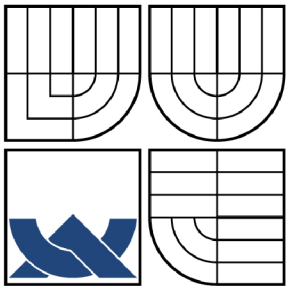
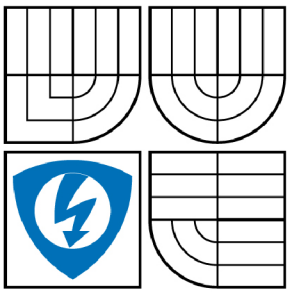


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

UNIVERZÁLNÍ ANALYZÁTOR SÉRIOVÝCH SBĚRNIC

UNIVERSAL ANALYZER OF SERIAL BUSES

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

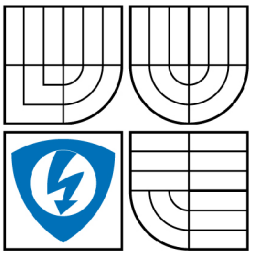
AUTOR PRÁCE — Bc. MATÚŠ GAJDOŠ

AUTHOR

VEDOUCÍ PRÁCE — Ing. PETR SYSEL, Ph.D.

SUPERVISOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student: Bc. Matúš Gajdoš ***ID:*** 83794
Ročník: 2 ***Akademický rok:*** 2008/2009

NÁZEV TÉMATU:

Univerzální analyzátor sériových sběrnic

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se se sériovými sběrnicemi I2C, SPI, UART, SMBUS, atd. Navrhněte blokové schéma analyzátoru pro sledování provozu na těchto sběrnicích, který bude umožňovat také filtrování zpráv podle adresy, překlad zpráv mezi různými sběrnicemi, odesílání zpráv z počítače, apod. Analyzátor bude možné připojit k počítači, kde bude možné sledovaný provoz minimálně uložit do souboru. Vyberte vhodný procesor pro realizaci analyzátoru, navrhněte obvodové zapojení, desku plošných spojů a analyzátor realizujte.

DOPORUČENÁ LITERATURA:

[1] Pinker, J. Mikroprocesory a mikropočítače. První vydání. Praha, BEN: 2004. ISBN 80-7300-110-1
[2] Vlach, J., Vlachová, V. Počítačová rozhraní: přenos dat a řídicí systémy. Druhé vydání. Praha, BEN: 2002. ISBN 80-7300-010-5
[3] Vacek, V. Sériová komunikace ve Win32. První vydání. Praha, BEN: 2003. ISBN 80-7300-086-5

Termín zadání: 9.2.2009 ***Termín odevzdání:*** 26.5.2009

Vedoucí práce: Ing. Petr Sysel, Ph.D.

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Cílem této diplomové práce je navrhnout a zrealizovat univerzální analyzátor sériových sběrnic. V úvodu jsou podrobně popsány specifikace a vlastnosti nejčastěji používaných sériových sběrnic I²C, SMBus, PMBus a rozhraní UART.

Druhá část práce je zaměřena na samotný návrh a realizaci řídicí jednotky, která tvoří základ celého analyzátoru. Ta poskytuje možnost připojení na neznámou sběrnici, na sběrnice I²C, SMBus, PMBus a SPI a na rozhraní UART. Hlavní části řídicí jednotky je mikrokontrolér PIC24HJ128GP204, který vykonává i částečnou analýzu průběhu signálů na připojené sběrnici.

V další části je popsána vytvořená uživatelská aplikace pro počítač PC, která prostřednictvím USB portu řídí a nastavuje řídicí jednotku a získává z ní částečně analyzovaná data. Z nich poté dokončí celkovou analýzu a získané výsledky prezentuje uživateli v grafické anebo textové podobě. Aplikace umožňuje tyto výsledky i exportovat do několika grafických formátu. Podporované jsou formáty PS, SVG, PDF, PNG a BMP. V případě, že řídicí jednotka není k dispozici, je možné analýzu některých sběrnic v uživatelské aplikaci alespoň odsimulovat a vyzkoušet tak některé z funkcí, které aplikace nabízí. Pro tento případ je k dispozici simulace analýzy neznámé sběrnice a sběrnice I²C. Uživatelská aplikace obsahuje také i funkci generování vlastního průběhu signálů, který je možné vytvářet interaktivně myší v grafické reprezentaci signálů. Takto vytvořený průběh je možné odeslat řídicí jednotce, která ho aplikuje na výstupních linkách. Tato funkce není v současné verzi programu řídicí jednotky podporována.

Uživatelská aplikace obsahuje i některé prvky, pro které nejsou vytvořené metody a jsou připravené na další rozšiřování funkcí analyzátoru. Patří sem zejména formuláře a ovládací prvky pro konverzi dat mezi sběrnicemi.

KLÍČOVÁ SLOVA

Sběrnice, I2C, SMBus, PMBus, SPI, UART, komunikace, analýza, C#, .NET Framework

ABSTRACT

The purpose of this master's thesis is to design and to realize the universal analyzer of serial buses. At the beginning there are described in detail specifications and properties of buses I²C, SMBus, PMBus and of the UART interface.

The second part is targeted especially to design and to realize the control unit, which is the main part of the whole analyzer. It provides connecting to the unknown bus, to buses I²C, SMBus, PMBus and SPI and to the UART interface. The main part of the control unit is the microcontroller PIC24HJ128GP204, which partially analyses connected bus.

In the next part there is created the user application for PC, which controls and sets the control unit and receives from it partially analyzed data through the USB port. These data are used for the rest of the analysis that is done by the user application. Results may be presented to user in graphical or text form. The user application provides export of results into some of image formats. Supported formats are PS, SVG, PDF, PNG and BMP. In case of the control unit is not available, the user may choose the simulator to simulate some functions that the application provides. It supports simulation of the analysis of the unknown bus and the I²C bus. The user application also contains the function of the generating own signals. Signals can be created interactively by using the mouse in the graphical representation of this signal. They can be sent to the control unit and after that it applies them to output. This function is not supported in current version of the control unit software.

The user application includes some items, which do not have implemented any methods and they are prepared for the future upgrading of the analyzer. It means especially user controls and components for signal conversions between any buses.

KEYWORDS

Bus, I2C, SMBus, PMBus, SPI, UART, communication, analysis, C#, .NET Framework

BIBLIOGRAFICKÁ CITACE MÉ PRÁCE

GAJDOŠ, M. *Univerzální analyzátor sériových sběrnic.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 85 s. Vedoucí diplomové práce Ing. Petr Sysel, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svoji diplomovou práci na téma „Univerzální analyzátor sériových sběrnic“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této práce jsem neporušil autorský práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne …… . ……

(podpis autora)

POĎAKOVANIE

Ďakujem vedúcemu diplomovej práce Ing. Petrovi Syslovi, Ph.D. za veľmi užitočnú metodickú pomoc a cenné rady pri spracovaní diplomovej práce.

V Brne dňa …… . ……

(podpis autora)

OBSAH

Úvod **13**

1 Zbernica I²C **14**

1.1 Úvod ... 14

1.2 Vlastnosti ... 14

1.3 Špecifikácia ... 15

1.4 Časovanie signálov a prenos dát ... 15

1.5 Taktovací signál a synchronizácia ... 17

1.6 Riadenie zbernice a arbitráž ... 17

1.7 Adresovanie ... 18

1.8 ŠTART procedúra ... 20

2 Zbernica SMBus **22**

2.1 Úvod ... 22

2.2 Špecifikácia ... 22

2.3 Časovanie signálov a prenos dát ... 23

2.4 Adresovanie ... 25

2.5 Komunikácia ... 25

2.6 Kontrola chýb paketov ... 26

2.7 Protokoly SMBus ... 26

2.7.1 Quick Command ... 27

2.7.2 Send Byte ... 27

2.7.3 Receive Byte ... 28

2.7.4 Write Byte/Word ... 28

2.7.5 Read Byte/Word ... 28

2.7.6 Process Call ... 29

2.7.7 Block Write ... 29

2.7.8 Block Read ... 29

2.7.9 Block Write-Block Read Process Call ... 30

2.7.10 Host Notify ... 30

2.8 Porovnanie zbernice SMBus s I²C ... 31

3 Zbernica PMBus **32**

3.1 Úvod ... 32

3.2 Vlastnosti ... 32

4 Zbernica SPI **33**

4.1 Úvod ... 33

4.2 Vlastnosti ... 33

4.3 Špecifikácia ... 34

4.4 Časovanie signálov a prenos dát ... 34

4.5 Taktovací signál ... 35

4.6 Zapojenie Daisy-chained ... 36

5 Rozhranie UART **37**

5.1 Úvod ... 37

5.2 Prenos sériových dát ... 37

5.3 Štandardy RS-232 a RS-485 ... 38

6 Analýza sériových zberníc **40**

6.1 Úvod ... 40

6.2 Riadiaca jednotka ... 40

6.2.1 Konštrukčné riešenie ... 41

6.2.2 Programové riešenie ... 46

6.3 Užívateľská aplikácia ... 54

6.4 Problémy a možné riešenia ... 63

7 Záver **65**

Literatúra **66**

ZOZNAM OBRÁZKOV

Obr. 1.1: Aplikácia I²C v základňovej stanici bezdrôtového telefónu ... 14
Obr. 1.2: ŠTART, prenos bitov a STOP na zbernici I²C. ... 16
Obr. 1.3: Potvrdzovanie prenosu dát na zbernici I²C. ... 16
Obr. 1.4: Synchronizácia taktovacieho signálu medzi *mastrami* na zbernici I²C ... 17
Obr. 1.5: Prevzatie riadenia zbernice I²C jedným z dvoch *mastrov*. ... 18
Obr. 1.6: Kompletný dátový prenos na zbernici I²C. ... 19
Obr. 1.7: Odosielanie dát pri 10-bitovom adresovaní na zbernici I²C ... 19
Obr. 1.8: Príjem dát pri 10-bitovom adresovaní na zbernici I²C. ... 20
Obr. 1.9: Špeciálna ŠTART procedúra na zbernici I²C. ... 20
Obr. 2.1: Časovanie signálov na zbernici SMBus. ... 24
Obr. 2.2: Protokol Quick Command. ... 27
Obr. 2.3: Protokol Send Byte bez PEC (hore) a s PEC (dole). ... 28
Obr. 2.4: Protokol Receive Byte bez PEC (hore) a s PEC (dole). ... 28
Obr. 2.5: Protokol Write Byte (hore) a Write Word (dole). ... 28
Obr. 2.6: Protokol Read Byte. ... 29
Obr. 2.7: Protokol Process Call. ... 29
Obr. 2.8: Protokol Block Write. ... 29
Obr. 2.9: Protokol Block Read. ... 30
Obr. 2.10: Protokol Block Write-Block Read Process Call s PEC ... 30
Obr. 2.11: Protokol Host Notify. ... 31
Obr. 4.1: Zapojenie SPI: vľavo typické, vpravo zreťazené (Daisy-chained). ... 34
Obr. 4.2: Prenos dát po SPI prostredníctvom dvoch posuvných registrov. ... 35
Obr. 4.3: Časovanie signálov na zbernici SPI. ... 36
Obr. 5.1: Prevod paralelných dát na sériové pomocou posuvného registra ... 37
Obr. 5.2: Časový priebeh prenosu znaku „V“ (0101 0110b) na UART. ... 38
Obr. 5.3: Časový priebeh prenosu znaku „V“ na RS-232 (hore) a RS-485 (dole). ... 39
Obr. 6.1: Komunikácia medzi mikrokontrolérom a USB prevodníkom. ... 43
Obr. 6.2: Rozmiestnenie konektorov na riadiacej jednotke ... 44
Obr. 6.3: Riadiaca jednotka analyzátora. ... 46
Obr. 6.4: Stručný vývojový diagram programu riadiacej jednotky analyzátora. ... 48
Obr. 6.5: Priebeh analýzy neznámej zbernice a vytvorený dátový rámec. ... 49

Obr. 6.6: Príklad dátového rámca pri analýze neznámej zbernice. 50
Obr. 6.7: Príklad dátového rámca pri analýze zbernice I²C 52
Obr. 6.8: Príklad dátového rámca pri analýze zbernice SPI. 53
Obr. 6.9: Príklad dátového rámca pri analýze rozhrania UART. 54
Obr. 6.10: Hlavné okno užívateľskej aplikácie. 55
Obr. 6.11: Okno simulátora. 56
Obr. 6.12: Okno s výslednými priebehmi po analýze neznámej zbernice. 57
Obr. 6.13: Graf vyexportovaný do formátu PDF zobrazený vo Foxit Reader. 58
Obr. 6.14: Okno so zachytenými stavmi pri analýze zbernice I²C. 59
Obr. 6.15: Okno so zachytenými stavmi pri analýze zbernice SPI. 60
Obr. 6.16: Okno so zachytenými stavmi pri analýze rozhrania UART. 61
Obr. 6.17: Okno pre generovanie vlastného signálu. 62
Obr. 6.18: Okno pre automatické generovanie periodického signálu. 62
Obr. 6.19: Okno diagnostiky modulu. 63
Obr. 6.20: Príklad chybového hlásenia užívateľskej aplikácie. 64

ZOZNAM TABULIEK

Tab. 2.1: Časovanie signálov na zbernici SMBus. 25
Tab. 4.1: Význam parametrov CPOL a CPHA. 35
Tab. 6.1: Možnosti prepojenia konektorov JP3 a JP4. 45

ÚVOD

V telekomunikáciách a v oblasti počítačov predstavuje sériová komunikácia proces prenosu bitu po bite za sebou prostredníctvom telekomunikačného kanálu alebo zbernice. Priamym opakom je paralelná komunikácia, kde je prenášaných až niekoľko bitov súčasne.

Sériová komunikácia sa používa v diaľkových spojoch a vo väčšine počítačových sietí, kde by cena vedení a problémy spojené so synchronizáciou pri viacnásobných spojoch robili paralelnú komunikáciu nepraktickou a v niektorých prípadoch v podstate nerealizovateľnou. V súčasnosti sa sériová komunikácia vo veľkej miere presadzuje aj oblasti krátkych spojov a vysokorýchlostných zberníc [1]. Dôvodom nasadzovania sériových spojov namiesto paralelných boli a sú okrem jednoduchšieho prepojovania aj problémy spojené s neustálym zvyšovaním rýchlostí prenosov, s čím je spätá najmä parazitná kapacita a presluchy susedných spojov.

V oblasti integrovaných obvodov všeobecne platí, že čím viac má obvod pinov, tým vyššia je jeho cena. Aj z tohto dôvodu v oblasti komunikácie integrovaných obvodov a najmä tam, kde rýchlosť nie je podstatnou, došlo k redukcii pinov a na komunikáciu boli použité rozhrania sériových zberníc.

Prvá časť tejto práce je zameraná na analýzu niektorých vybraných sériových zberníc. Postupne v jednotlivých kapitolách sú veľmi podrobne opísané vlastnosti, špecifikácie a časovanie signálov na sériových zberniciach I^2C, SMBus, PMBus, SPI a rozhraní UART, čo poskytuje čitateľovi veľmi blízke zoznámenie s týmito zbernicami a umožňuje jednoduché pochopenie ich fungovania.

Druhá časť práce je venovaná návrhu a realizácii univerzálneho systému pre analýzu týchto sériových zberníc a rozhraní. Celková analýza je rozdelená a je vykonávaná v dvoch krokoch. Prvým je čiastočná analýza riadiacou jednotkou a druhým je konečná analýza a zobrazenie výsledkov prostredníctvom užívateľskej aplikácie spustenej na počítači PC.

1 ZBERNICA I²C

1.1 Úvod

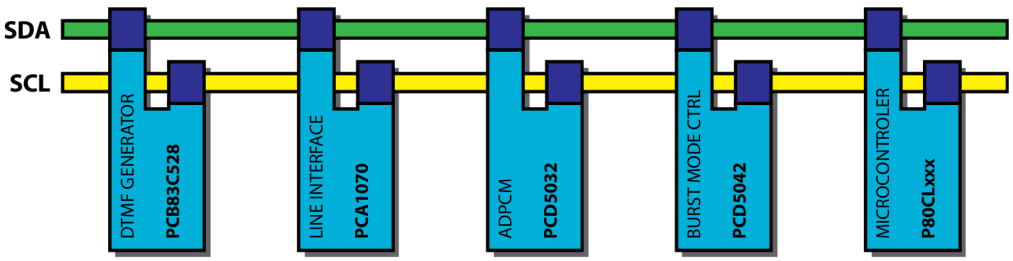
V spotrebiteľskej elektronike, telekomunikáciách a priemyselnej elektronike je často krát veľa spoločného aj napriek tomu, že majú často úplne odlišné využitie. V dnešnej dobe sa už takmer v každom elektronickom zariadení vyskytujú inteligentné riadiace systémy realizované ako jednočipové mikrokontroléry, obvody pre riadenie LCD radičov, vstupno-výstupné porty, dátové prevodníky a aplikačne orientované obvody ako napr. tunery, signálové procesory apod. Pre zjednotenie týchto systémov pre návrhárov a výrobcov a pre zvýšenie hardvérovej efektivity vyvinula spoločnosť Philips zbernicu IIC (Inter-IC Control) označovanú skôr ako I²C™ [2].

1.2 Vlastnosti

Pri komunikácii medzi zariadeniami existuje vzťah *master* (nadriadený) – *slave* (podriadený), pričom *slave* zariadení môže byť pripojených na jednu zbernicu niekoľko a sú riadené vždy len jedným *mastrom*. Výber zariadenia pre komunikáciu sa uskutočňuje na základe jedinečnej adresy. Zariadenia sú k zbernici I²C pripojené prostredníctvom dvoch sériových liniek. Sú to:

- Sériová dátová linka SDA
- a sériová taktovacia linka SCL.

Príklad aplikácie I²C v bežnom užívateľskom zariadení je možné vidieť na obr. 1.1. Zbernica je tzv. true multi-master zbernicou, čo umožňuje pripojenie viacerých *mastrov* na jednu zbernicu súčasne. Existujú tu techniky pre detekciu kolízii a arbitráž (rozhodovanie pri spore) na prevenciu proti poškodeným dátam pre prípad, že prenos zaháji niekoľko *mastrov* súčasne. Sériový 8-bitovo orientovaný obojsmerný prenos po zbernici môže byť uskutočnený rýchlosťou až do 100 kbit/s v štandardnom móde, až do 400 kbit/s v rýchlom móde a až do



Obr. 1.1: Aplikácia I²C v základňovej stanici bezdrôtového telefónu.

3,4 Mbit/s vo veľmi rýchlom móde. Celkový počet zariadení pripojených na jednu zbernicu je limitovaný len celkovou kapacitou zbernice, ktorá nesmie presiahnuť 400 pF.

Integrované obvody s implementáciou I²C umožňujú vytvorenie koncovej aplikácie priamo z funkčného vývojového diagramu. Funkčné bloky vo vývojom diagrame priamo odpovedajú zapojeniu integrovaných obvodov. Jednotlivé obvody je možné pripájať k zbernici a odpájať od zbernice bez pridávania a odoberania ďalších externých súčiastok a zariadení.

