ano, spustí se zpráva pro uživatele, která jej vyzve k výběru přenosové rychlosti. Potom procedura skončí. Následují řádky otevírající vybraný port. Tyto příkazy jsou stejné jako v případě *Conter-Signal*. O funkci této procedury viz. dále.

```
1. Private Sub Command1_Click()
2.     If sign = False Then
3.         msg = MsgBox("Nejprve vyber přenosovou rychlost!", vbExclamation +
4.             vbOKOnly)
5.     End If
6.     'v pripade chyby odskoci na navesti chyba_portu
7.     On Error GoTo Chyba_portu
8.     If MSComm1.PortOpen = False Then
9.         For port = 1 To 20
10.            If Comb1.Text = com(port) Then
11.                MSComm1.CommPort = port
12.                List1.AddItem radek + "vybrán port " + CStr(port) + "..."
13.            End If
14.        Next port
15.        MSComm1.PortOpen = True
16.        Command1.Caption = "Odpojit od portu"
17.        List1.AddItem radek + "připojeno..."
18.    Else
19.        MSComm1.PortOpen = False
20.        Command1.Caption = "Připojit na port"
21.        List1.AddItem radek + "port byl odpojen..."
22.    End If
23. Chyba_portu:
24.     If Comb1.Text = "Porty" Then
25.         msg = MsgBox("Nejprve vyber port!", vbExclamation + vbOKOnly)
26.     End If
27. End Sub
```

Nyní přejdeme k proceduře *Timer1_Timer*. Jejím úkolem je každých 500ms testovat (hodnota vlastnosti *Interval* čítače *Timer1*), která přenosová rychlost (odpovídá volbám *Option1* až *Option6*) je právě vybrána a podle toho nastavit parametry portu. K tomuto účelu je použit vícenásobný blok *If ... End If*. Je-li vybrána některá z možností, nastaví se přitom také proměnná *sign* na hodnotu *True*. V opačném případě se nastaví na hodnotu *False*. Tato informace nám pak poslouží v předchozí proceduře.

```
1. Private Sub Timer1_Timer()  
2.     'volba prenosovych rychlosti  
3.     If Option1.Value = True Then  
4.         MSComm1.Settings = "2400,n,8,1"  
5.         sign = True  
6.     ElseIf Option2.Value = True Then  
7.         MSComm1.Settings = "4800,n,8,1"  
8.         sign = True  
9.     ElseIf Option3.Value = True Then  
10.        MSComm1.Settings = "9600,n,8,1"  
11.        sign = True  
12.     ElseIf Option4.Value = True Then  
13.        MSComm1.Settings = "19200,n,8,1"  
14.        sign = True  
15.     ElseIf Option5.Value = True Then  
16.        MSComm1.Settings = "38400,n,8,1"  
17.        sign = True  
18.     ElseIf Option6.Value = True Then  
19.        MSComm1.Settings = "57600,n,8,1"  
20.        sign = True  
21.     Else  
22.        sign = False  
23.     End If  
24. End Sub
```