1.3 Špecifikácia

Prenos dát medzi zariadeniami sa uskutočňuje prostredníctvom dvoch vodičov SDA a SCL. Sú to obojsmerné linky pripojené k úrovni H (napájacie napätie) pomocou pull-up rezistorov a v systémoch s veľmi rýchlym prenosom dát cez prúdový zdroj [3]. Po linke SCL je prenášaný taktovací signál kvôli synchronizácii a po linke SDA prebieha samotný prenos dát jedným alebo druhým smerom. Ak je zbernica voľná (neprebieha žiadny prenos), obe linky majú úroveň H. Všetky pripojené zariadenia musia mať komunikačné linky typu open-drain alebo open-collector, aby celé zapojenie spĺňalo funkciu wired-AND. Každé zariadenie má v danom systéme jedinečnú adresu a môže pracovať ako vysielač dát alebo dáta môže prijímať. Z hľadiska riadenia zbernice môže zariadenie pracovať ako *master* (nadriadený) alebo *slave* (podriadený) v závislosti od jeho funkcie. *Master* iniciuje a ukončuje dátový prenos na zbernici a zároveň generuje taktovací signál pre všetky *slave*.

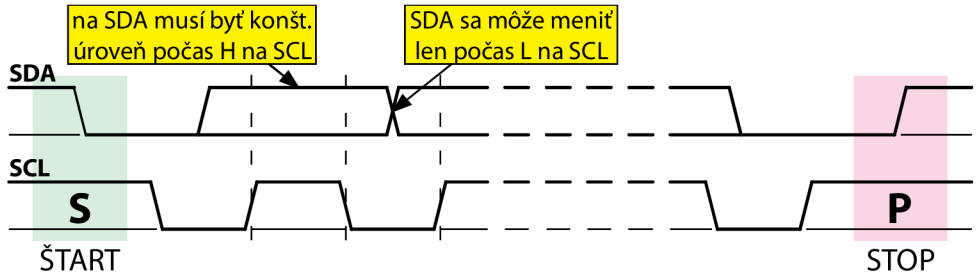
Ako bolo už spomínané, I²C zbernica podporuje pripojenie viacerých *mastrov* súčasne, ktoré tak môžu obsluhovať rovnaké *slave* striedavo. Aby sa však predišlo chaosu, ak by prenos zahájilo viac *mastrov* súčasne, majú I²C zariadenia implementované metódy na riešenie kolízii a arbitráž.

Každý *master* generuje na zbernici vlastný taktovací signál, ktorým synchronizuje všetky *slave*. Tento signál môže byť zámerné podržaný na úrovni L niektorým *slaveom* (stretching). Pomalé zariadenia tak dokážu znižovať prenosovú rýchlosť na zbernici, čím sa predíde zahlteniu alebo strate dát. Podržať taktovací signál na úrovni L môžu aj *mastre* pri riešení kolízii.

1.4 Časovanie signálov a prenos dát

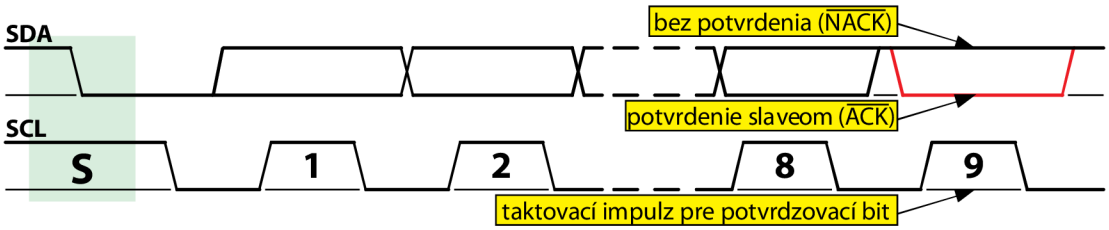
Pri dátovom prenose sa logická úroveň signálu na linke SDA môže meniť len v čase, keď je na linke SCL úroveň L a počas úrovne H musí zostať konštantná ako ukazuje obr. 1.2.

Zmena úrovne na linke SDA v čase, keď je na linke SCL úroveň H predstavuje špeciálne udalosti. Prechod z úrovne H na L predstavuje ŠTART (S) a prechod z úrovne L na H naopak STOP (P).



Obr. 1.2: ŠTART, prenos bitov a STOP na zbernici I²C.

Každý prenášaný bajt musí byť dlhý 8 bitov. Ako prvý je vždy posielaný najvyšší bit a posledný najnižší bit je ukončený potvrdzovaním – bitom $\overline{\text{ACK}}$ (acknowledge). Počet za sebou prenášaných bajtov nie je limitovaný a závisí len na konkrétnej aplikácii. V prípade, že *slave* nie je schopný prijať alebo odoslať nasledujúci bajt, či už z dôvodu obsluhy iného zariadenia alebo vnútorného prerušenia, môže podržať na linke SCL úroveň L až do doby, kým je opäť schopné spracovávať dáta prijaté alebo odoslané cez zbernicu I²C.



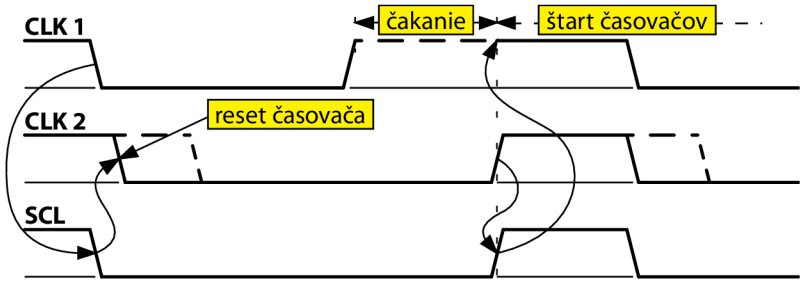
Obr. 1.3: Potvrdzovanie prenosu dát na zbernici I²C.

Pri prenose dát je potvrdzovanie $\overline{\text{ACK}}$ bitom povinné. Na obr. 1.3 je znázornené, ako po odoslaní posledného bitu posielaného bajtu vysielacia strana uvoľní linku SDA a čaká na jej nastavenie na úroveň L prijímacou stranou, ktorá tým potvrdí príjem bajtu. V prípade, že sa *slave* nemôže zúčastniť komunikácie (napr. z dôvodu obsluhy real-time úlohy), môže ihneď po prijatí svojej adresy zamietnuť prenos zaslaním $\overline{\text{NACK}}$. *Master* odošle STOP, čím ukončí aktuálny prenos dát a neskôr sa môže pokúsiť o nový prenos, alebo odošle opakovaný ŠTART (Sr) a zahájí tým nový prenos dát ihneď. Od prvého bajtu počas celej komunikácie sa *slave* kedykoľvek môže rozhodnúť ukončiť prenos dát zaslaním $\overline{\text{NACK}}$. V špecifických prípadoch sa vyžaduje odlišný formát prenosu dát po zbernici I²C (napr. pri protokole C-BUS), kedy sa potvrdenie $\overline{\text{ACK}}$ zámerne neposiela.

Ak *master* požaduje ukončiť prenos počas prijímania dát, za prijatým posledným požadovaným bajtom nepošle vysielacej strane potvrdenie $\overline{\text{ACK}}$, tá sa odpojí od zbernice a *master* môže ukončiť prenos dát zaslaním STOP, alebo zahájí nový prenos odoslaním opakovaného ŠTART.

1.5 Taktovací signál a synchronizácia

Všetky *mastre* generujú na linke SCL svoj vlastný taktovací signál, ktorý nie je nijako frekvenčne obmedzený a závisí len od rýchlosti pripojených zariadení. Dáta sú na linke SDA platné len pri úrovni H na linke SCL, a preto v prípade, že je na zbernicu pripojených viac *mastrov*, je nutné taktovací signál medzi nimi vzájomne synchronizovať.



Obr. 1.4: Synchronizácia taktovacieho signálu medzi *mastrami* na zbernici I²C.

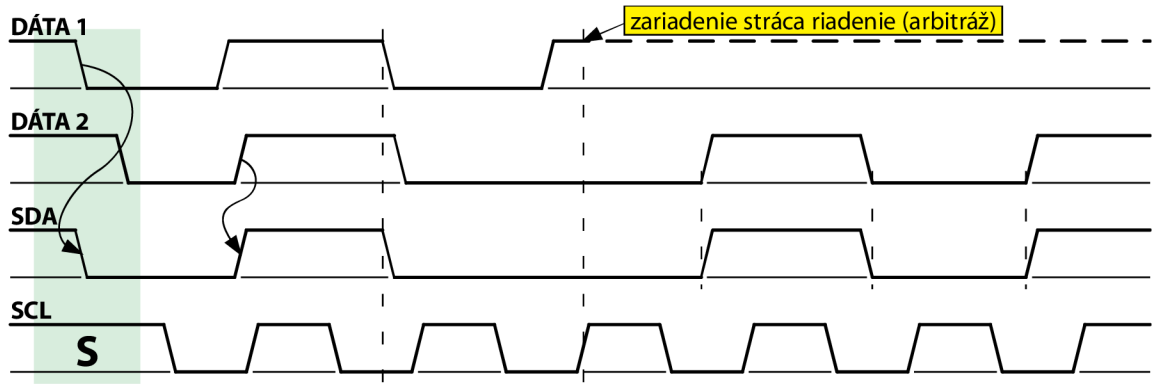
Ako ukazuje obr. 1.4, synchronizácia využíva zapojenie wired-AND na linke SCL. Ak niektorý *master* nastaví úroveň L na linke SCL, ostatné *mastre* resetujú svoje časovače pre úroveň H a dokončia začatú periódu taktovacieho signálu nastavením úrovne L na linke SCL. Najpomalšie zariadenie tak môže podržať úroveň L aj po dlhšiu dobu a ostatné zariadenia musia čakať. Zariadenia odštartujú novú periódu taktovacieho signálu až po uvoľnení SCL linky posledným zariadením. V ďalšej perióde nastaví úroveň L na SCL linke opäť najrýchlejšie zariadenie a celý proces sa opakuje.

1.6 Riadenie zbernice a arbitráž

Master môže zahájiť prenos dát len v prípade, že je zbernica voľná. Môže však nastať situácia, kedy dva *mastre* iniciujú prenos súčasne alebo len s veľmi malým oneskorením.

Na obr. 1.5 je uvedený príklad, v ktorom na začiatku zahájí prenos jeden *master* a v zapätí ďalší urobí to isté. Na zbernici sa tak objaví korektný ŠTART a v tomto momente ani jedno z týchto zariadení nevie, že na zbernici vysiela dáta ešte niekto ďalší. V nasledujúcej perióde taktovacieho signálu posielajú obidve zariadenia rovnaký bit a situácia na zbernici sa nemení. V ďalšej perióde sa však zariadenia pokúsia odoslať na zbernicu

odlišný bit. Keďže linka SDA spĺňa funkciu wired-AND, zariadenie posielajúce úroveň H prepne svoj výstup na dátovej linke do stavu vysokej impedancie a očakáva zmenu úrovne na zbernici zabezpečenú pull-up rezistorom, prípadne prúdovým zdrojom. Druhé zariadenie však v tomto momente posiela odlišný bit, takže nastaví na linke SDA úroveň L. Keďže zariadenia pri odosielaní zároveň monitorujú stav na zbernici, prvé zariadenie zistí, že na zbernici nedošlo k očakávanej zmene úrovne, stráca riadenie a na zbernici zostáva pasívne až do príchodu STOP a až potom sa môže pokúsiť odoslať dáta znovu.

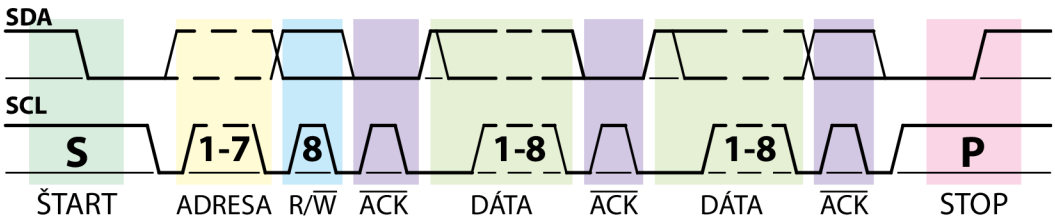


Obr. 1.5: Prevzatie riadenia zbernice I²C jedným z dvoch *mastrov*.

Táto metóda riešenia kolízii nie je pre prenášané dáta deštruktívna, a preto druhé zariadenie bez vedomia o vzniknutej situácii môže ďalej pokračovať v prenose a to bez akéhokoľvek poškodenia prenášaných dát.

1.7 Adresovanie

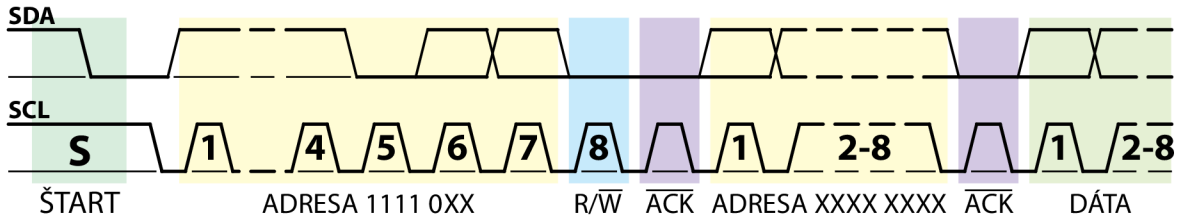
Na zbernici I²C je ŠTART vždy nasledovaný adresou, ktorá určí *slave* pre nasledujúcu komunikáciu. Výnimkou je tzv. general call, ktoré predstavuje adresovanie všetkých *slaveov* pripojených na zbernicu. Táto adresa však nie je podporovaná všetkými druhmi zariadení. Na zbernici I²C je možné používať 7-bitové a 10-bitové adresovanie a niekoľko špeciálnych adries. Pri použití 7-bitovej adresy je ôsmy bit, značený ako R/$\overline{W}$, použitý na určenie smeru toku dát. „Nula“ predstavuje zápis na zbernicu a „jedna“ čítanie. Ak sa adresované zariadenie nachádza na zbernici a je schopné zúčastniť sa požadovanej komunikácie, potvrdí to bitom $\overline{ACK}$. Nasleduje samotný prenos dát, ktorý zakončuje STOP. Príklad takéhoto dátového prenosu je znázornený na obr. 1.6. Ak *master* po odoslaní posledného bajtu ešte nechce ukončiť komunikáciu na zbernici, môže zaslať namiesto STOP opakovaný ŠTART (Sr), adresovať ďalšie zariadenie a pokračovať tak v ďalšom prenose dát. Rovnako ako platí pri prenose dát, aj pri posielaní adresy sa ako prvý posiela vždy najvyšší bit.



Obr. 1.6: Kompletný dátový prenos na zbernici I²C.

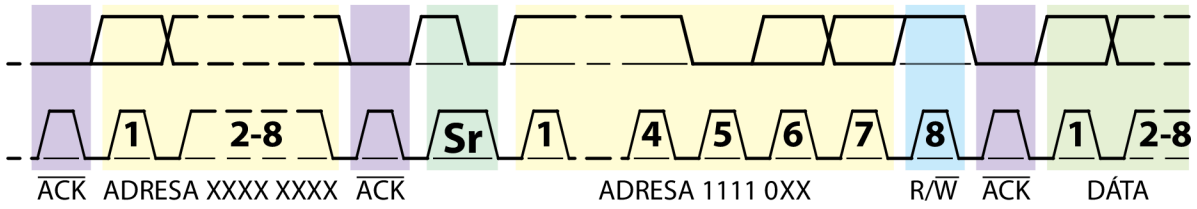
Adresa zariadenia sa môže skladať z fixnej časti, ktorá býva určená priamo výrobcom a z premenlivej časti, ktorú je možné zvoliť ľubovoľne (napr. vhodným prepojením pinov na zariadení). Je tak možné na jednu zbernicu pripojiť súčasne niekoľko rovnakých zariadení.

O prideľovanie fixných častí I²C adries pre zariadenia sa stará špeciálna komisia. Dve skupiny ôsmych adries (`0000 XXXb` a `1111 XXXb`) sú rezervované pre špeciálne účely. Patrí sem už spomínané volanie general call s adresou `0000 000b`, adresa pre protokol C-BUS `0000 001b`, adresy `1111 0XXb`, ktoré sú súčasťou adries pri 10-bitovom adresovaní a ďalšie.



Obr. 1.7: Odosielanie dát pri 10-bitovom adresovaní na zbernici I²C.

Použitie 10-bitovej adresy pri odosielaní dát *mastrom* je znázornené na obr. 1.7. Na začiatku je odoslaný ŠTART nasledovaný špeciálnou adresou `1111 0XXb`. Prvých 5 bitov predstavuje prefix určujúci 10-bitové adresovanie a posledné 2 bity reprezentujú dva najvyššie bity samotnej adresy. Nasledujúci bit – bit R/$\overline{W}$ je nastavený *mastrom* vždy na úroveň L. Predstavuje zápis spodných osem bitov adresy, ktorý bude nasledovať. *Slave* s 10-bitovou adresou, ktoré sú pripojené na zbernicu, skontrolujú prijatú adresu a v prípade, že adresa odpovedá nejakému zariadeniu alebo niekoľkým zariadeniam súčasne, zariadenia odošlú potvrdenie $\overline{ACK}$. Potom *master* odošle zvyšných osem bitov adresy a opäť bude očakávať potvrdenie, ktoré by malo byť zaslané už len jedným *slaveom*. Ak sa tak stane, *master* zahájí prenos samotných dát.

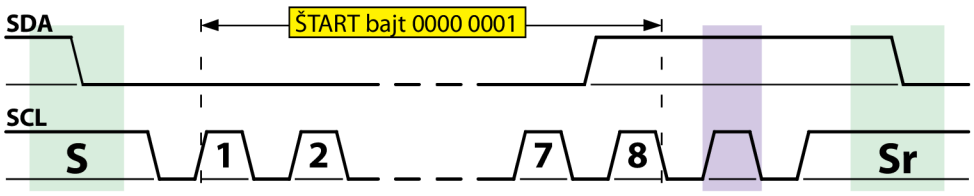


Obr. 1.8: Príjem dát pri 10-bitovom adresovaní na zbernici I²C.

Na obr. 1.8 je zobrazený príjem dát pri použití 10-bitového adresovania. Prenos je *mastrom* iniciovaný rovnako ako v predchádzajúcom prípade. Na začiatku je odoslaný ŠTART, adresa `1111 0XXb`, bit R/$\overline{W}$ signalizujúci zápis a ďalej sa očakáva potvrdenie od *slave*. Potom nasleduje druhá časť adresy `XXXX XXXXb` a opäť čakanie na príjem potvrdenia. V tomto momente je adresovaný slave pripravený na príjem dát od *mastra*. Aby sa zmenil smer toku dát, je odoslaný opakovaný ŠTART (Sr), adresa `1111 0XXb`, kde `XX` sú dva najvyššie bity 10-bitovej adresy a bit R/$\overline{W}$, ktorý v tomto momente mení smer toku dát na čítanie. Ak *slave* potvrdí požiadavku bitom $\overline{ACK}$, *master* môže začať zo zbernice čítať.

1.8 ŠTART procedúra

Mikrokontroléry môžu byť k zbernici I²C pripojené dvoma spôsobmi. Prvým je prípad, keď mikrokontrolér obsahuje hardvérovú implementáciu I²C rozhrania a môže byť naprogramovaný tak, že obslúži len prerušenie zapríčinené zmenou stavu na zbernici a ďalej sa zbernicou už nezaoberá. Druhým prípadom je naopak mikrokontrolér bez hardvérovej podpory I²C, ktorý musí vykonávať obsluhu zbernice softvérovo a musí v pravidelných krátkych intervaloch kontrolovať stav na zbernici, čo zbytočne spotrebúva jeho výkon. Aby sa zaťaženie mikrokontroléru monitorovaním zbernice zredukovalo, je možné využiť špeciálnu ŠTART procedúru. Ako je vidieť na obr. 1.9, táto procedúra pozostáva z bežného ŠTART (S), ŠTART bajtu (`0000 0001b`), potvrdzovacieho bitu (9. bit) a opakovaného ŠTART (Sr).



Obr. 1.9: Špeciálna ŠTART procedúra na zbernici I²C.

Po odoslaní ŠTART *mastrom*, ktorý chce zahájiť komunikáciu na zbernici, je odoslaný ŠTART bajt (`0000 0001b`). Aby sa zachoval rovnaký formát prenášaných rámcov, je ešte odoslaný taktovací impulz pre potvrdenie $\overline{ACK}$, ktorý však musí byť všetkými

slaveami ignorovaný. Ďalší mikrokontrolér tak pri použití tejto metódy môže väčšinu času vzorkovať zbernicu s menšou pravidelnosťou a využiť svoj výkon efektívnejšie. V prípade, že zistí prítomnosť nízkej úrovne na linke SDA, zvýši vzorkovaciu frekvenciu a čaká na ŠTART, ktorým sa zahájí samotná komunikácia.

2 ZBERNICA SMBUS

2.1 Úvod

Zbernica SMBus (System Management Bus) bola pôvodne navrhnutá ako komunikačná linka medzi inteligentnou batériou, jej nabíjačkou a mikrokontrolérom, ktorý ešte komunikoval s ostatnou časťou systému [4]. Jej prvá verzia bola definovaná spoločnosťou Intel v roku 1995. Dnes je podporovaná širokou škálou zariadení a je využívaná v rôznych systémoch.

SMBus, označovaná tiež SMB, predstavuje 2-vodičové rozhranie, ktoré umožňuje vzájomnú komunikáciu rôznych zariadení. Principiálne je založená na funkcii zbernice I²C.

Je určená primárne pre obvody pre správu systému a napájania. Systém ju tak môže s výhodou využívať pre výmenu správ so všetkými zariadeniami a nie je nutné vytvárať a používať individuálne komunikačné linky, čím sa redukuje počet potrebných pinov a vzájomných prepojení medzi zariadeniami a zároveň umožňuje jednoduché rozširovanie systému.

Zariadenia môžu systému prostredníctvom SMBus poskytovať informácie o sebe (typ, verzia, výrobca a mnohé ďalšie). Môžu sa tiež dozvedať o prechodoch systému do šetriaceho režimu a uložiť vopred svoj aktuálny stav, reportovať rôzne typy chýb, reagovať na riadiace príkazy a informovať systém o svojom stave [5].

2.2 Špecifikácia

Ako už bolo spomenuté v úvode, zbernica SMBus vychádza zo špecifikácie zbernice I²C. Preto majú tieto zbernice veľa spoločného. Aj pri SMBus prebieha komunikácia prostredníctvom dvoch obojsmerných liniek. Sériovej dátovej linky SMBDAT a sériovej taktovacej linky SMBCLK. Linky sú pripojené k úrovni H pull-up rezistormi alebo prúdovými zdrojmi a tak spĺňajú zapojenie wired-AND. Každé zariadenie musí mať v rámci jednej zbernice jedinečnú adresu a na zbernici platí vzťah *master-slave* (nadriadený-podriadený). Rovnako aj formát rámcov je na zbernici SMBus rovnaký ako pri I²C.

SMBus špecifikuje tri druhy zariadení, ktoré na zbernici môžu byť pripojené. Je to *slave*, ktorý rovnako ako na I²C prijíma a odpovedá na príkazy; *master*, ktorý zahajuje prenos, generuje taktovací signál, posiela príkazy a ukončuje prenos; *host*, ktorý predstavuje špeciálny *master* a tvorí hlavné rozhranie so systémovým procesorom.

Navyše ešte SMBus povoľuje používanie ďalších dvoch voliteľných liniek pre signály so špeciálnou funkciou, ktoré môžu byť zdieľané všetkými pripojenými zariadeniami.

Prvou je linka pre signál $\overline{\text{SMBSUS}}$, ktorá zmenou úrovne z H na L signalizuje prechod systému do Suspend módu. Tento mód predstavuje stav, kedy väčšina zariadení zastaví svoju činnosť, alebo sa vypne úplne a celý systém odoberá minimum energie. Po obnovení systému sa na tejto linke zmení úroveň opäť z L na H a systém tak obnoví činnosť všetkých suspendovaných zariadení. Pred prechodom do Suspend módu môže systém použiť komunikačné linky SMBDAT a SMBCLK na naprogramovanie reakcie jednotlivých zariadení na príchod signálu $\overline{\text{SMBSUS}}$ (napr. môže naprogramovať vypnutie napájania pre pevné disky a ponechanie napájania pre radič klávesnice po prechode do Suspend módu).

Ďalšiu linku využíva signál pre prerušenie $\overline{\text{SMBALERT}}$. Linka spĺňa zapojenie wired-AND a je aktívna v úrovni L rovnako ako linky SMBDAT a SMBCLK. Signál môže poslať *slave*, ktorý požaduje okamžitú komunikáciu s *mastrom*. Ten prostredníctvom špeciálnej adresy ARA adresuje daného *slavea*, čím ho zároveň požiada o jeho skutočnú adresu. Ak $\overline{\text{SMBALERT}}$ vyvolalo viac zariadení súčasne, arbitrážou pri posielaní adresy vyhrá zariadenia s najvyššou prioritou (s najnižšou adresou). Po potvrdení bitom $\overline{\text{ACK}}$ musí *slave* uvoľniť linku $\overline{\text{SMBALERT}}$. Ak aj po skončení komunikácie *master* zistí aktívny signál $\overline{\text{SMBALERT}}$, opäť zaháji novú komunikáciu adresou ARA.

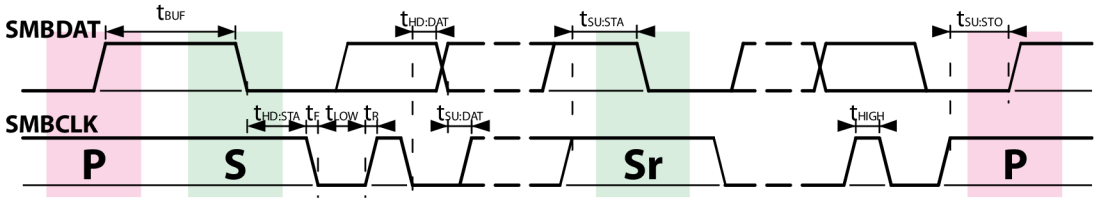
2.3 Časovanie signálov a prenos dát

Základný princíp a spôsob prenosu dát po zbernici SMBus je na drobné rozdiely takmer totožný s prenosom dát po zbernici I²C. Signál sa na dátovej linke môže meniť len pri úrovni L na taktovacej linke, v opačnom prípade zmeny dátového signálu počas úrovne H taktovacieho signálu predstavujú rovnako ako pri I²C špeciálne udalosti ŠTART a STOP.

Formát rámcov prenášaných po SMBus je tiež zhodný s I²C. Prenos je zahájený odoslaním ŠTART, nasledovaný je 7-bitovou adresou a príznakovým bitom $\text{R}/\overline{\text{W}}$, ktorý určí smer prenosu dát. Na rozdiel od zbernice I²C sa na SMBus očakáva od *slavea* potvrdenie prijatej adresy vždy, aj keď nie je schopný zúčastniť sa nasledujúcej komunikácie. Toto je jednoduchý mechanizmus, ako zistiť na zbernici SMBus prítomnosť odnímateľných zariadení. Okrem indikácie, že *slave* je zaneprázdnený a nemôže ďalej komunikovať na zbernici, využíva SMBus potvrdzovací bit $\overline{\text{ACK}}$ aj na signalizáciu prijatých poškodených dát,

alebo prijatého nesprávneho príkazu. SMBus týmto poskytuje jedinú metódu signalizácie potreby opakovaného prenosu dát.

Tak, ako na I²C, aj pri SMBus sa využíva rovnaká metóda synchronizácie taktovacieho signálu na linke SMBCLK v prípade pripojenia niekoľkých *mastrov* na jednu zbernicu súčasne. Taktiež mnohonásobný prístup a problém kolízii sa na zbernici SMBus rieši arbitrážou. Existuje tu aj spomaľovanie prenosu dát pomalšími zariadeniami predlžovaním taktovacieho signálu na úrovni L (stretching), avšak na rozdiel od I²C, existuje tu obmedzenie v maximálnej dobe trvania, ktoré je uvedené nižšie.



Obr. 2.1: Časovanie signálov na zbernici SMBus.

Na rozdiel od zbernice I²C je na zbernici SMBus striktnejšie špecifikované časovanie a úrovne napätí signálov na linkách. Obr. 2.1 znázorňuje priebeh signálov a v tab. 2.1 sú ich konkrétne hodnoty udávané špecifikáciou.

Tab. 2.1: Časovanie signálov na zbernici SMBus.

Symbol	Parameter	Obmedzenia		Jednotka
		Min	Max	
f_{SMB}	Taktovacia frekvencia SMBus	10	100	kHz
t_{BUF}	Pauza medzi STOP a novým ŠTART	4,7	-	µs
$t_{HD:STA}$	Čas od príchodu (opakovaného) ŠTART a prvým taktom SMBCLK	4,0	-	µs
$t_{SU:STA}$	Čas od posledného taktu SMBCLK a opakovaného ŠTART	4,7	-	µs
$t_{SU:STO}$	Čas od posledného taktu SMBCLK a STOP	4,0	-	µs
$t_{HD:DAT}$	Doba, počas ktorej ešte musí zostať úroveň SMBDAT konštantná po príchode L na SMBCLK	300	-	ns
$t_{SU:DAT}$	Doba, počas ktorej už musí byť úroveň na SMBDAT konštantná pred príchodom H na SMBCLK	250	-	ns
$t_{TIMEOUT}$	Čas potrebný na detekciu low timeout	25	35	ms
t_{LOW}	Trvanie úrovne L	4,7	-	µs
t_{HIGH}	Trvanie úrovne H	4,0	-	µs
$t_{LOW:SEXT}$	Predlžovanie úrovne L na SMBCLK (*slave*)	-	25	ms
$t_{LOW:MEXT}$	Predlžovanie úrovne L na SMBCLK (*master*)	-	10	ms
t_F	Doba spádovej hrany	-	300	ns
t_R	Doba nábežnej hrany	-	1.000	ns
t_{POR}	Doba, počas ktorej musí byť zariadenie pripravené na komunikáciu po pripojení napájania	-	500	ms

2.4 Adresovanie

Ako už bolo spomínané, každý *slave* pripojený na zbernicu musí mať jedinečnú adresu. Na zbernici je možné používať 7-bitové adresy a niekoľko špeciálnych adries.

Zo špecifikácie I²C sú prevzaté všetky špeciálne adresy a SMBus ich rozširuje ešte o niekoľko ďalších. Patrí sem adresa *hosta* (`0001 000b`), ktorá má s ohľadom na arbitrážne riešenie kolízii najvyššiu prioritu (je najnižšou možnou adresou), adresa ARA (`0001 100b`), čo je špeciálna adresa používaná pri prerušení signálom $\overline{\text{SMBALERT}}$ a adresa DDA (`1100 001b`), ktorú používa protokol ARP pre dynamické prideľovanie adries na SMBus.

2.5 Komunikácia

Typické zariadenie SMBus obsahuje súbor príkazov umožňujúcich čítanie a zápis dát. Všetky príkazy sú dlhé 8 bitov (1 bajt), avšak argumenty a návratové hodnoty môžu mať variabilnú dĺžku. Použitie príkazu, ktorý neexistuje alebo nie je podporovaný, môže spôsobiť chybové hlásenie.

Existuje jedenásť príkazových protokolov, ktoré by mali byť podporované každým SMBus zariadením. Sú to: Quick Command (Rýchly Príkaz), Send Byte (Odoslanie Bajtu), Receive Byte (Príjem Bajtu), Write Byte (Zápis Bajtu), Write Word (Zápis Slova), Read Byte (Čítanie Bajtu), Read Word (Čítanie Slova), Process Call (Volanie Procesu), Block Read (Čítanie Bloku), Block Write (Zápis Bloku) a Block Write-Block Read Process Call (Zápis Bloku-Čítanie Bloku Volanie Procesu).

2.6 Kontrola chýb paketov

Mechanizmus PEC zvyšuje spoľahlivosť prenášaných dát a bol zahrnutý do špecifikácie SMBus vo verzii 1.1. Jeho implementácia je voliteľná, avšak napr. počas ARP procesu je vyžadovaná. Zariadenie s implementáciou PEC musí byť schopné komunikovať aj so zariadením bez jeho implementácie. PEC je aplikovaný ako 1 bajt, ktorý je posielaný ako posledný po odoslaní celej správy. Predstavuje zabezpečenie prostredníctvom 8-bitového CRC kódu, ktorý je vypočítaný z cieľovej adresy vrátane bitu $\overline{R/W}$ a celej správy.

Master môže používať PEC kedykoľvek a počas akéhokoľvek prenosu. Nutné je však, aby vopred vedel, či druhá strana (*slave*) tento mechanizmus podporuje. Existuje na to niekoľko metód.

Zariadenie s implementáciou PEC, ktoré pracuje ako *slave*, musí byť vždy schopné spolupracovať s *mastrom* s podporou a aj s *mastrom* bez podpory tohto mechanizmu. Ak je PEC použitý, zariadenie musí byť schopné overiť korektnosť prijatej správy a spracovať ju len v prípade, že prenos bol bezchybný. V prípade chyby musí správu ignorovať a informovať o tom bitom $\overline{NACK}$.

Pri použití PEC je pri každom prenose každou stranou (vysielačom aj prijímačom) počítaný vlastný 8-bitový CRC pre každý paket. Na výpočet sa používa nižšie uvedený polynóm, ktorý je aplikovaný na jednotlivé bity v poradí, ako boli prijaté.

$$C(x) = x^8 + x^2 + x^1 + 1 \tag{2.1}$$

Existuje niekoľko metód počítania CRC pri použití PEC a voľba konkrétnej závisí len od výrobcu zariadenia.

2.7 Protokoly SMBus

Každý prenos správy po zbernici SMBus sa riadi jedným z definovaných SMBus protokolov. SMBus protokol je súbor možných formátov dátových rámcov vytvorených na

špecifikácii I²C. Zariadenia s implementáciou I²C môžu komunikovať na SMBus len v prípade, že sú kompatibilné so špecifikáciou SMBus. Zariadenia, ktoré sa neriadia týmito protokolmi, nemôžu pracovať na zbernici prostredníctvom metód definovaných špecifikáciami SMBus a ACPI.

ACPI definuje štandard rozhrania pre komunikáciu hardvéru a softvéru medzi ovládačom zbernice operačného systému a kontrolérom SMBus na osobnom počítači. Poskytuje tak priamy prístup k zariadeniam pripojeným k zbernici SMBus a komunikáciu s nimi cez ovládač zbernice. Umožňuje tak napr. monitorovanie hardvéru (teplota procesora, otáčky ventilátora apod.) priamo v operačnom systéme cez rozhranie programu [6].

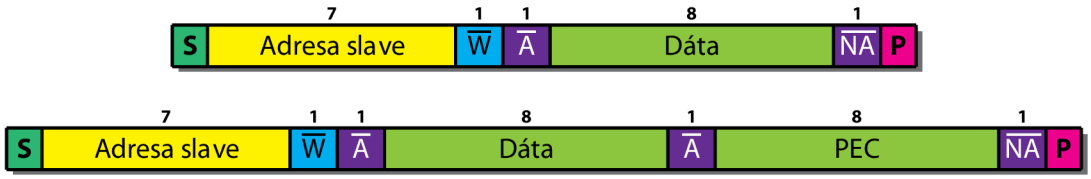
2.7.1 Quick Command

Je zložený len z adresy *slavea* a bitu R/$\overline{W}$. Žiadne ďalšie dáta nie sú už ani vysielané, ani prijímané. Ide o veľmi krátky príkaz, ktorý môže byť použitý napr. na jednoduché vypnutie alebo zapnutie zariadenia. Je vhodný pre veľmi malé zariadenia s limitovanou podporou SMBus špecifikácie.

Obr. 2.2: Protokol Quick Command.

2.7.2 Send Byte

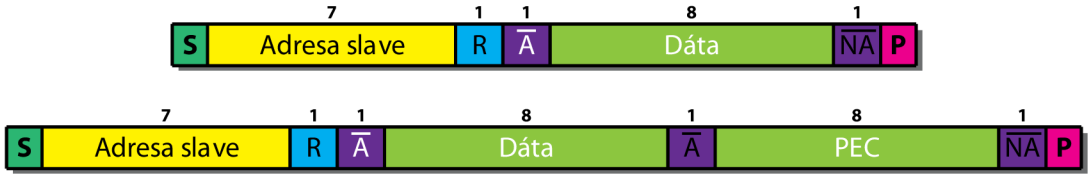
Master odošle na zbernicu adresu *slavea*, ktorému bude smerovaný nasledovný 1 bajt. Ten môže predstavovať napr. až 256 rôznych príkazov (pri použití všetkých 8 bitov), 128 nastavení a vlastností, ktoré je možné zapínať a vypínať (pri použití 7 bitov pre nastavenia a 1 bitu pre zapínanie/vypínanie), alebo údaj reprezentuje priamo napr. úroveň hlasitosti apod.



Obr. 2.3: Protokol Send Byte bez PEC (hore) a s PEC (dole).

2.7.3 Receive Byte

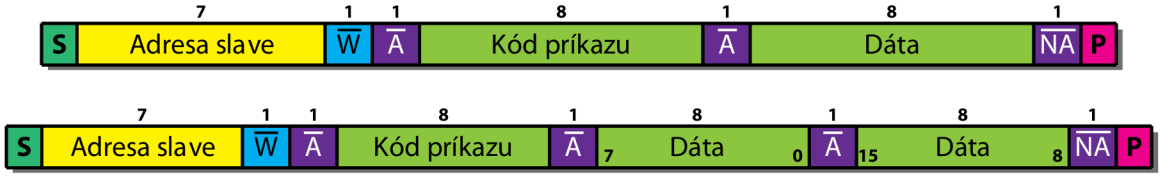
Tento protokol je podobný predchádzajúcemu, rozdiel je len v smere prenosu dát. V tomto prípade dáta alebo informácie, ktoré požaduje *master*, posiela *slave*. Prenos dát zakončuje *master* zaslaním bitu NACK po prijatí požadovaného bajtu.



Obr. 2.4: Protokol Receive Byte bez PEC (hore) a s PEC (dole).

2.7.4 Write Byte/Word

Prvý bajt protokolu určuje kód príkazu. Ďalší bajt, prípadne dva predstavujú dáta, ktoré majú byť zapísané. Rovnako ako v predchádzajúcich dvoch protokoloch, aj tento existuje s podporou a aj bez podpory PEC.



Obr. 2.5: Protokol Write Byte (hore) a Write Word (dole).

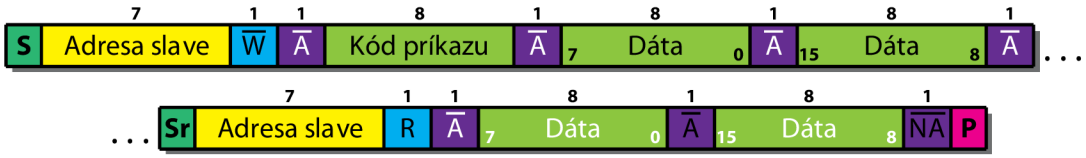
2.7.5 Read Byte/Word

Čítanie bajtu alebo slova je o čosi zložitejšie ako ich zápis v protokole Write Byte/Word. Na začiatku odošle *master* na zbernicu adresu cieľa a kód príkazu. Potom pošle opakovaný ŠTART a opäť adresu cieľa. Po adrese nasleduje príznak čítania a od tohto momentu začína posielať na zbernicu dáta *slave*, ktorý odošle jeden až dva bajty. Prenos dát môže zastaviť *master* jedine potvrdzovacím bitom NACK a až potom môže ukončiť celú komunikáciu zaslaním STOP. Aj tento protokol existuje s podporou a aj bez podpory PEC.