Procedura *Timer2_Timer* obsluhuje načtení a zobrazení příchozích dat. Samotná procedura je poměrně krátká a používá vlastnost *InBufferCount* objektu *MSComm1*. Tato vlastnost udává počet znaků čekajících na přečtení v přijímací vyrovnávací paměti. Této skutečnosti proto můžeme využít k testování přijatých znaků. Jestliže je toto číslo nenulové (řádek 2), načteme je do proměnné *vyst_data* (řádek 3) a zobrazíme ve Stavovém řádku (řádek 4). Pokud v paměti žádné nepřečtené znaky nejsou procedura skončí (řádek 6).

```

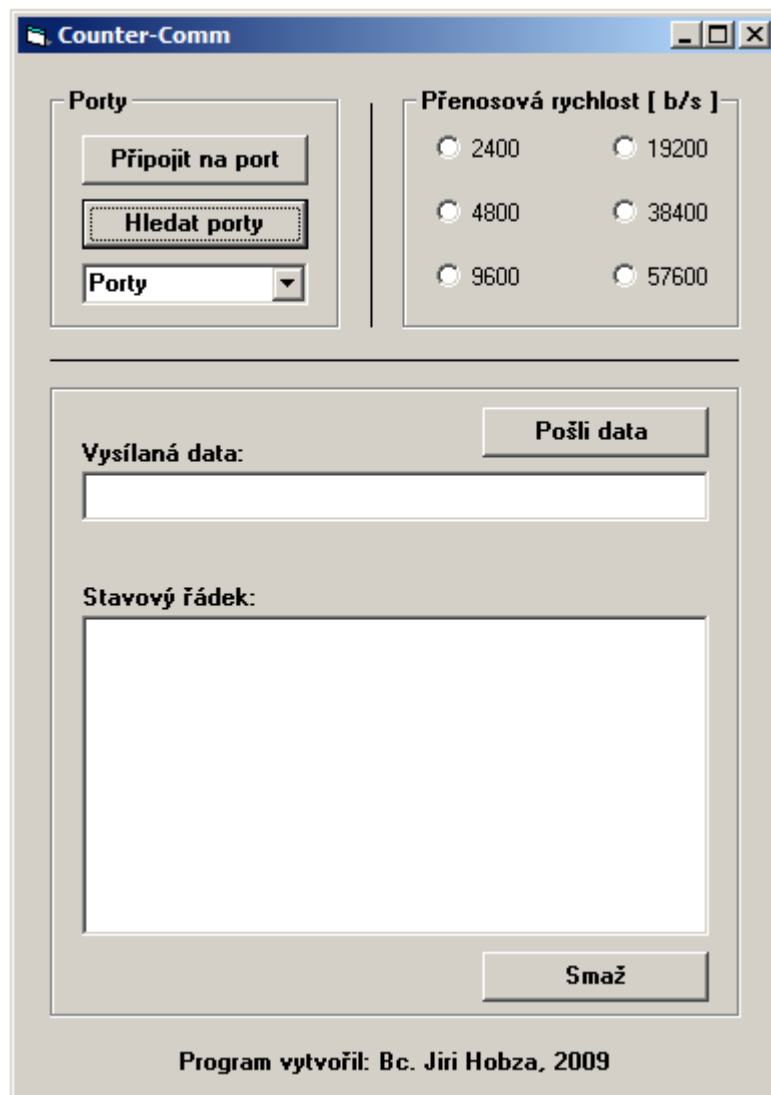
1. Private Sub Timer2_Timer()
2.     If MSComm1.InBufferCount <> 0 Then
3.         vyst_data = MSComm1.Input
4.         List1.AddItem radek + "Přijata data: " + vyst_data
5.     Else
6.         Exit Sub
7.     End If
8. End Sub

```

Oba dva časovače se spouštějí při načtení okna (procedura *Form_Load*) a to příkazy `Timer1.Interval = 500` a `Timer1.Enabled = True` (obdobně pro časovač 2).

4.4.1.4 Použití programu Counter-Comm

Během vývoje čítače impulsů jsme pro potřeby testování vysílání a příjmu dat, vyvinuli program Counter-Comm. Okno programu tvoří rámečky s názvy *Porty*, *Přenosová rychlost* a rámeček s textovými políčky *Vysílaná data* a *Stavový řádek*. Výběr portu funguje stejně jako v případě pomůcky Counter-Signal. V seznamu *Porty* vybereme port, ke kterému se chceme připojit a klikneme na tlačítko *Připojit na port*. Stejně tlačítko slouží také k odpojení se od portu. Je-li k počítači připojeno další zařízení a přitom program USB Test již běží, nebude další port rozpoznán. Pro obnovení seznamu dostupných portů poslouží tlačítko *Hledat porty*. Oproti Counter-Signal je zde k dispozici také volba přenosové rychlosti. Platná je vždy jen jedna volba, ale lze ji změnit kdykoli během komunikace. Pro vystavení řetězce dat na port, klikneme do textového pole *Vysílaná data*. Stisknutím tlačítka *Pošli data:* se data odešlou. Pokud čekáme na odeslaný řetězec nějakou odezvu (například místo malých písmen velká, příjem v opačném pořadí či příjem stejného řetězce), bude okamžitě zobrazena ve stavovém řádku. Tímto způsobem budou zobrazena také ostatní příchozí data.