Obr. 2.6: Protokol Read Byte.

2.7.6 Process Call

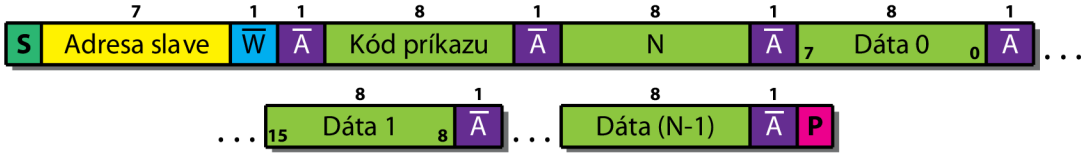
Tento protokol slúži na odosielanie dát s čakaním na odpoveď. *Master* odošle po adrese príznak zápisu a zapíše kód príkazu a niekoľko ďalších bajtov. Potom pošle opakovaný ŠTART, opäť adresu a príznak čítania. Od tohto okamihu bude očakávať odpoveď od *slavea*, ktorý odošle jeden až dva bajty. Rovnako, ako v predchádzajúcich protokoloch, aj tento existuje s podporou a aj bez podpory PEC.



Obr. 2.7: Protokol Process Call.

2.7.7 Block Write

Zápis bloku začína *master* odoslaním adresy cieľa s príznakom zápisu, ktorá je nasledovaná kódom príkazu. Nasledujúci bajt N, ktorý *master* odošle, určí dĺžku bloku dát v bajtoch, ktorý bude nasledovať. Táto dĺžka nesmie byť nulová a nesmie presiahnuť 32 bajtov. V prípade, že je použitý PEC, do dĺžky nie je započítaný.



Obr. 2.8: Protokol Block Write.

2.7.8 Block Read

Protokol Čítanie Bloku je podobný predchádzajúcemu protokolu. Rozdielom je len opakovaný ŠTART a príznak čítania odoslaný po adrese cieľa, ktorý zmení smer toku dát. *Master* ukončuje prenos dát potvrdzovacím bitom NACK, za ktorým ihneď odošle STOP. Aj tento protokol podporuje PEC.

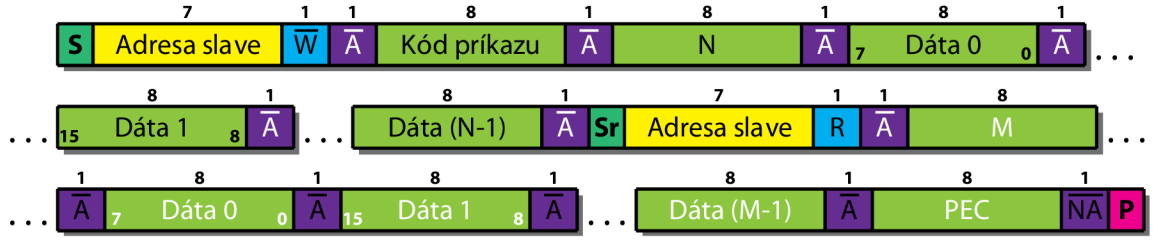
Obr. 2.9: Protokol Block Read.

2.7.9 Block Write-Block Read Process Call

Tento protokol sa skladá z dvoch častí. Prvá pozostáva z odoslania cieľovej adresy s príznakom zápisu, za ktorou nasleduje bajt N určujúci veľkosť nasledujúceho bloku dát v bajtoch. Potom *master* odošle samotný blok dát.

Druhá časť sa začína odoslaním opakovaného ŠTART a adresy cieľa s príznakom čítania. Nasledujúci bajt M odoslaný *slaveom* určuje veľkosť bloku dát v bajtoch, ktorý *slave* ešte odošle.

N a M musia obsahovať hodnoty väčšie ako nula a ich súčet nesmie presiahnuť 32. V prípade tohto protokolu sa odporúča použitie PEC, ktorý je vypočítaný z celého prenosu a je posielaný *slaveom* ako posledný bajt. Do veľkosti N a M sa nepočíta.



Obr. 2.10: Protokol Block Write-Block Read Process Call s PEC.

2.7.10 Host Notify

Protokol Host Notify (Notifikácia *Hosta*) poskytuje jediný spôsob, ako môže *master* odoslať dáta *hostovi*. Ide o čiastočne upravený Write Word protokol, v ktorom *master* namiesto kódu príkazu posiela svoju adresu. Tento protokol môže byť použitý len v prípade, že je *host* v režime *slave*.

Komunikáciu zahajuje *master* odoslaním adresy *hosta* (`0001 000b`) s príznakovým bitom zápisu. Kód príkazu nahradzuje v tomto prípade adresa odosielateľa, vďaka ktorej *host* môže identifikovať odosielateľa. Za ňou nasledujú dva bajty, ktoré reprezentujú stav zariadenia. Ich význam určuje priamo výrobca zariadenia.

Každý *host* musí podporovať tento protokol a okrem neho môže mať implementovanú ešte linku pre signál $\overline{\text{SMBALERT}}$.

Obr. 2.11: Protokol Host Notify.

2.8 Porovnanie zbernice SMBus s I²C

Medzi zbernicami SMBus a I²C existuje niekoľko zásadných rozdielov. Ako už bolo vyššie spomenuté, najväčším rozdielom je časovanie signálov, ktoré je na dátovej a taktovacej linke zbernice SMBus určené striktnejšie ako na I²C.

Na zbernici SMBus je definovaná minimálna frekvencia taktovacieho signálu (f_{SMB}) a to 10 kHz. Zbernica I²C nešpecifikuje tento parameter.

SMBus definuje minimálny čas trvania úrovne signálu na dátovej linke ($t_{HD:DAT}$), ktorá musí zostať konštantná po zmene úrovne z H na L na taktovacej linke ešte po dobu minimálne 300 ns. I²C definuje tento čas na 0 ns.

Maximálna frekvencia taktovacieho signálu je na zbernici SMBus (f_{SMB}) definovaná na 100 kHz. I²C podporuje tri operačné módy a to 100 kHz pre štandardný mód, 400 kHz pre rýchly mód a 3,4 MHz pre veľmi rýchly mód.

Na SMBus je určená maximálna dovolená doba trvania úrovne L na taktovacej linke ($t_{TIMEOUT}$) a to 35 ms, ktorá po prekročení 25 ms predstavuje špeciálnu udalosť. I²C nešpecifikuje žiadne takéto obmedzenie.

SMBus špecifikuje maximálne trvanie úrovne L na taktovacej linke počas prenosu dát na 25 ms pre *slave* ($t_{LOW:SEXT}$) a 10 ms pre *master* ($t_{LOW:MEXT}$). Opäť na I²C nie sú špecifikované podobné parametre.

Zbernica SMBus definuje čas nábežnej (t_R) a spádovej (t_F) hrany signálov, čo pri I²C definované nie je.

3 ZBERNICA PMBUS

3.1 Úvod

Zbernica Power Management Bus, označovaná skôr PMBus™, vychádza zo zbernice SMBus a je zameraná hlavne na digitálny manažment napájacích zdrojov. Ide o otvorený štandard vyvinutý a vlastnený System Management Interface Forum (SM-IF). Rovnako ako SMBus, aj PMBus poskytuje relatívne pomalý 2-vodičový komunikačný protokol založený na špecifikácii I²C [7].

3.2 Vlastnosti

Na najnižšej úrovni PMBus kopíruje špecifikáciu SMBus verzie 1.1 s drobnými odlišnosťami a vylepšeniami.

Na PMBus môže taktovací signál na taktovacej linke dosahovať až 400 kHz, pri SMBus je maximálna frekvencia taktovacieho signálu obmedzená na 100 kHz.

Zbernica PMBus umožňuje prenos blokov dát do veľkosti až 255 bajtov, na SMBus je limitovaný na 32 bajtov.

Rovnako ako na zbernici SMBus verzie 2.0, je na PMBus dovolené len 7-bitové adresovanie.

Niektoré príkazy PMBus využívajú protokoly Block Write/Read Process Call zbernice SMBus verzie 2.0.

Taktiež signál $\overline{\text{SMBALERT}}$ a Host Notify protokol zbernice SMBus môžu byť na zbernici PMBus použité.

Od zariadení PMBus je požadované, aby podporovali Group protokol. Ten poskytuje možnosť odloženia akcie (činnosti) zariadení až do príchodu ukončovacieho STOP. *Master*/*host* tak môže postupne nastaviť všetky zariadenia a spoločne zosynchronizovať (spustiť) ich akcie. V špecifikácii SMBus takýto protokol nie je.

PMBus definuje Extended Command protokol, ktorý rozširuje kódy príkazov o ďalší bajt.

4 ZBERNICA SPI

4.1 Úvod

Zbernica SPI™ (Serial Peripheral Interface) predstavuje synchrónnu dátovú linku a svoje pomenovanie dostala od spoločnosti Motorola [8]. Vychádza zo základnej verzie zbernice Microwire™ vlastnenej spoločnosťou National Semiconductor, ktorá v súčasnosti predstavuje už len jej obmedzenú verziu [9].

4.2 Vlastnosti

Rovnako ako pri zbernici I²C, aj tu medzi zariadeniami pripojenými na zbernicu existuje vzťah *master* (nadriadený) – *slave* (podriadený). Dátový prenos vždy iniciuje, riadi a ukončuje *master*. Na rozdiel od zbernice I²C a SMBus má zbernica SPI niekoľko výhod:

- Komunikácia prebieha vo full-duplex móde,
- umožňuje vyššie prenosové rýchlosti,
- kompletná flexibilita komunikačného protokolu; nie je presne určená veľkosť a formát dátových rámcov,
- jednoduchší hardvér; menšie požiadavky na spotrebu, *slave* nepotrebujú jedinečnú adresu, nie je potrebné riešiť kolízie.

SPI má aj niekoľko nevýhod:

- Pripojenie na zbernicu vyžaduje až 4 piny, prípadne 3 piny v špeciálnych prípadoch,
- žiadne hardvérové riadenie toku,
- žiadne hardvérové potvrdzovanie od *slave*; *master* tak môže odosielať dáta na prázdnu zbernicu,
- podpora len jedného *mastra* na jednej zbernici,
- neexistuje tu žiadny formálny štandard.

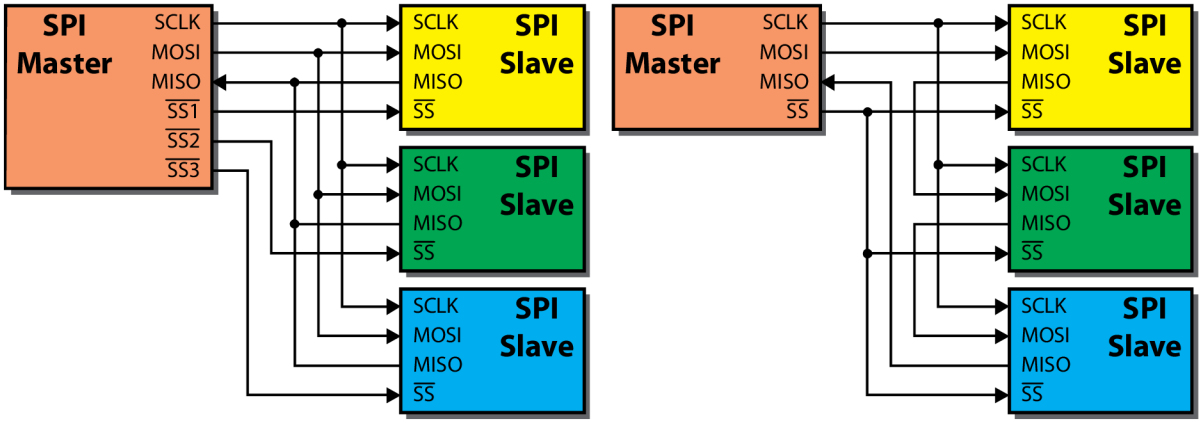
Zariadenia sú k zbernici pripojené prostredníctvom štyroch liniek. Sú to:

- Sériová taktovacia linka SCLK,
- sériová dátová výstupná linka (z pohľadu *mastra*) MOSI,
- sériová dátová vstupná linka (z pohľadu *mastra*) MISO

- a linka pre výber *slavea* $\overline{SS}$; v špeciálnych prípadoch nemusí byť použitá.

4.3 Špecifikácia

Zbernica môže pracovať len s jedným *mastrom* a s jedným alebo niekoľkými *slaveami*. Ak je pripojený k zbernici len jeden *slave*, $\overline{SS}$ pin je možné na jeho strane spojiť s úrovňou L v prípade, že to umožňuje. Niektoré zariadenia ale naopak vyžadujú na linke $\overline{SS}$ zmenu úrovne z H na úroveň L, čím sa zariadenie aktivuje (napr. prevod v ADC). V prípade, že je takýchto zariadení pripojených na zbernicu viac, je nutné, aby *master* mohol poskytnúť nezávisle signály $\overline{SS}$ pre každý *slave*. Príklad typického a špeciálneho zapojenia Daisy-chained je zobrazený na obr. 4.1.



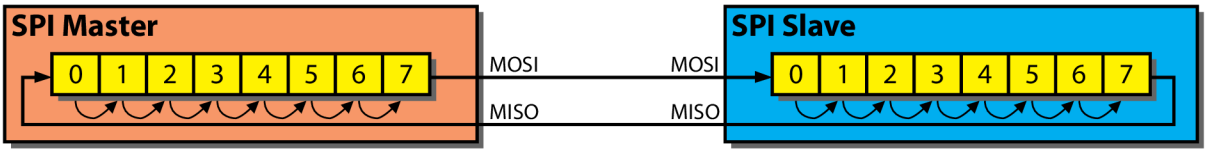
Obr. 4.1: Zapojenie SPI: vľavo typické, vpravo zreťazené (Daisy-chained).

4.4 Časovanie signálov a prenos dát

Master pred zahájením prenosu dát zvolí frekvenciu taktovacieho signálu s ohľadom na maximálnu pracovnú frekvencia *slavea*, s ktorým plánuje komunikovať. Štandardne sa pohybuje v rozsahu 1-70 MHz. Nastavením úrovne L na konkrétnej linke $\overline{SS}$ zvolí *master* konkrétneho *slavea* pre nasledovnú komunikáciu. Ostatné zariadenia (s neaktívnym $\overline{SS}$) musia zostať na linkách MOSI a MISO pasívne bez akejkoľvek reakcie. Niektoré zariadenia potrebujú pred samotnou komunikáciou nejaký čas (napr. na prevod v ADC), preto *master* v takomto prípade môže spustiť taktovací signál s oneskorením jednej periódy.

Prenos dát na SPI prebieha vo full-duplex móde. Počas jednej periódy taktovacieho signálu tak posiela *master* po linke MOSI bit k *slaveu* a ten zároveň posiela bit po linke MISO k *mastru*. Principiálne je tento proces realizovaný pomocou dvoch posuvných registrov. Jeden je na strane *mastra* a jeden na strane *slavea*. Registre sú navzájom prepojené

do kruhu tak, ako ukazuje obr. 4.2. Ako prvý opúšťa register vo vysielacom zariadení vždy bit s najväčšou váhou a vstupuje na prijímacej strane na miesto bitu s najnižšou váhou, pričom posunie ostatné bity vpravo. Bity sú takto posúvané a posielané až do posledného (s najmenšou váhou). Po odoslaní posledného bitu si každá strana uloží obsah registra na spracovanie a v prípade potreby registre naplnia novými dátami a proces prenosu zopakujú. Dĺžka prenášaného slova je variabilná a závisí len od výrobcov a od typu vzájomne komunikujúcich zariadení (štandardne: 8 bitov; ADC: 12 bitov; audio kodeky: 16 bitov apod.). Niektoré zariadenia sú navrhnuté tak, že po prekročení určitého počtu taktovacích impulzov začnú ignorovať ďalšiu komunikáciu na zbernici SPI. Niektoré zariadenia naopak naďalej „posúvajú“ prijaté dáta na svoj výstup. Táto vlastnosť opäť závisí len od výrobcu a od typu zariadenia.



Obr. 4.2: Prenos dát po SPI prostredníctvom dvoch posuvných registrov.

Ak *master* už ďalej nechce komunikovať so *slaveom*, ukončí generovanie taktovacieho signálu a uvoľní *slavea* uvoľnením konkrétnej linky $\overline{SS}$.

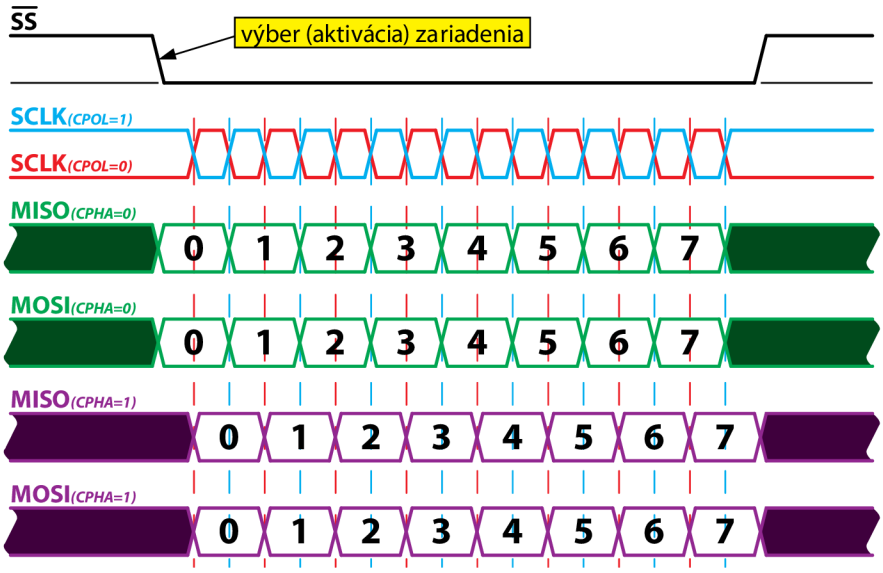
4.5 Taktovací signál

Pred každým prenosom dát je nutné, aby *master* nakonfiguroval polaritu CPOL a fázu CPHA taktovacieho signálu s ohľadom na prenášané dáta. V tab. 4.1 je popísané, ako tieto parametre ovplyvňujú prenos dát po zbernici SPI a na obr. 4.3 sú znázornené časové priebehy signálov s konkrétnymi nastaveniami týchto parametrov.

Tab. 4.1: Význam parametrov CPOL a CPHA.

	CPHA=0	CPHA=1
CPOL=0	Dáta sú platné pri nábežnej hrane taktovacieho signálu a môžu sa meniť len pri spádovej hrane.	Dáta sú platné pri spádovej hrane taktovacieho signálu a môžu sa meniť len pri nábežnej hrane.
CPOL=1	Dáta sú platné pri spádovej hrane taktovacieho signálu a môžu sa meniť len pri nábežnej hrane.	Dáta sú platné pri nábežnj hrane taktovacieho signálu a môžu sa meniť len pri spádovej hrane.

Nastavenie CPHA=0 spôsobí, že signál je na linkách MOSI a MISO platný (vzorkovaný) už pri prvej hrane taktovacieho signálu. Naopak pri nastavení CPHA=1 budú dáta platné až pri druhej hrane taktovacieho signálu.



Obr. 4.3: Časovanie signálov na zbernici SPI.

4.6 Zapojenie Daisy-chained

Niektoré zariadenia sú navrhnuté tak, aby mohli byť súčasťou zreťazeného zapojenia daisy-chained. Ako je zobrazené v pravej časti obr. 4.1, výstup MOSI z *mastra* je pripojený k vstupu MOSI prvého *slavea*, jeho výstup MISO je prepojený na vstup MOSI ďalšieho atď. Výstup MISO posledného *slavea* uzatvára reťaz prepojením na vstup MISO *mastra*. Posielané dáta potom prechádzajú celou reťazou, ktorá je poskladaná z posuvných registrov jednotlivých zariadení a spolu vytvárajú jeden veľký posuvný register. Prvý bajt odoslaný *mastrom* sa teda dostane k druhému *slaveu* až pri odosielaní druhého bajtu a k tretiemu *slaveu* až pri odosielaní tretieho bajtu atď. Takéto zapojenie využíva napr. SGPIO a JTAG.

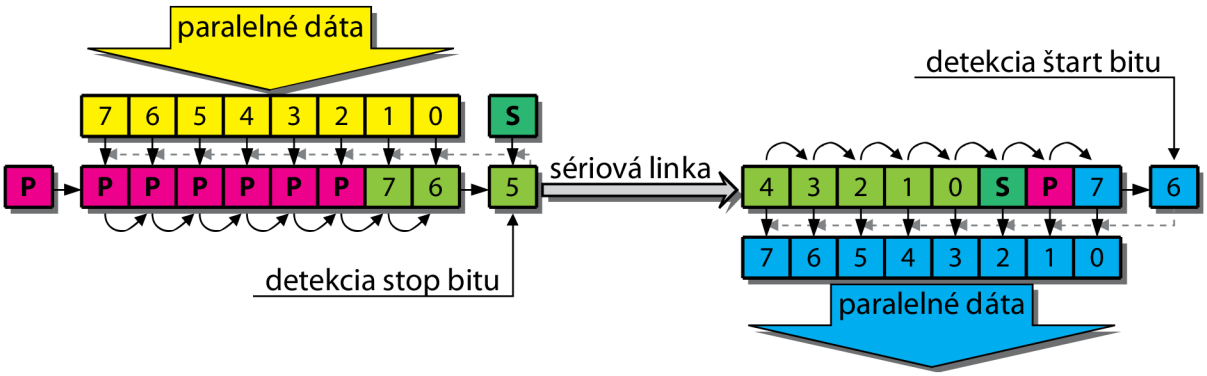
5 ROZHRANIE UART

5.1 Úvod

Universal asynchronous receiver/transmitter býva súčasťou riadiacich systémov (mikrokontrolérov), ktorým tak umožňuje sériovú komunikáciu s okolím. Predstavuje základ pre niektoré ďalšie štandardy ako napr. RS-232, RS-485 a iné [10].

5.2 Prenos sériových dát

UART v podstate predstavuje prevodník paralelne interpretovaných dát na sériovú formu a naopak. Prevod je principiálne realizovaný pomocou posuvného registra a pre každý smer prenosu dát je potrebný jeden takýto register. Na obr. 5.1 je vidieť príklad prevodu paralelných dát na sériové na strane vysielača, prenos sériových dát po jednej sériovej linke a spätný prevod dát na paralelnú formu na strane prijímača.



Obr. 5.1: Prevod paralelných dát na sériové pomocou posuvného registra.

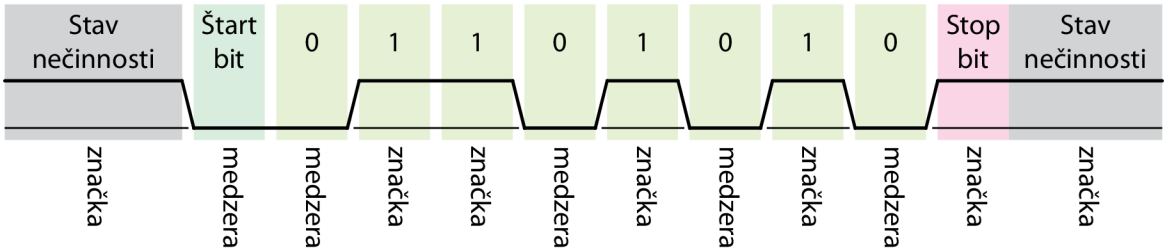
Na komunikáciu sa využívajú dve sériové jednosmerné dátové linky. Linka RxD, ktorá je určená pre príjem dát a linka TxD, ktorá je určená pre odosielanie. Prostredníctvom UART môžu byť dáta prenášané v half-duplex a aj vo full-duplex móde.

UART poskytuje možnosť vzájomného prepojenia koncových staníc formou bod-bod a bod-multibod. Prvý prípad predstavuje obojsmernú vzájomnú komunikáciu medzi dvoma koncovými stanicami. Druhý prípad je častejší najmä v oblasti riadenia a predstavuje prepojenie, kde jedna koncová stanica (riadiaca) posiela dáta niekoľkým koncovým staniciam (riadeným) prostredníctvom spoločnej zbernice.

UART zvyčajne nerobí priamo konverziu úrovní vlastného signálu na úrovne signálov používaných inými zariadeniami a technológiami (RS-232, RS-485, IrDA, Bluetooth apod.). Typicky sa pre tento účel používajú špeciálne obvody (napr. MAX232).

Aby bolo možné všeobecne popisovať priebeh signálov prenášaných cez UART bez ohľadu na použitú technológiu, význam úrovne L je reprezentovaný „medzerou“ (z angl. „space“) a význam úrovne H „značkou“ (z angl. „mark“).

Na obr. 5.2 je znázornený asynchrónny prenos dát cez UART. Ako je vidieť, možno ho rozdeliť na tri časti. Prenos sa začína štart bitom – medzerou. Ten je reprezentovaný zmenou polarity (zo stavu nečinnosti). Štart bit prijímacej strane slúži na synchronizáciu. Nasleduje samotný prenos dát, ktorý môže byť dlhý päť až osem bitov. Ako prvý sa vždy posiela bit s najnižšou váhou. Dáta môžu byť zabezpečené ešte paritným bitom. Jeho použitie je voliteľné. Prenos zakončuje stop – značka, ktorý musí byť dlhý minimálne jeden, jeden a pól, alebo dva bity. Stop bit(y) je totožný so stavom nečinnosti a teda poskytuje prijímacej strane viac času pred prenosom ďalšieho znaku. Aby pri prenose väčšieho objemu dát nedochádzalo k chybám (stratám dát), býva prijímacia strana vybavená vyrovnávacou pamäťou typu FIFO.

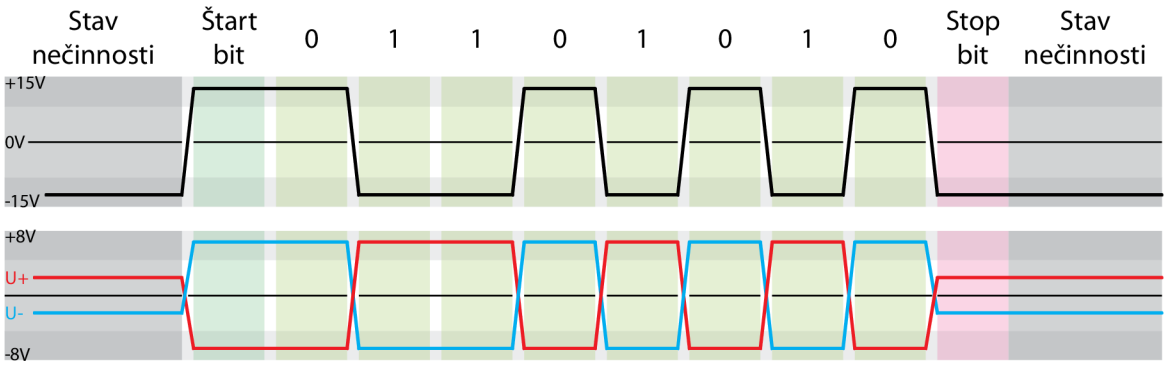


Obr. 5.2: Časový priebeh prenosu znaku „V“ (`0101 0110b`) na UART.

V posledných rokoch je UART spomínaný najmä v súvislosti so štandardom RS-232 a to v oblasti prenosu dát v priemyselných systémoch. Môže byť použitý napr. na komunikáciu mikrokontroléru a počítača PC.

5.3 Štandardy RS-232 a RS-485

Tieto štandardy vychádzajú priamo z rozhrania UART. Bližšie však špecifikujú niektoré ďalšie vlastnosti ako napr. úrovne napätí, prúdové zaťaženie liniek, časovanie a pod. Príklad prenosu dát po RS-232 a RS-485 a znázornenie úrovní napätí pri prenose je možné vidieť na obr. 5.3.



Obr. 5.3: Časový priebeh prenosu znaku „V“ na RS-232 (hore) a RS-485 (dole).

6 ANALÝZA SÉRIOVÝCH ZBERNÍC

6.1 Úvod

V súčasnosti existuje a je ľahko dostupných mnoho nástrojov na efektívnu analýzu sériových zberníc. Niektoré z nich dokážu analyzovať a sledovať prevádzku len na vopred známej zbernici s vopred známym zapojením komunikačných liniek. Niektoré sú naopak schopné analyzovať aj úplne neznáme zbernice, čo je aj prípad nižšie navrhnutého analyzátora.

6.2 Riadiaca jednotka

Hlavnou časťou celého analyzátora je riadiaca jednotka. Tú je možné univerzálne pripojovať na akúkoľvek zbernicu či rozhranie, analyzovať tak ich činnosť a zobrazovať získané výsledky prostredníctvom užívateľskej aplikácie v PC. Ako základ riadiacej jednotky analyzátora bol zvolený mikrokontrolér PIC24HJ128GP204 od spoločnosti Microchip. Ide o jeden z najvýkonnejších 16-bitových mikrokontrolérov z ponuky tejto spoločnosti, ktorý okrem iného obsahuje aj hardvérové implementácie zberníc I^2C, SPI a rozhrania UART. Pri návrhu riadiacej jednotky sa však vychádzalo najmä z požiadavky možnej analýzy aj neznámej zbernice s neznámym zapojením liniek. A z tohto dôvodu nebolo možné použiť hardvérové implementácie zberníc, kde by sa požadovalo správne pripojenie a samozrejme aj dodržanie istých štandardov. Za celý priebeh analýzy a sledovanie komunikácie na zberniciach teda zodpovedá len riadiaci program mikrokontroléra a užívateľská aplikácia.

Neznámu zbernicu je možné analyzovať niekoľkými spôsobmi. Na začiatku môžu byť neznáme analyzované linky vzorkované riadiacou jednotkou, ktorá ďalšie spracovanie získaných dát už nevykonáva a v tejto úlohe ju plne zastupuje priamo užívateľská aplikácia v PC. Takéto riešenie je efektívne len v prípade analýzy liniek s relatívne pomalými zmenami signálu, kde je tak možné používať nižšie vzorkovacie frekvencie. V opačnom prípade by sa od riadiacej jednotky vyžadovalo odosielanie veľkého množstva neanalyzovaných dát k PC.

Druhou možnosťou je ponechanie čiastočnej analýzy neznámych liniek na riadiacej jednotke a zvyšok na užívateľskej aplikácii. Riadiaca jednotka môže v takomto prípade zisťovať napr. dobu trvania jednotlivých úrovni a tieto informácie posielať na ďalšie spracovanie do PC. Takéto riešenie síce vyžaduje od riadiacej jednotky vyšší výkon, avšak množstvo prenášaných dát do PC môže byť o to menšie.

Posledným riešením je ponechanie celej analýzy na riadiacej jednotke, ktorá by posielala užívateľskej aplikácii už priamo výsledky na zobrazenie. Riadiaca jednotka by v tomto prípade musela byť dostatočne výkonná, avšak do PC by bolo posielaných len minimum dát.

Navrhnutý analyzátor patrí do druhej skupiny, i keď rozsah analýzy vykonávanej riadiacou jednotkou by bolo možné podľa konkrétnych požiadaviek kladených na celkovú analýzu zmeniť zmenou riadiaceho programu (firmware) v riadiacej jednotke a úpravou užívateľskej aplikácie. Ako komunikačný kanál pre prenos získaných dát od riadiacej jednotky do PC bol zvolený USB port.

6.2.1 Konštrukčné riešenie

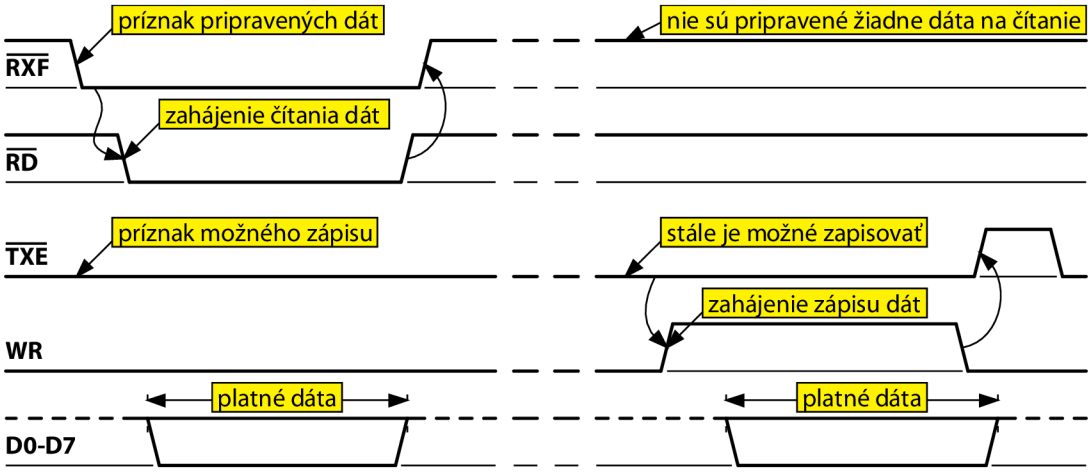
Ako už bolo spomenuté vyššie, srdcom riadiacej jednotky je mikrokontrolér PIC24HJ128GP204. Hlavným dôvodom výberu práve tohto typu je jeho výkon. Mikrokontrolér je vybavený štandardnou inštrukčnou sadou mikrokontrolérov rady PIC24H, avšak poskytuje výkon v spracovaní programu s rýchlosťou až 40.000.000 inštrukcií za 1 sekundu pri použití kryštálu s frekvenciou 10 MHz a vnútorného PLL (Phase-Locked Loop) obvodu [11]. Mikrokontrolér okrem pripojenia primárneho kryštálu umožňuje aj pripojenie sekundárneho hodinového kryštálu s pracovnou frekvenciou 32,768 kHz [12], [16]. To poskytuje okrem dohľadu nad správnym fungovaním primárneho oscilátora a iných funkcií aj privedenie generovaného signálu na vstup vnútorného časovača [14], ktorý v obvode riadiacej jednotky okrem riadenia blikania LED diód robí aj dohľad nad hlavným programom pri analýze.

a) Rozhranie pre komunikáciu s PC

Aby bolo možné efektívne využiť pri analýze výkon mikrokontroléra, bolo potrebné zvoliť aj dostatočne rýchle rozhranie pre komunikáciu s PC. Ako najvhodnejšie riešenie sa núkalo použitie univerzálnej sériovej zbernice USB. Spoločnosť Microchip má v ponuke produktov aj mikrokontroléry s takýmto rozhraním, tie ale neposkytujú taký výkon ako zvolený mikrokontrolér, ktorý však USB rozhranie implementované nemá. Z toho dôvodu bolo nutné nejakým spôsobom prispôsobiť jeho najrýchlejšie dostupné rozhranie k rozhraniu USB. Najviac pre tento účel vyhovoval integrovaný port PMP (Parallel Master Port), ktorý predstavuje 8-bitové paralelné rozhranie. Keďže s týmto portom zdieľajú spoločné piny mikrokontroléra aj obe implementované rozhrania zberníc I^2C, nie je možné v budúcnosti počítať s ich využitím napríklad pri rozširovaní analyzátora o funkciu konverzie. Ako prevodník paralelných dát na USB bol zvolený integrovaný obvod FT245BL od spoločnosti

FTDI Chip. Tento obvod poskytuje prevod medzi USB a paralelným FIFO rozhraním s rýchlosťou prenosu dát až do 1 MB/s a disponuje vnútornou vyrovnávacou pamäťou s veľkosťou 384 bajtov pre zápis a 128 bajtov pre čítanie [18]. Hoci PMP umožňuje konfiguráciu veľkého množstva parametrov pre prispôsobenie komunikácie pre širokú škálu aplikácií [15], nie je možné nastaviť ho pre plnú hardvérovú podporu komunikácie s obvodom FT245BL. Prevodník je s mikrokontrolérom prepojený prostredníctvom ôsmych dátových, dvoch riadiacich a dvoch stavových liniek. Obidve stavové linky musia byť obsluhované softvérovo, pretože PMP neposkytuje žiadne metódy pre ich obsluhu. Integrovaný obvod FT245BL je aj súčasťou modulu UMP2, ktorý je vyrábaný spoločnosťou ASIX. Ten v sebe zahŕňa už aj pamäť EEPROM, kryštál a všetky ďalšie diskrétne súčiastky potrebné pre správnu činnosť integrovaného obvodu FT245BL. Modul UMP2 umožňuje pripojenie ku všetkým potrebným linkám tohto obvodu prostredníctvom prevedenia DIP28 [19]. Kvôli jednoduchosti aplikácie a možnosti využiť UMP2 modul v prípade potreby aj v iných obvodoch, bol použitý aj v riadiacej jednotke analyzátora. Z toho dôvodu modul nie je priamo pripájkovaný na plošný spoj riadiacej jednotky, ale je k nemu pripojený prostredníctvom pätice DIP28. Rozhranie USB je vedené z modulu UMP2 cez päticu na plošný spoj riadiacej jednotky, kde je zakončené konektorom mini-USB.

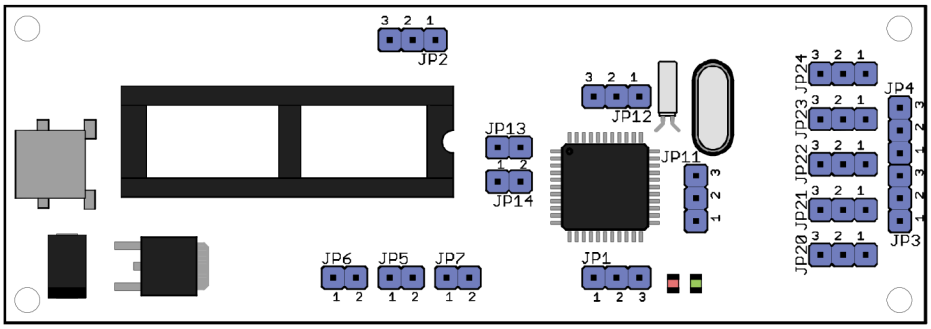
Príklad prenosu dát medzi mikrokontrolérom a prevodníkom je znázornený na obr. 6.1. Na začiatku mikrokontrolér zistí prostredníctvom stavovej linky $\overline{\text{RXF}}$, že vo vnútornej vyrovnávacej pamäti prevodníka pre odosielanie dát čakajú dáta na prečítanie. Mikrokontrolér zahájí čítanie nastavením riadiacej linky $\overline{\text{RD}}$ na úroveň L. Prevodník v tomto okamihu pripraví na linkách D0 až D7 dáta a mikrokontrolér ich prečíta. Potom mikrokontrolér nastaví na riadiacej linke $\overline{\text{RD}}$ úroveň H a v prípade, že prevodník už nemá ďalšie dáta na odoslanie, uvoľní stavovú linku $\overline{\text{RXF}}$ na úroveň H. V prípade, že mikrokontrolér požaduje odoslanie dát, skontroluje stavovú linku $\overline{\text{TXE}}$, ktorá informuje, či je možné dáta odoslať. Ak je na linke úroveň L, mikrokontrolér zahájí vysielanie dát nastavením riadiacej linky WR na úroveň H a zároveň nastaví dáta na linkách D0 až D7. Dáta sú prenesené do vnútornej vyrovnávacej pamäte prevodníka pre príjem dát a mikrokontrolér ukončuje prenos dát nastavením riadiacej linky WR na úroveň L. V čase, keď po linkách D0 až D7 neprebieha žiadna komunikácia s prevodníkom, sú tieto linky v stave vysokej impedancie a je ich možné využiť ako zbernicu pre komunikáciu mikrokontroléra aj s iným zariadením.