Obr. 45: Okno programu Counter-Comm

4.4.1.5 Zdrojový kód Counter-Prog

Program Counter-Prog byl vytvořen, jak napovídá název, za účelem programování mikroprocesoru ATtiny2313 pomocí sběrnice SPI. Avšak význam tohoto programu, zvláště té části, která obsluhuje sběrnici SPI, je však daleko univerzálnější. Serial Peripheral Interface neboli SPI je jméno standardu, který zavedla společnost Motorola. Je to synchronní sériová datová linka. Dvě zařízení typu Master/Slave pomocí ní komunikují v plně duplexním režimu. Dokonce je možná varianta s více podřízenými zařízeními, ale musíme použít další linku pro jejich výběr (tzv. chipselect signál). Výhodami tohoto standardu je plně duplexní režim a vyšší propustnost než u sběrnic I²C nebo SMBus. Protokol je založen na přenosu bitů, takže nemusíme používat například jen 8-bitový formát přenosového rámce. Navíc, protože zařízení Slave používá hodinový signál zařízení Master, odpadá použití dalšího oscilátoru. Protokol se také obejde bez adresování. Má však i své stinné

stránky. Při použití více než jednoho Slave zařízení, potřebujeme stejný počet chipselect vodičů. Při komunikaci nedostáváme od podřízeného čipu žádné potvrzení o přijetí dat. Může se tedy stát, že budeme číst data tam, kde žádná nejsou a nebudeme o tom ani vědět. Oproti jiným komunikačním protokolům, jako například RS-232, lze tento používat jen na velmi krátké vzdálenosti.

I přes všechna negativa se stal velice oblíbeným díky své jednoduchosti. Důkazem, že se dnes stále hojně používá je i tato práce. Algoritmus představený v následujícím textu, na jehož základě lze programovat mikrokontroléry ATMEL podporující sériové stahování dat, je možné aplikovat také na jiné čipy. Mohou to být různé typy pamětí (EEPROM, FLASH), MMC a SD paměťové karty, AD a DA převodníky, audio čipy nebo senzory neelektrických veličin.

Jako první popíšeme proceduru tlačítka (symbol tři teček), kterým se otevírá dialogové okno pro otevření souboru. Procedura začíná voláním dialogového okna a přiřazením cesty k vybranému souboru řetězci *soubor* (řádek 3 a 4). Pokud žádný soubor nevybereme, způsobíme chybu, kterou zachytává příkaz na řádku osm. Ten vyvolá odskočení na návěští *Chyba_souboru*. Jestliže je řetězec *soubor* prázdný, informační řádek nás o této skutečnosti informuje (řádky 44 až 46). Nyní přistoupíme k samotnému načtení souboru *.hex. Filozofie je následující. Data ze souboru jsou ukládána do tří polí. Jejich rozměr závisí na proměnné *pocet_radku*. Pro načítání jednotlivých znaků si vyhradíme proměnnou *c* (řádek 9). Otevřeme soubor jako binární a nastavíme počet řádků na nula (řádky 11 a 12). Následuje smyčka Do ... Loop. Příkazem na řádku 13 načteme jeden znak a uložíme jej do proměnné *c*. Jestliže je to konec řádku (podle ASCII tabulky odpovídá číslu 10), počet řádků se zvýší o jeden (řádky 14 až 17). Smyčka pokračuje až do konce souboru (řádek 17). Nyní už můžeme upřesnit rozměr polí (řádky 19 až 21) a zároveň je naplnit. Tím se potřebná data uloží do paměti vždy, když otevřeme požadovaný soubor *.hex. K tomuto účelu slouží cyklus For ... Next (řádky 25 až 39). Tento druh smyčky je vhodný, pokud známe počet průchodů smyčkou. V našem případě inkrementujeme proměnnou *i* od jedné do *pocet_radku*. Na jejím začátku ukládáme jednotlivé znaky do proměnné *řádek* až do načtení ASCII znaku pro konec řádku. Při každém průchodu smyčkou inkrementujeme proměnnou *pozice*, která jak název napovídá, určuje aktuální pozici znaku v řádku. Na řádcích 35 až 37, pak potřebná data vyjmeme z řetězce *řádek* a přiřadíme je polím. Pole *data_s()* obsahující bajty zkompilevaného programu je typu string (řetězec). To proto, že samotné vyjmutí z řetězce a konverze do binárního tvaru, se provádí až v proceduře pro programování paměti.


```

1. Private Sub Command1_Click()
2.     CommonDialog1.ShowOpen
3.     soubor = CommonDialog1.FileName
4.     Text1.Text = soubor
5.     'obslouzi nacteni .hex souboru
6.     'nejprve zachyti chybu pokud stiskneme storno
7.     On Error GoTo chyba_souboru
8.     'zjistí počet radku
9.     c = Space$(1)
10.    Open soubor For Binary As #1
11.    pocet_radku = 0
12.    Do
13.        Get #1, , c
14.        If c = CStr(Chr(10)) Then
15.            pocet_radku = pocet_radku + 1
16.        End If
17.    Loop Until EOF(1)
18.    'upresneni rozmeru poli
19.    ReDim pocet_dat(1 To pocet_radku)
20.    ReDim typ_dat(1 To pocet_radku)
21.    ReDim data_s(1 To pocet_radku)
22.    'nactu data
23.    pozice = 1
24.    'postupuju po radcich
25.    For i = 1 To pocet_radku
26.        'nejprve nacti cely radek jako retezec
27.        radek = ""
28.        Do
29.            Get #1, pozice, c
30.            pozice = pozice + 1
31.            radek = radek + c
32.        'az do konce radku
33.        Loop Until c = CStr(Chr(10))
34.        'vyjme potrebna data z retezce "radek"
35.        pocet_dat(i) = Val("&H" & Mid$(radek, 2, 2))
36.        typ_dat(i) = Val("&H" & Mid$(radek, 8, 2))
37.        data_s(i) = Mid$(radek, 10, 32)
38.        'a dalsi radek
39.    Next i
40.    Close #1
41.    List1.AddItem ">> načten soubor *.hex"
42.    'navesti pro chybove hlaseni

```

```

43. chyba_souboru:
44.     If soubor = "" Then
45.         List1.AddItem ">> soubor nevybrán!"
46.     End If
47. End Sub

```

Nejdůležitější částí zdrojového kódu programu USB Prog je ale procedura, obsluhující čtení a zápis bitů na sběrnici SPI. Během vývoje čítače impulsů se ukázala jako klíčová schopnost, dostatečně přesně měřit čas programovými metodami. K tomuto účelu jsme použili funkce *QueryPerformanceCounter* a *QueryPerformanceFrequency*. První funkce vrací hodnotu čítače s vysokým rozlišením, která představuje počet pulsů od spuštění programu. Druhá funkce vrací frekvenci tohoto čítače závislou na kmitočtu procesoru, ale s pevně danou hodnotou. Obě funkce jsou vázané na Windows (verze 95 a vyšší) a tak lze s nimi pracovat na všech stanicích, kde je nainstalován tento operační systém.

Procedura začíná cyklem For ... Next (řádky 2 až 4), který nuluje proměnnou *inbajt()*. Ta slouží k ukládání rámce čtyř bajtů, přicházejících z MCU. Následuje další cyklus, který se provádí právě tolikrát, z kolika bajtů se skládá rámec instrukcí pro mikrokontrolér (celkem ze čtyř). Do tohoto cyklu je vnořený další, který se tentokrát provede osmkrát a to proto, že jeden bajt sestává z osmi bitů. Tento nejvnitřnější cyklus začíná čtením linky MISO (řádek 9 až 15). Je-li stav na lince logická jedna (pozor! - signály jsou negované), přičte se ke k-tému bajtu nula (hodnota se nezmění). V opačném případě se ke k-tému bajtu přičte 1-tý bit, čímž se hodnota čteného bajtu změní. Následuje vystavení bitu na vývod MOSI. Řádky 23 až 31 pak simulují kladný puls hodinového signálu na SCK. Zároveň můžeme vidět jak se smyčkou odměřuje čas.

```

1. Private Sub cti_posli()
2.     For k = 1 To 4
3.         inbajt(k) = 0
4.     Next k
5.     'rozklad na bity
6.     For k = 1 To 4
7.         For l = 1 To 8
8.             'precte stav MISO
9.             If MSCComm1.CTSHolding And True Then
10.                inbajt(k) = inbajt(k) + 0
11.                'List1.AddItem inbajt(k)
12.            Else
13.                inbajt(k) = inbajt(k) + bity(l)