Obr. 6.1: Komunikácia medzi mikrokontrolérom a USB prevodníkom.

b) Napájanie riadiacej jednotky

Pre napájanie riadiacej jednotky je použitý USB port počítača. Ten poskytuje napájacie napätie +5 V a štandardne dokáže dodávať pripojenému zariadeniu prúd až do 500 mA, čo pre činnosť prevodníka úplne postačuje. Napájanie cez USB port je možné zvoliť nastavením jumpera (skratovacej prepojky) na konektore JP2 do polohy 2-3. Riadiacu jednotku je možné napájať aj pomocou externého zdroja. Ten sa v prípade potreby pripája na piny 1 (–) a 2 (+) konektora JP2. Napájanie mikrokontroléra je v oboch prípadoch riadené cez tranzistor IRF7210 integrovaným obvodom FT245BL. Ak tento obvod nekomunikuje s operačným systémom (nie je spustený operačný systém, nie sú nainštalované ovládače a iné), ponecháva napájanie mikrokontroléra vypnuté. Mikrokontrolér pre správnu činnosť vyžaduje napájacie napätie +3,3 V s toleranciou ±10 %. To zabezpečuje použitý napäťový stabilizátor LF33CDT. V prípade potreby dokáže tento stabilizátor prostredníctvom pinov 1 (–) a 2 (+) na konektoroch JP5 až JP7 napájať aj externe obvody. Maximálna prúdová zaťažiteľnosť stabilizátora je 1 A, avšak je potrebné brať ohľad na maximálnu prúdovú zaťažiteľnosť hlavného napájacieho zdroja, čo je v prípade napájania cez USB port spomínaných 500 mA. Konkrétne rozmiestnenie jednotlivých konektorov na riadiacej jednotke je znázornené na obr. 6.2.



Obr. 6.2: Rozmiestnenie konektorov na riadiacej jednotke.

c) Pripojenie analyzovanej zbernice

Riadiaca jednotka sa k neznámej zbernici pripája cez konektory JP3 a JP4. Podrobné zapojenie konektorov pre jednotlivé zbernice je uvedené v tab. 6.1. Mikrokontrolér je od analyzovanej zbernice oddelený prostredníctvom sady NPN tranzistorov. To umožňuje pripojenie riadiacej jednotky aj do neznámeho obvodu bez nutnosti vopred poznať alebo zisťovať napájacie a pracovné napätia obvodu a zároveň toto zapojenie spĺňa funkciu wired-AND. Na riadiacej jednotke je možné nastavením jumperov na konektoroch JP20 až JP24 prepojiť analyzovanú zbernicu cez tzv. pull-up rezistory k napätiu +5 V (prepojenie 2-3), alebo k napätiu +3,3 V (prepojenie 1-2), alebo je možné ponechať tranzistory na zbernici s otvoreným kolektorom (bez prepojenia jumperom). V prípade potreby je samozrejme možné obísť tranzistory a pripojiť zbernicu priamo k mikrokontroléru. Na priame pripojenie slúžia konektory JP11 až JP14. Ich presné zapojenie a význam je možné nájsť v schéme zapojenia, ktorá je uvedená v prílohách. Na porty RC0 až RC2 mikrokontroléra je možné pripájať signály s maximálnym napätím +3,3 V. Porty RC3 až RC9 mikrokontroléra sú schopné bez poškodenia pracovať aj so signálom s maximálnym napätím +5 V.

Tab. 6.1: Význam jednotlivých pinov konektorov JP3 a JP4.

Konektory	Zbernica			
	Neznáma	I²C	SPI	UART
JP4 3	D4	SCL	SCLK	DATA
JP4 2	D3	SDA	MISO	–
JP4 1	D2	–	MOSI	–
JP3 3	D1	–	SS0#	–
JP3 2	D0	–	SS1#	–
JP3 1	GND	GND	GND	GND

d) Indikácia stavu riadiacej jednotky

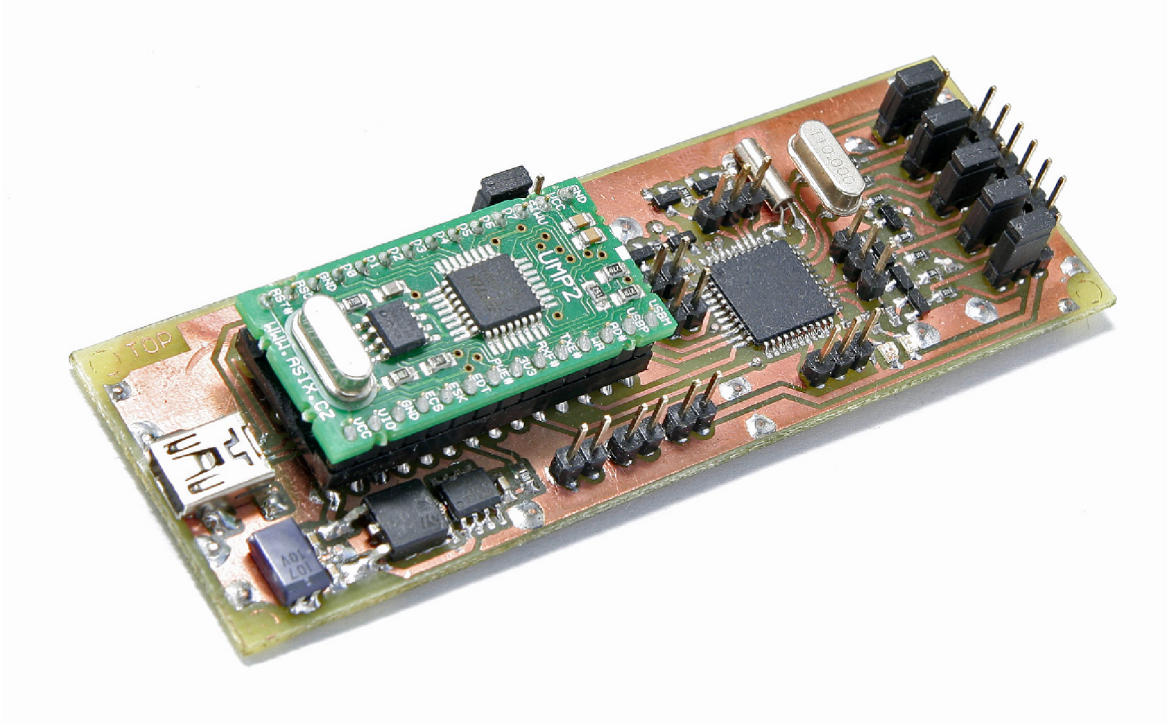
Na informovanie obsluhy o stave riadiacej jednotky slúžia dve LED diódy, ktoré sú umiestnené vpravo od konektora JP1 (viď. obr. 6.2). V prípade, že žiadna LED dióda nesvieti, obvod nie je napájaný, alebo modul UMP2 nekomunikuje s PC. Červená LED (bližšie ku konektoru) blikaním signalizuje úspešnú inicializáciu všetkých potrebných periférii a registrov mikrokontroléra a čakanie na spojenie s užívateľskou aplikáciou. Zelená LED (ďalej od konektora) blikaním signalizuje úspešné spojenie s užívateľskou aplikáciou a pokiaľ LED svieti bez blikania, prebieha analýza na niektorej zo zberníc alebo diagnostika riadiacej jednotky.

e) Aktualizácia firmware

Riadiaci program mikrokontroléra (firmware) je možné kedykoľvek v prípade potreby aktualizovať. Existujú dve metódy programovania vnútornej pamäte programu. Pri vývoji programu riadiacej jednotky bol mikrokontrolér programovaný metódou ICSP™ (In-Circuit Serial Programming) cez zariadenie MPLAB ICD2 od spoločnosti Microchip, ktoré sa pripája cez konektor JP1. Komunikačné linky vyvedené na tomto konektore sú zdieľané obidvoma stavovými linkami modulu UMP2, a preto je nutné v prípade programovania vytiahnuť tento modul z pätice. Z dôvodu tohto zdieľania nebolo možné pri vývoji odlaďovať program s použitím funkcie ICD (In-Circuit Debugger). Druhou metódou programovania, ktorú mikrokontrolér podporuje, je metóda RTSP (Run-Time Self-Programming). Tá umožňuje, aby riadiaci program aktualizoval sám seba. Je tak možné prostredníctvom komunikačnej linky zaslať mikrokontroléru novú verziu riadiaceho programu a ten sa aktualizuje sám bez pripájania ďalších zariadení.

f) Plošný spoj

Plošný spoj riadiacej jednotky bol navrhnutý v návrhovom systéme Eagle 5.4.0 a je realizovaný ako dvojvrstvový s použitím technológie SMT. Väčšina súčiastok je v prevedení SMD a výnimkou sú len konektory, pätica pre modul UMP2 a kryštály. Prevažná časť súčiastok je umiestnená na vrchnej strane plošného spoja (TOP) a len niekoľko nevyhnutných (kvôli minimálnej dĺžke prepoja) je na strane spodnej (BOTTOM). Umiestnenie súčiastok na plošnom spoji a vedenie spojov bolo realizované manuálne bez použitia akýchkoľvek automatických metód. Schéma zapojenia súčiastok riadiacej jednotky a masky a osadzovacie plány plošného spoja je možné nájsť v prílohách. Na obr. 6.3 je zobrazená zhotovená riadiaca jednotka.



Obr. 6.3: Riadiaca jednotka analyzátora.

6.2.2 Programové riešenie

Riadiaci program mikrokontroléra bol vyvíjaný v IDE (Integrated Development Environment) programu MPLAB verzie 8.30 od spoločnosti Microchip. Toto vývojové prostredie a použitý mikrokontrolér poskytujú vývojárovi niekoľko možností vývoja riadiaceho programu mikrokontroléra. Okrem podpory jazyka symbolických inštrukcií pre kompilátory MPASM a MPLAB ASM30, je podporovaný aj vyšší programovací jazyk – jazyk C, ktorý existuje ešte v dvoch variantoch. Pre maximálne efektívne využitie výkonu mikrokontroléra bol zvolený ako najvhodnejší práve jazyk pre MPLAB ASM30, ktorý

poskytuje vytvorenie strojového kódu z jazyka symbolických inštrukcií a je určený pre vývoj aplikácií na báze mikrokontrolérov dsPIC30F/33F DSC a PIC24X MCU [17].

a) Taktovanie mikrokontroléra

Väčšinu prevádzkového času pracuje mikrokontrolér v riadiacej jednotke na frekvencii 80 MHz, čo umožňuje beh programu s rýchlosťou až 40 MIPS. Táto frekvencia je odvodená z externého primárneho kryštálu s frekvenciou 10 MHz a integrovaného PLL obvodu, ktorý túto frekvenciu násobí ôsmimi. Mikrokontrolér je však naprogramovaný tak, aby bol po spustení taktovaný vnútorným FRC (Fast RC) oscilátorom s frekvenciou 7,37 MHz. To umožňuje rýchlejšie spustenie po pripojení napájania a poskytuje možnosť behu programu aj pri poruche primárneho externého kryštálu. Softvérovo je po spustení pri počiatočnom nastavovaní periférii mikrokontroléra spustený aj oscilátor s externe pripojeným primárnym kryštálom, je nastavený a aktivovaný vnútorný PLL obvod a až po ustálení generovaného taktovacieho signálu je mikrokontrolér prepnutý na tento zdroj signálu. V tejto procedúre je zahrnuté aj nastavenie a spustenie oscilátora s externe pripojeným sekundárnym kryštálom s frekvenciou 32,768 kHz. Ten môže byť použitý napríklad pre obvody dohľadu, RTCC (Real Time Calendar/Clock) a iné. V obvode riadiacej jednotky je tento oscilátor použitý ako zdroj signálu pre vnútorný časovač TIMER1, ktorý je nastavený tak, aby každú 1 sekundu generoval prerušenie. V obsluhe tohto prerušenia sú potom nastavované niektoré príznakové bity stavového registra (pretečenia subčasovačov, nastavenie čakacích stavov pri analýze a iné) a je tu riešená aj obsluha blikania diód LED na základe príznakových bitov stavového registra.

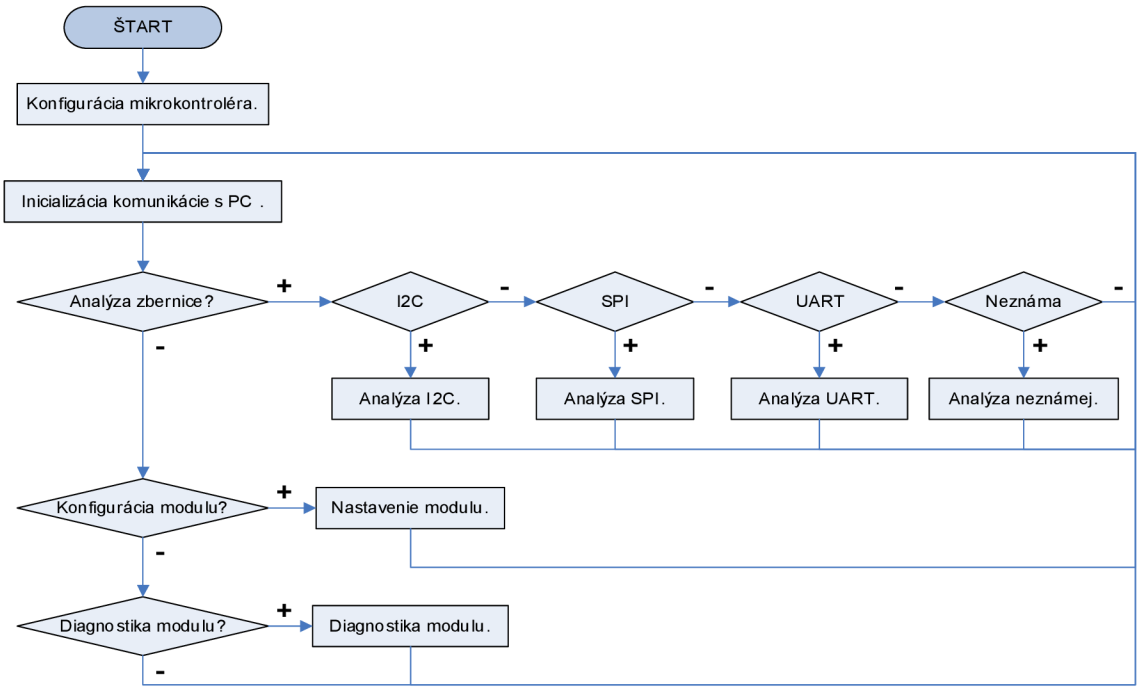
b) Port PMP

Pri úvodnom nastavovaní periférii mikrokontroléra je riadiacim programom nastavený a spustený aj port PMP, ktorý mikrokontrolér využíva na komunikáciu s modulom UMP2. Tento port podporuje spolu štyri hlavné módy a pri každom je možné nastaviť ešte širokú škálu možností. Najbližšie ku komunikačnému protokolu modulu UMP2 má mód *Master Mode 2*, ktorý pri komunikácii využíva niekoľko riadiacich, osem alebo šestnásť adresných a osem dátových liniek. Z riadiacich sú v tomto prípade použité len linky PMRD a PMWR a adresné linky sa nepoužívajú vôbec. Port PMP podporuje aj funkciu DMA, a tak pri komunikácii dokáže zapisovať priamo do RAM a čítať priamo z RAM bez akejkoľvek obsluhy riadiacim programom. Keďže však rozhranie nedisponuje žiadnou stavovou linkou potrebnou pre komunikáciu s modulom UMP2, nie je možné využiť túto funkciu a okrem toho je nutné stav modulu UMP2 sledovať softvérovo. Pred každým odosielaním alebo

prijímaním dát je preto zisťovaný stav prevodníka a až potom je zahájená komunikácia prostredníctvom portu PMP.

c) Indikácia stavu riadiacej jednotky

Riadiaci program po nastavení všetkých periférii zostáva čakať v hlavnej časti programu na inicializáciu komunikácie s užívateľskou aplikáciou. Tento stav je signalizovaný obsluhe blikaním červenej LED diódy na riadiacej jednotke. Ak prebehne inicializácia komunikácie v poriadku, obsluha je o tom informovaná zhasnutím červenej a blikaním zelenej LED diódy. V tomto okamihu v pravidelných intervaloch dochádza k prenosu dát medzi počítačom a riadiacou jednotkou. Ak po určitú dobu užívateľská aplikácia nezašle riadiacej jednotke požadované dáta (inicializácia), prechádza do stavu čakania na inicializáciu. Tento stav je obsluhe opäť signalizovaný blikaním červenej LED diódy. V prípade prebiehajúcej analýzy alebo diagnostiky zostáva zelená LED dióda svietiť nepretržite. Stručný vývojový diagram riadiaceho programu je uvedený na obr. 6.4 a podrobný vývojový diagram je možné nájsť v prílohách.