```

```

14.         'List1.AddItem inbajt(k)
15.     End If
16.     'vystavi na MOSI
17.     If outbajt(k) And bity(1) Then
18.         MSComm1.Break = False
19.     Else
20.         MSComm1.Break = True
21.     End If
22.     'SCK na log. 1
23.     MSComm1.DTREnable = False
24.     'ceka
25.     QueryPerformanceCounter citac1
26.     Do
27.         QueryPerformanceCounter citac2
28.         cas = (citac2 - citac1) / f
29.     Loop Until cas >= 0.0001
30.     'SCK na log. 0
31.     MSComm1.DTREnable = True
32.     'smycka se opakuje
33.     Next l
34. Next k
35. End Sub

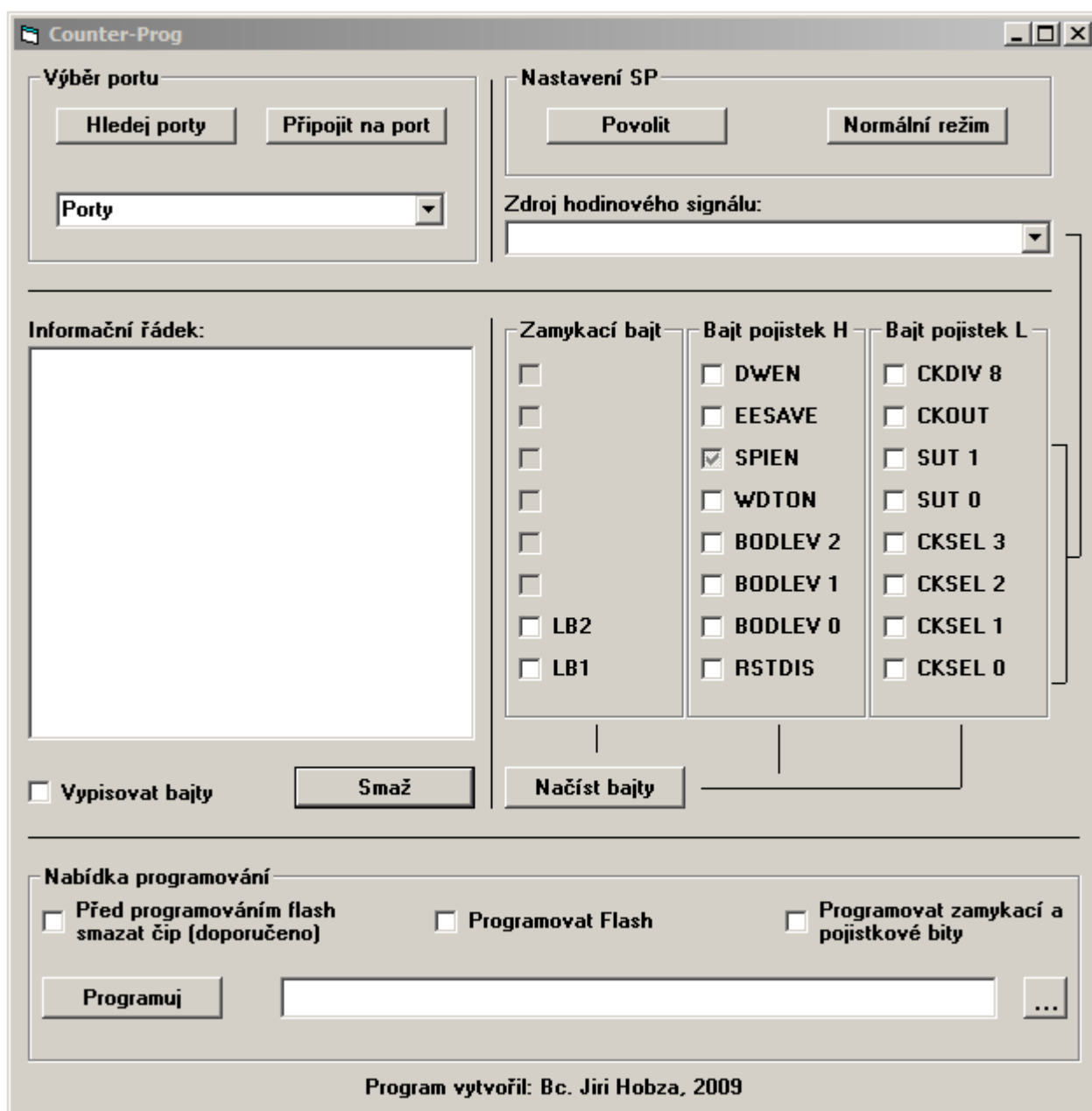
```