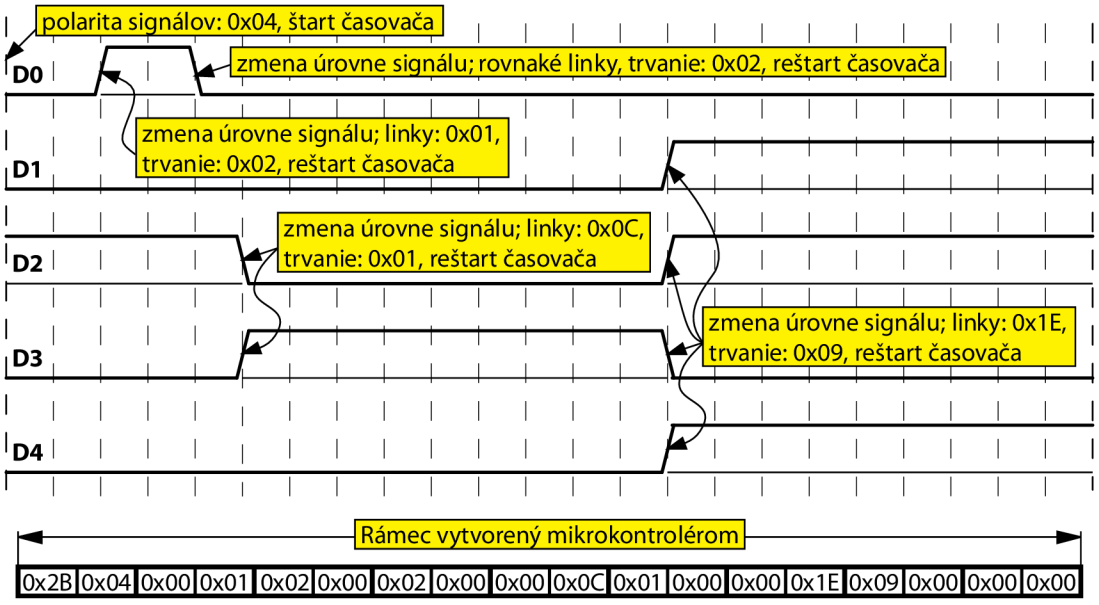


Obr. 6.4: Stručný vývojový diagram programu riadiacej jednotky analyzátora.

d) Analýza neznámej zbernice

Pri analýze neznámej zbernice sa využíva časovač TIMER2, ktorého vstup je pripojený na vnútorný taktovací signál mikrokontroléra. Analyzovaný môže byť priebeh signálu na niektorej z liniek D0 až D4, alebo na všetkých linkách súčasne. Ich presné

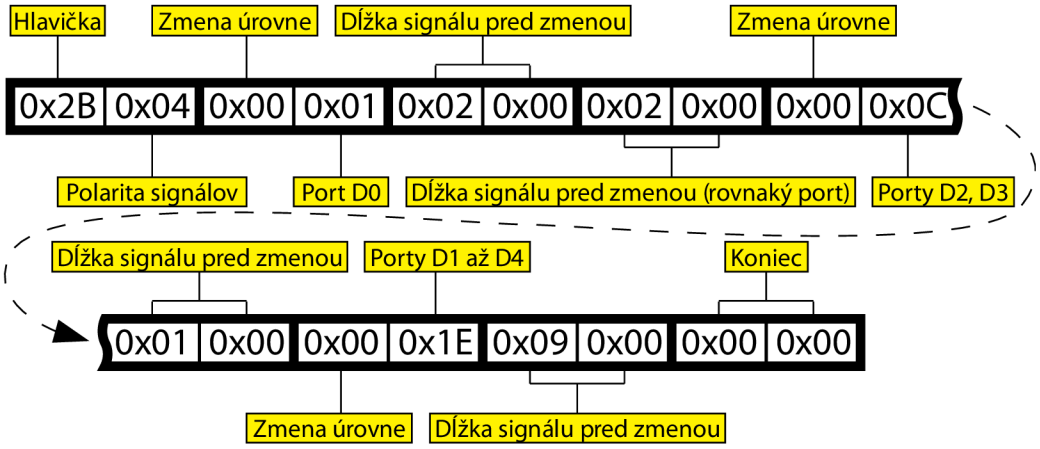
zapojenie je znázornené v tab. 6.1. Pred zahájením analýzy odošle užívateľská aplikácia riadiacej jednotke okrem iného aj 16-bitový parameter, ktorým sa nastaví perióda tohto časovača. Týmto spôsobom je možné v rámci možností nastaviť akúkoľvek vzorkovaciu frekvenciu analýzy. Obsluha prerušenia potom tvorí hlavný program analýzy neznámej zbernice. Najdlhšie trvanie tejto obsluhy, a teda celého podprogramu pre analýzu neznámej zbernice, je približne 40 inštrukčných cyklov. Túto hodnotu možno pokladať za hraničné minimum pri nastavovaní periódy časovača a odpovedá vzorkovacej frekvencii 1 MHz. Maximálna hodnota periódy časovača, ktorú možno nastaviť, je 65.535 a odpovedá vzorkovacej frekvencii približne 610 Hz. Mikrokontrolér pri vzorkovaní zbernice vykonáva už aj jej čiastočnú analýzu. Dáta, ktoré pripraví na odoslanie, nepredstavujú konkrétny stav liniek, ale zmenu oproti predchádzajúcemu stavu a jeho trvanie. Pri pomalých zmenách signálu na analyzovaných linkách je tak prenos dát od riadiacej jednotky k PC len minimálny. Na obr. 6.5 je znázornený príklad priebehu signálov na analyzovanej neznámej zbernici a v spodnej časti je výsledný dátový rámec vytvorený mikrokontrolérom, ktorý je odoslaný užívateľskej aplikácii.



Obr. 6.5: Priebeh analýzy neznámej zbernice a vytvorený dátový rámec.

Význam jednotlivých bajtov rámca vytvoreného v predchádzajúcom príklade je uvedený na obr. 6.6. Z dôvodu 16-bitovej architektúry mikrokontroléra a kvôli jednoduchšiemu signalizovaniu významu posielaných dát je každý vytvorený rámec zarovnaný na 16 bitov (2 bajty). Na začiatok každého rámca je vložená hlavička `0x2B` (1 bajt), ktorá je nasledovaná počiatočnou polaritou signálu (1 bajt). Každá linka predstavuje

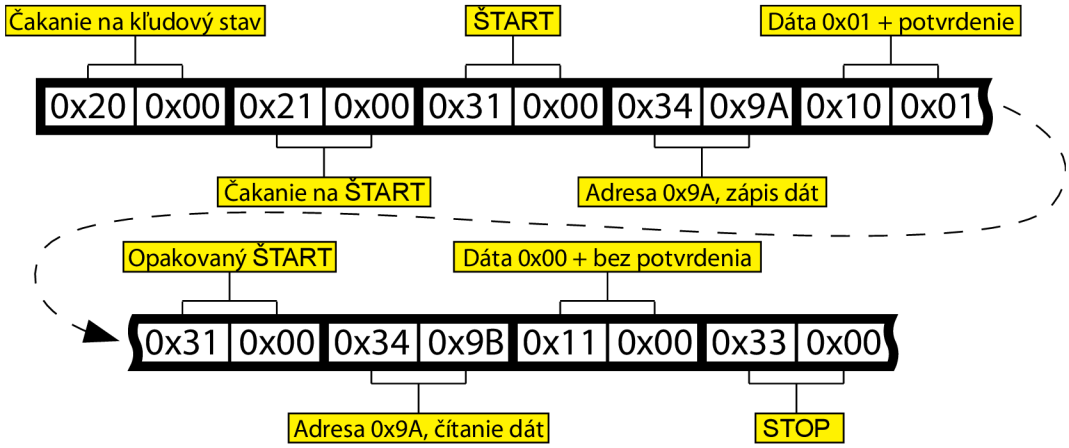
v tomto údaji 1 bit. To znamená, že kladná polarita na linke D0 odpovedá hodnote `0x01`, na linke D1 hodnote `0x02`, na linke D2 hodnote `0x04`, na linke D3 hodnote `0x08` a na linke D4 hodnote `0x10`. V prípade kombinácie kladnej polarity na viacerých linkách súčasne stačí hodnoty odpovedajúce jednotlivým linkám sčítať. V ďalšom 1 bajte sa očakáva hodnota `0x00`, ktorá signalizuje zmenu stavu na niektorej z liniek, alebo hodnota `0xFF`, ktorej význam je vysvetlený nižšie. Ďalší 1 bajt identifikuje, na ktorej linke v prípade zmeny k zmene došlo. Opäť platí, že každý 1 bit odpovedá jednej linke. Nasledujúce 2 bajty predstavujú dĺžku signálu od poslednej zmeny, pričom nižší 1 bajt je prenášaný ako prvý. V ďalšom kroku môže dôjsť k zmene na rovnakých linkách ako v predchádzajúcom prípade a nasledujúce 2 bajty budú predstavovať opäť dĺžku signálu od poslednej zmeny. Môže však dôjsť k zmene na iných linkách, čo bude indikované riadiacou hodnotou `0x00` (1 bajt). Ďalší 1 bajt identifikuje linky, na ktorých zmena nastala. Aby bolo možné odlíšiť riadiaci bajt `0x00` od dĺžky, býva dĺžka v prípade, že je spodný 1 bajt nulový, zvýšená vždy o jedna. Tento prípad môže nastať len ak doba trvania úrovne prekročí hodnotu `0x00FF`. Vtedy je namiesto hodnoty napríklad `0x0100` zaslaná hodnota `0x0101`. V najhoršom prípade tak vzniká chyba približne 0,39 %. Ak sa signál nemení dlhšiu dobu, dôjde k pretečeniu vnútorného časovača a do rámca je vložená hodnota `0xFFFF` (2 bajty). Doba, za ktorú dôjde k pretečeniu, závisí od nastavenia periódy časovača. Rámec ukončuje hodnota `0x0000` (2 bajty). Táto hodnota je s indikáciou zmeny úrovne `0x00` nezameniteľná, keďže tá indikuje zmenu úrovne, na niektorej z liniek musí dôjsť k zmene, a preto je druhý bajt v takomto prípade vždy nenulový.



Obr. 6.6: Príklad dátového rámca pri analýze neznámej zbernice.

e) Analýza zbernice I²C

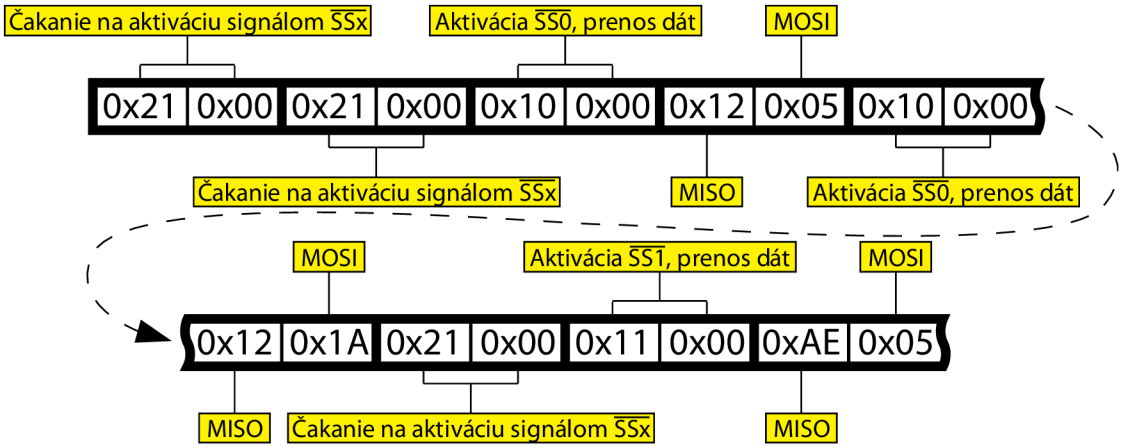
Pri analýze zbernice I²C je celý výkon mikrokontroléra sústredený na testovanie stavu liniek SCL a SDA. Zapojenie týchto liniek je znázornené v tab. 6.1. Pred začiatkom analýzy odošle užívateľská aplikácia riadiacej jednotke adresu pre filtrovanie prevádzky (1 bajt) a konfiguráciu (1 bajt). Konfigurácia obsahuje nastavenie polarity liniek a nastavenie potvrdzovania prijatých dát riadiacou jednotkou. Program pri analýze testuje stavy liniek postupne krok po kroku, pričom očakáva od analyzovanej zbernice dodržiavanie štandardu I²C. V prípade akéhokoľvek rozdielu v komunikácii je tento stav označený za chybu. Na začiatku analýzy sa očakáva kľudový stav na oboch linkách, ktorý je hlásený užívateľskej aplikácii prostredníctvom TIMER1 každú 1 sekundu návratovým kódom `0x0020` (2 bajty). Za kľudový stav sa považuje úroveň H na oboch linkách (pri neinvertujúcom nastavení liniek). Ak tento stav nastane, v ďalšom kroku sa bude očakávať ŠTART. Aj tento stav je hlásený užívateľskej aplikácii každú 1 sekundu, avšak návratový kód je v tomto prípade `0x0021` (2 bajty). Chyby sú hlásené každú 1 sekundu návratovým kódom `0xYY28` (2 bajty), kde `YY` je počet chýb od posledného hlásenia chyby. Zistený korektný ŠTART alebo korektný opakovaný ŠTART je užívateľskej aplikácii hlásený návratovým kódom `0x0031` (2 bajty). Po ňom je očakávaná 7-bitová adresa s ôsmym bitom $R/\overline{W}$, ktorý určí nasledujúci smer toku dát. Adresa a bit $R/\overline{W}$ sú užívateľskej aplikácii odoslané s návratovým kódom `0xYY34` (2 bajty), pričom `YY` predstavuje práve spomínanú adresu. Dáta získané zo zbernice I²C sú posielané užívateľskej aplikácii spolu s návratovým kódom `0xYY10` (2 bajty) v prípade zistenia potvrdených dát, alebo `0xYY11` (2 bajty) v prípade dát bez potvrdenia, kde `YY` sú samotné dáta. Zistený STOP je užívateľskej aplikácii hlásený návratovým kódom `0x0033` (2 bajty) a koniec analýzy I²C návratovým kódom `0x0000` (2 bajty). Príklad dátového rámca vytvoreného pri analýze zbernice I²C je na obr. 6.7.



Obr. 6.7: Príklad dátového rámca pri analýze zbernice I²C.

f) Analýza zbernice SPI

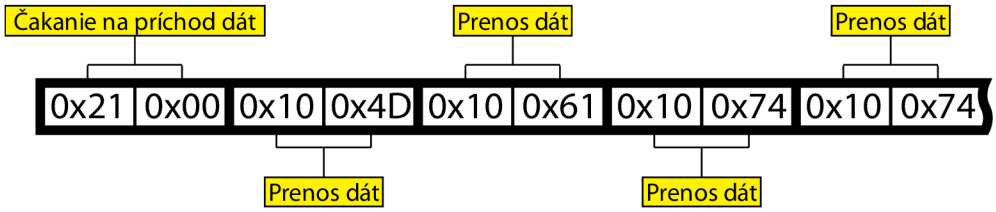
Rovnako ako pri analýze zbernice I²C, aj pri analýze zbernice SPI je celý výkon mikrokontroléra sústredený na testovanie stavu liniek. V tomto prípade ide o linky SCLK, MISO, MOSI, $\overline{SS0}$ a $\overline{SS1}$. Ich zapojenie je naznačené v tab. 6.1. Pred začiatkom analýzy odošle užívateľská aplikácia riadiacej jednotke dva konfiguračné bajty. Najnižšie 2 bity nižšieho 1 bajtu nastavia polaritu CPOL signálu SCLK a fázu CPHA. Ostatných 6 bitov a horný 1 bajt sa posiela vždy nulový. Na začiatku je užívateľskej aplikácii odoslaný návratový kód `0x0021` (2 bajty), ktorý signalizuje kľudový stav a čakanie na aktiváciu prenosu dát niektorým zo signálov $\overline{SSx}$. Tento návratový kód je počas úrovne H súčasne na oboch linkách $\overline{SSx}$ posielaný užívateľskej aplikácii každú 1 sekundu. Prenos dát začína až prechodom na úroveň L na niektorej z týchto liniek. Pri prenose dát je posielaný najprv návratový kód `0x0010` (2 bajty) pri aktivácii signálom $\overline{SS0}$, alebo `0x0011` (2 bajty) pri aktivácii signálom $\overline{SS1}$ a až potom sa posielajú samotné dáta `0xYYZZ` (2 bajty), pričom `YY` predstavuje dáta získané z linky MOSI a `ZZ` dáta z linky MISO. Pri jednej aktivácii signálom $\overline{SSx}$ môže byť prenesených aj niekoľko dvojíc bajtov za sebou. Vždy je však ako prvý odosielaný návratový kód `0x0010` (2 bajty) alebo `0x0011` (2 bajty). V prípade zaplnenia vyhradenej vnútornej vyrovnávacej pamäte mikrokontroléra je odoslaný užívateľskej aplikácii návratový kód `0x0000` (2 bajty) a analýza sa ukončí. Príklad dátového rámca vytvoreného pri analýze zbernice SPI je uvedený na obr. 6.8.



Obr. 6.8: Príklad dátového rámca pri analýze zbernice SPI.

g) Analýza rozhrania UART

Na analýzu signálu na rozhraní UART sa využíva prerušenie od časovača TIMER3. Ten má rovnako ako TIMER2 pripojený vstup na zdroj vnútorného taktovacieho signálu. Pripojenie riadiacej jednotky na neznámu linku UART je znázornené v tab. 6.1. Pred zahájením analýzy posiela užívateľská aplikácia riadiacej jednotke dva konfiguračné bajty. V jednom z nich sú nastavenia pre periódu časovača na požadovanú rýchlosť analyzovaného UART rozhrania. Podporovaných je šesť rýchlostí prenosu dát: 1.200 Bd, 2.400 Bd, 4.800 Bd, 9.600 Bd, 38.400 Bd a 115.200 Bd. Analyzované rozhranie je vzorkované dvojnásobnou rýchlosťou k rýchlosti prenosu dát. Po spustení analýzy je užívateľskej aplikácii odosielaný každú 1 sekundu návratový kód `0x0021` (2 bajty), ktorý signalizuje čakanie na príchod dát. Dáta sú prenášané spolu s návratovým kódom `0xYY10` (2 bajty), kde `YY` sú samotné dáta. V prípade zaplnenia vyhradenej vyrovnávacej pamäte mikrokontroléra je zaslaný užívateľskej aplikácii návratový kód `0x0000` (2 bajty) a analýza sa ukončí. Príklad dátového rámca vytvoreného pri analýze rozhrania UART je uvedený na obr. 6.9.



Obr. 6.9: Príklad dátového rámca pri analýze rozhrania UART.

h) Hlavný program a dohľad

Priebeh každej analýzy je signalizovaný obsluhe neprerušovaným svietením zelenej LED diódy. Podprogram každej analýzy je vždy kontrolovaný dohľadovým časovačom TIMER1. Je to dôležité najmä v prípade, keď sa očakáva príchod dát od užívateľskej aplikácie a dôjde napríklad k výpadku vzájomnej komunikácie riadiacej jednotky s PC. Keďže snímanie stavu stavovej linky $\overline{\text{RXF}}$ modulu UMP2 vo väčšine prípadov zablokuje činnosť mikrokontroléra, je potrebné po určitom čase pri neúspešnej komunikácii prerušiť vykonávanú časť programu, vrátiť riadenie na začiatok programu a signalizovať vzniknutý stav obsluhe prostredníctvom červenej LED diódy.

Každú analýzu môže užívateľská aplikácia kedykoľvek prerušiť zaslaním riadiaceho príkazu `0x585c` (2 bajty) riadiacej jednotke. Podprogram analýzy v takomto prípade ukončí odosielanie akýchkoľvek dát, zastaví čítače spustené analýzou a ukončí sa. Mikrokontrolér sa tak vráti k vykonávaniu časti programu úvodnej inicializácie spojenia. Obsluhe je tento stav hlásený blikaním zelenej LED diódy.

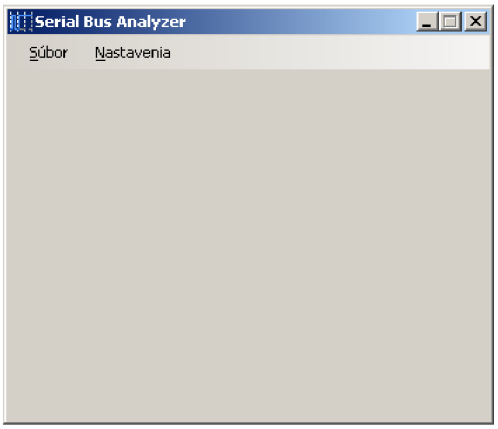
Na linkách D0 až D4 riadiacej jednotky je možné v prípade potreby nastaviť aj vlastný signál a taktiež je možné čítať aj aktuálny stav liniek. Táto funkcia je prístupná v diagnostickom móde.