4.4.1.6 Použití programu Counter-Prog


Na obrázku 46 je okno programu po spuštění. V levém horním rohu je umístěn rámeček pro výběr portu. Jeho ovládání je stejné jako v předchozích aplikacích. Vedle něj se nachází tlačítka *Povolit* a *Normální režim*. Tato tlačítka slouží pro nastavení sběrnice sériového programování. Pokud jsme již připojeni k nějakému portu, stiskem tlačítka *Povolit*, odešleme povolovací instrukci (viz. sériové stahování dat). Je-li vše v pořádku, *Stavový řádek* nás o úspěchu právě provedené operace informuje. Stejně tak i v případě, pokud by se odeslání povolovací instrukce nezdařilo. Například z důvodu nepřipojení k žádnému portu. Dále poznamenejme, že jakmile je sériové programování připraveno, zůstává aktivní dokud neklikneme na tlačítko *Normální režim*.

V režimu sériového programování je mimo jiné možné také číst a nastavovat doplňkové bity. Rámeček s nimi se nachází vedle stavového řádku. Nejvýznamnější z nich jsou bity CKSEL a SUT, sloužící pro nastavení zdroje hodinového signálu. Abychom si nemuseli pamatovat všechny kombinace, je možné některé z nich (některé jsou rezervované a další nelze použít pro krystal kmitočtu 20MHz) vybrat v nabídce *Zdroj hodinového signálu*. Příslušné bity se pak automaticky

nastaví. Obdobně stiskem tlačítka *Načíst bajty* načteme bity CKSEL a SUT a nastavení odpovídající jejich kombinaci se zobrazí v poli *Zdroj hodinového signálu*. Spolu s těmito bity dojde také k přečtení ostatních.



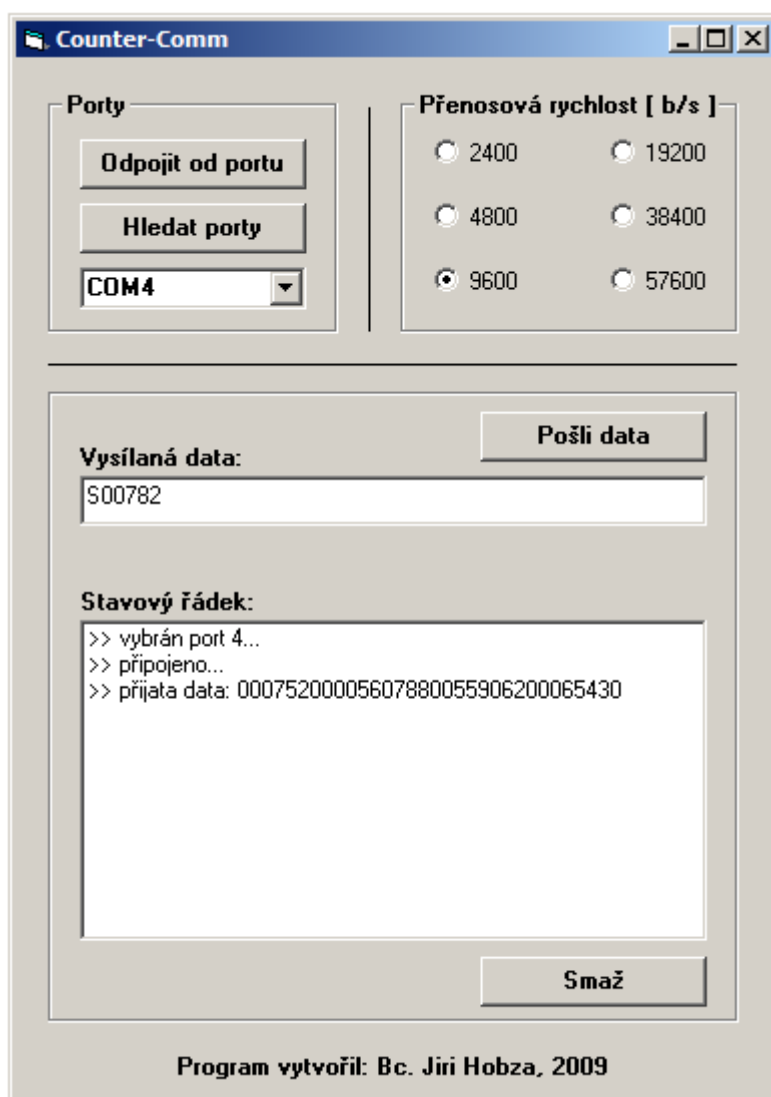
Obr. 46: Okno programu Counter-Prog

Rámeček *Nabídka programování* slouží, jak název napovídá, k programování paměti Flash, ale i ke smazání čipu nebo k programování doplňkových bitů potom, co jsou nastaveny. K programování paměti musíme nejprve stisknout tlačítko se symbolem tří teček . Otevře se klasické dialogové okno Windows, pomocí kterého vybereme soubor *.hex a stiskneme tlačítko OK. Název i cesta k

souboru se zobrazí v poli vedle tlačítka *Programuj*. Poté zaškrtneme políčka *Před programováním smazat čip* a *Programovat Flash*. Není sice nutné před programováním paměť mazat, ale je to doporučeno. Vyhneme se tak situaci, kdy v některých paměťových buňkách zůstane informace z předchozího programování. Poté stiskneme tlačítko *Programuj* a během několika sekund se paměť naprogramuje. Při programování doplňkových bitů není nutné nejdříve mazat Flash. Pouze v případě odblokování zamykacích bitů (viz [6]).

5 Použití čítače impulsů

Čítač impulsů může pracovat ve dvou režimech. V režimu čítání a v režimu programování. Nejčastěji zřejmě bude pracovat jako čítač, což je ten nejjednodušší mód.



Obr. 47: Obsluha čítače impulsů

Připojíme čítač k USB. Pokud jsme tak již neučinili, přepneme páčkový prepínač do polohy *Režim čítání*. Tím je čítač připraven přijímat instrukce z portu USB. Ke komunikaci s ním využijeme doprovodný program Counter-Comm, nebo jiný software tohoto typu. Po připojení na příslušný port můžeme odeslat spouštěcí instrukci. Tu čítač očekává ve tvaru velkého S a pěticeferné doby čítání v milisekundách (viz. obr. 47). Po odeslání instrukce začne čítač provádět měření. Jeho výsledek, tedy počet načítaných impulsů z jednofotonového detektoru, pak odešle zpět na port USB. Data budou přijata ve formě čtyř osmimístných číslic. První osmice odpovídá vstupu číslo čtyři a poslední vstupu číslo jedna. Nezapomeňme, že doba čítání musí korelovat s frekvencí příchozího signálu, jinak může čítač přetéct.

Druhý režim slouží pro programování MCU ATtiny2313. K jeho aktivaci je nutné přepnout páčkový prepínač do polohy *Režim programování*. Pak již stačí jen postupovat podle návodu v předchozí kapitole.

6 Závěr

Cílem diplomové práce bylo navrhnout a zrealizovat zařízení pro čítání impulsů z jednofotonového detektoru. Je jakýmsi vedlejším cílem práce, aby zařízení bylo co nejuniverzálnější. Práce přitom měla být koncipována částečně také jako manuál, pro ty, kdo by si chtěli podobné zařízení postavit. Domníváme se, práce tak byla koncipována správně.

Teoretická část práce seznamuje čtenáře s nejdůležitějšími daty z katalogových listů, aby se rychle zorientoval v problematice. Pro doplňující informace je mu samozřejmě nabídnuta potřebná literatura. Součástí práce je také výroba zařízení. Tím se zabývá praktická část práce.

Nejprve popisuje schéma zapojení a pak návrh odpovídající DPS. Dále se práce zabývá popisem ovládacího programu pro mikrokontrolér. Velkým přínosem praktické části je také vyřešení programování MCU ATtiny2313, pomocí aplikace Counter-Prog. Tím přínosem je skutečnost, že sběrnice SPI je univerzální a je používána u mnoha jiných čipů. Proto na ně může být snadno rozšířena. V práci jsou také vysvětleny nejdůležitější části zdrojových souborů, a proto je možné tyto programy realizovat také v jiných jazycích.

Vzniklo tedy zařízení splňující parametry, které byly stanoveny v úvodu práce. Jmenujme ty nejdůležitější:

- snadné připojení k počítači prostřednictvím USB,
- napájení zařízení přímo z portu USB (+5 V, 490mA max.),
- detekce TTL impulsů od délky 4ns,
- maximální vzorkovací kmitočet 1MHz,
- programování MCU ATtiny2313 přímo na DPS,
- možnost rozšíření programování na další MCU či jiné čipy,
- nízká cena oproti komerčním zařízením (asi 550 až 600Kč).

Doufáme, že samotná práce včetně jejího výsledku bude pro laboratoře Katedry optiky přínosem. Snad také nezůstane jen u měření v laboratoři a najde uplatnění také při výuce předmětů zaměřených na optoelektroniku.

Literatura

Monografie:

- [1] Matoušek, David Ing.: USB prakticky s obvody FTDI, BEN – technická literatura, Praha 2003, ISBN 80-7300-103-9.
- [2] Váňa, Vladimír: Mikrokontroléry ATMEL AVR - popis procesoru a instrukční soubor, BEN - technická literatura, Praha 2003, ISBN 80-7300-083-0.
- [3] Petroustos, Evangelos: Visual Basic 6 - průvodce programátora, GRADA Publishing, 1999, ISBN 80-7169-801-6.

Článek v časopise:

- [4] Matoušek, David Ing.: Programování a aplikace mikrořadiče ATtiny 2313 I., A Radio – Konstrukční elektronika (5, 2006), ISSN 1211 – 3557.
- [5] Matoušek, David Ing.: Programování a aplikace mikrořadiče ATtiny 2313 II., A Radio – Konstrukční elektronika (1, 2007), ISSN 1211 – 3557.

Internet, dokumentace výrobce a katalogové listy:

- [6] URL: <http://www.atmel.com/dyn/resources/prod_documents/doc2543.pdf> .
- [7] URL: <http://www.atmel.com/dyn/resources/prod_documents/doc2521.pdf> .
- [8] URL: <www.atmel.com/dyn/resources/prod_documents/DOC0943.PDF> .
- [9] URL: <http://www.ftdichip.com/Documents/DataSheets/DS_FT232BM.pdf>.
- [10] URL: <<http://www.datasheetcatalog.org/datasheet2/d/0jt5rgh3d5lefzcrypxs4fpgfgpy.pdf>>.
- [11] URL: <<http://www.datasheetcatalog.org/datasheet/philips/74HCHCT14CNV2.pdf>>.
- [12] URL: <<http://www.mcu.cz/startpage.php>>.
- [13] URL: <<http://www.elcad.cz/eagle/>> .
- [14] URL: <<http://www.mcu.cz/startpage.php>> .
- [15] URL: <<http://msdn.microsoft.com/cs-cz/default.aspx>> .