6.3 Užívateľská aplikácia

Užívateľská aplikácia bola vyvíjaná vo vývojovom prostredí Microsoft Visual Studio 2008 a je naprogramovaná v jazyku C#. Bola testovaná a odladená na operačnom systéme Microsoft Windows XP s SP2, avšak nemal by byť problém s jej fungovaním aj na novších verziách Windows. Pre správne fungovanie aplikácie je nutná inštalácia balíka Microsoft .NET Framework verzia 3.0 a viac. Aby mohla aplikácia komunikovať s riadiacou jednotkou, je nutné nainštalovať správne ovládače pre použitý integrovaný obvod FT245BL. Ide konkrétne o ovládače D2XX, ktoré umožňujú priamu komunikáciu s týmto obvodom prostredníctvom knižnice DLL, ktorá je používaná pre komunikáciu aplikácie s riadiacou

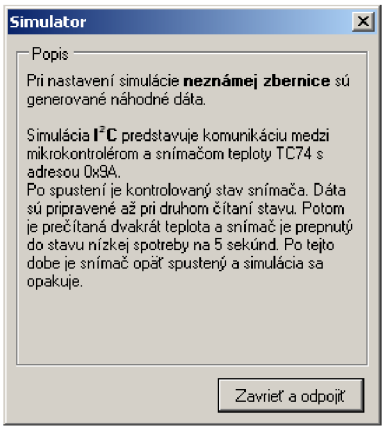
jednotkou. Balík .NET a ovládače je možné nájsť na priloženom CD. Ostatné potrebné knižnice sú umiestnené priamo v priečinku hlavného programu, takže by mal program fungovať bez akýchkoľvek problémov.

Užívateľská aplikácia sa spúšťa súborom „SerialBusAnalyzer.exe“ s ikonou . Hlavné okno aplikácie je zobrazené na obr. 6.10. Pred začiatkom analýzy, ktorá sa volí cez menu Súbor > Nová > Analýza, je potrebné cez menu Nastavenia > Modul zvoliť a následne pripojiť zariadenie, ktoré sa použije na analýzu. Ak užívateľ nemá k dispozícii hardvérový modul xBusAnalyzer (riadiacu jednotku), môže zvoliť Simulácia a otestovať tak aspoň niektoré z funkcií, ktoré aplikácia ponúka. V aktuálnej verzii je v simulácii podporovaná len analýza neznámej a I²C zbernice. S pripojeným hardvérovým modulom je možné analyzovať okrem týchto dvoch aj zbernicu SPI a rozhranie UART.



Obr. 6.10: Hlavné okno užívateľskej aplikácie.

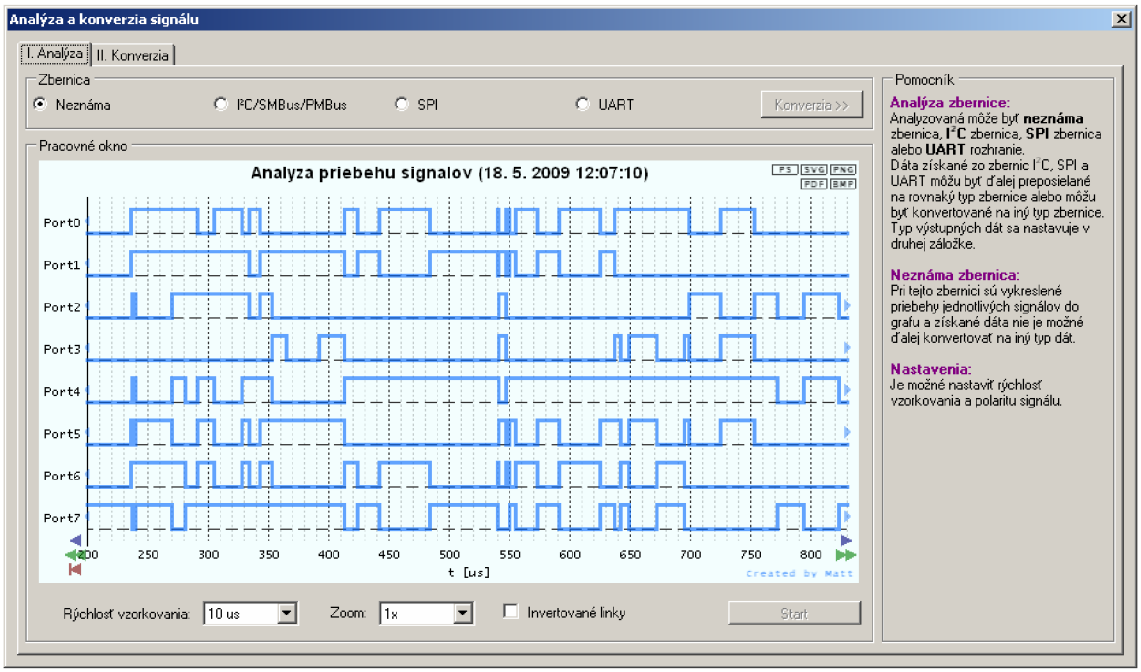
Pri voľbe z menu Nastavenia > Modul > Pripojiť s nastaveným zariadením Simulácia sa zobrazí užívateľovi okno, ako je zobrazené na obr. 6.11, ktoré simuluje pripojené zariadenie. Toto okno obsahuje len tlačidlo Zavrieť a odpojiť, ktoré slúži na ukončenie virtuálneho generovania signálu. Pokiaľ je toto okno otvorené, zariadenie zostáva virtuálne pripojené a je preto nutné ponechať ho otvorené aj počas celej analýzy. Simulátor umožňuje vykonať maximálne jednu analýzu na jedno pripojenie. V prípade, že je požadovaná ďalšia analýza, je nutné zavrieť (odpojiť) okno simulátora a opätovne ho pripojiť.



Obr. 6.11: Okno simulátora.

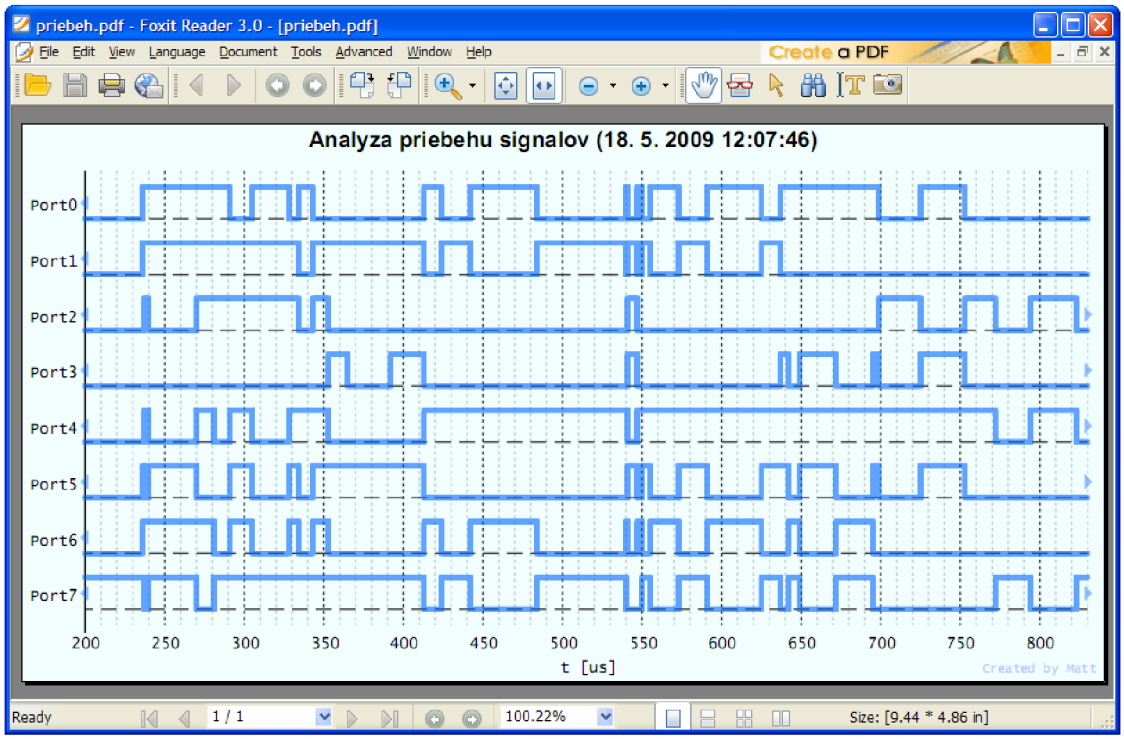
Okno analýzy je možné vyvolať cez menu Súbor > Nová > Analýza. V novootvorenom okne má užívateľ na výber medzi záložkami I. Analýza a II. Konverzia. Ako už z názvov vyplýva, v prvej záložke sa vykonáva analýza zberníc, kde je možné zvoliť niektorú zo štyroch možností analýzy.

Pri analýze neznámej zbernice môže užívateľ nastaviť rýchlosť vzorkovania signálu a polaritu signálov. Analýza sa spúšťa tlačidlom Start a môže byť ukončená manuálne rovnakým tlačidlom s popisom Stop, alebo automaticky po naplnení vyrovnávacej pamäte v riadiacej jednotke. Po ukončení analýzy si môže užívateľ pomocou funkcie Zoom zobraziť väčšie detaily získaných priebehov v časovej oblasti a pomocou šípok umiestených vpravo a vľavo pod grafom sa môže pohybovať v grafe pozdĺž celej časovej osi. Príklad priebehu signálov získaného pri analýze neznámej zbernice je zobrazený na obr. 6.12. Tento priebeh je možné v prípade potreby exportovať do niekoľkých výstupných formátov kliknutím na požadovaný typ v pravej hornej časti grafu. Podporované sú formáty PS, SVG, PDF, PNG a BMP. Vyexportovaný súbor je umiestnený do hlavného priečinka aplikácie a je pomenovaný „priebeh.xxx“, kde xxx je prípona súboru podľa zvoleného typu. Dáta získané touto analýzou nie je možné ďalej konvertovať na iný typ zbernice.



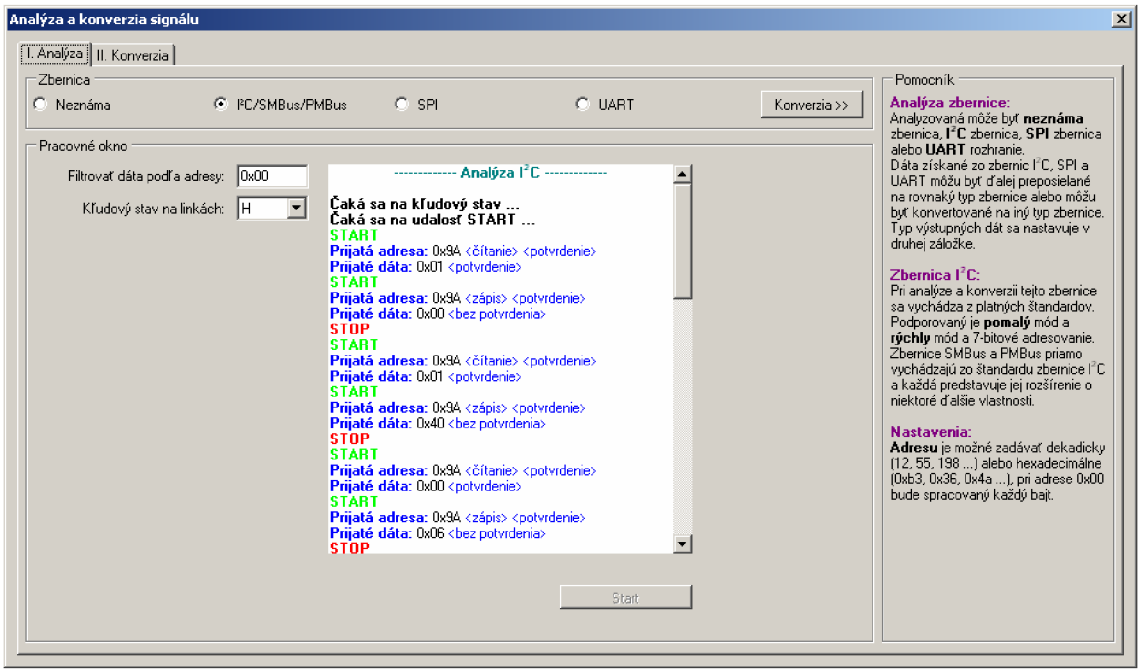
Obr. 6.12: Okno s výslednými priebehmi po analýze neznámej zbernice.

Na obr. 6.13 je zachytený dokument „priebeh.pdf“ vytvorený užívateľskou aplikáciou, ktorý je otvorený v programe Foxit Reader. Obrázok je v tomto dokumente vytvorený vo vektorovej grafike a jeho jednotlivé prvky tvoria objekty. Je teda možné ďalej upravovať rozmery obrázka, dopĺňať ho o nové objekty a odstraňovať aktuálne objekty jednoducho a bez akejkoľvek straty kvality. Rovnaké výhody vektorovej grafiky ponúkajú aj formáty PS a SVG.



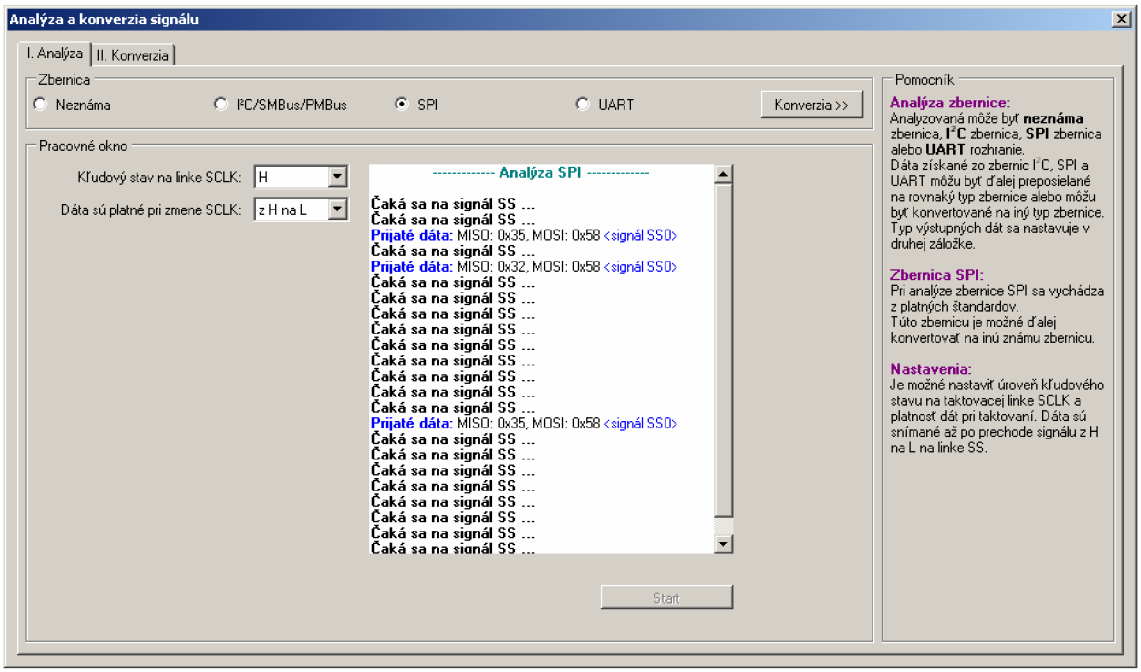
Obr. 6.13: Graf vyexportovaný do formátu PDF zobrazený vo Foxit Reader.

Pri analýze zbernice I²C/SMBus/PMBus môže užívateľ nastaviť filtrovanie prenášaných dát podľa adresy, ktorá sa zadáva v dekadickej forme priamo číslom alebo vo forme hexadecimálnej s prefixom `0x` pred hexadecimálnym číslom. Pri voľbe adresy `0x00` nie sú filtrované žiadne dáta. Zadanie zlej hodnoty do tohto poľa je signalizované červeným podfarbením poľa. Užívateľ môže tiež zvoliť úroveň kľudového stavu (polaritu) na linkách SDA a SCL. Po nastavení týchto parametrov je možné analýzu spustiť rovnako ako v predchádzajúcom prípade tlačidlom Start. Analýza sa ukončuje automaticky pri naplnení vyrovnávacej pamäte v riadiacej jednotke, alebo ju užívateľ môže zastaviť rovnakým tlačidlom Stop. Príklad takejto analýzy na zbernici I²C je zobrazený na obr. 6.14. Stavy zistené na analyzovanej zbernici sú vpisované do textového poľa v strednej časti okna. Toto pole je voľne meniteľné, takže je možné doňho vpisovať vlastné poznámky, alebo odstraňovať niektoré získané stavy. Výsledný text je možné preniesť do inej aplikácie označením požadovanej časti a jej skopírovaním do schránky (CTRL+C). V cieľovej aplikácii, kde majú byť získané dáta použité, je potom možné text vložiť ako formátovaný (so zachovaním farieb a štýlov písma) alebo neformátovaný (obyčajný text).



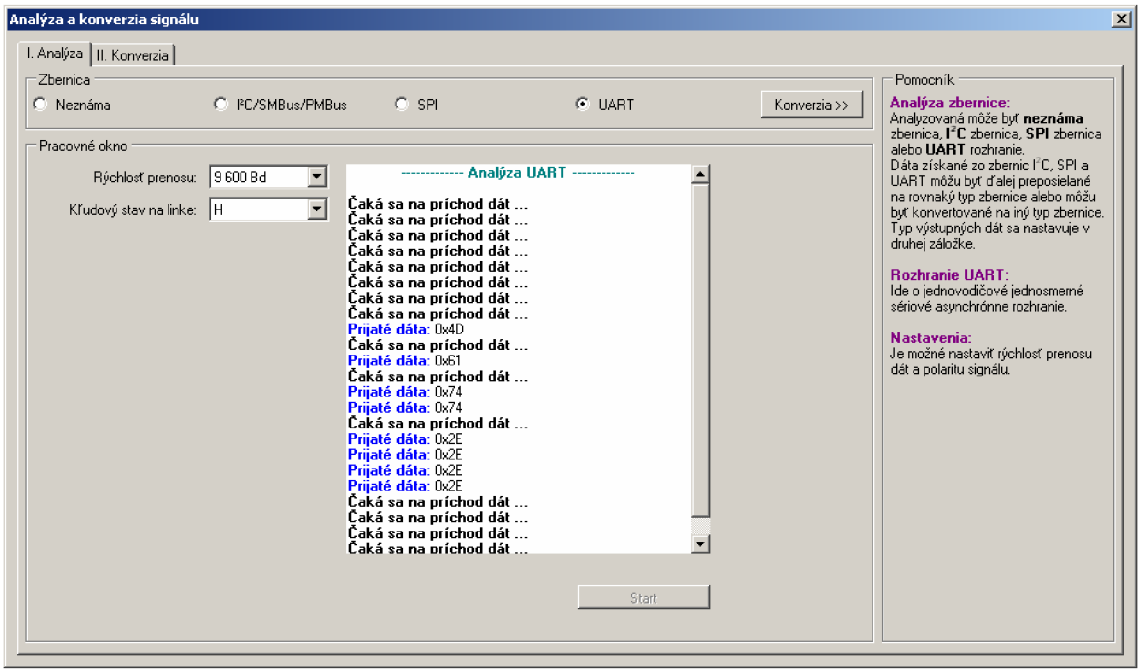
Obr. 6.14: Okno so zachytenými stavmi pri analýze zbernice I²C.

Treťou možnosťou analýzy je analýza zbernice SPI. Pri tejto zbernici je pred samotnou analýzou možné nastaviť kľudový stav na linke SCLK (polaritu) a stav SCLK, pri ktorom budú dáta snímané z dátových liniek (fázu). Analýza sa spúšťa tlačidlom Start a ukončuje sa automaticky pri naplnení vnútornej vyrovnávacej pamäte v riadiacej jednotke, alebo ju užívateľ môže zastaviť manuálne stlačením rovnakého tlačidla Stop. Príklad analýzy na zbernici SPI je znázornený na obr. 6.15. Získané dáta sú vkladané do textového poľa v strednej časti obrazovky, ktoré je aj v tomto prípade voľne meniteľné a umožňuje vpisovať vlastné poznámky, alebo odstraňovať zachytené stavy získané pri analýze zbernice SPI.



Obr. 6.15: Okno so zachytenými stavmi pri analýze zbernice SPI.

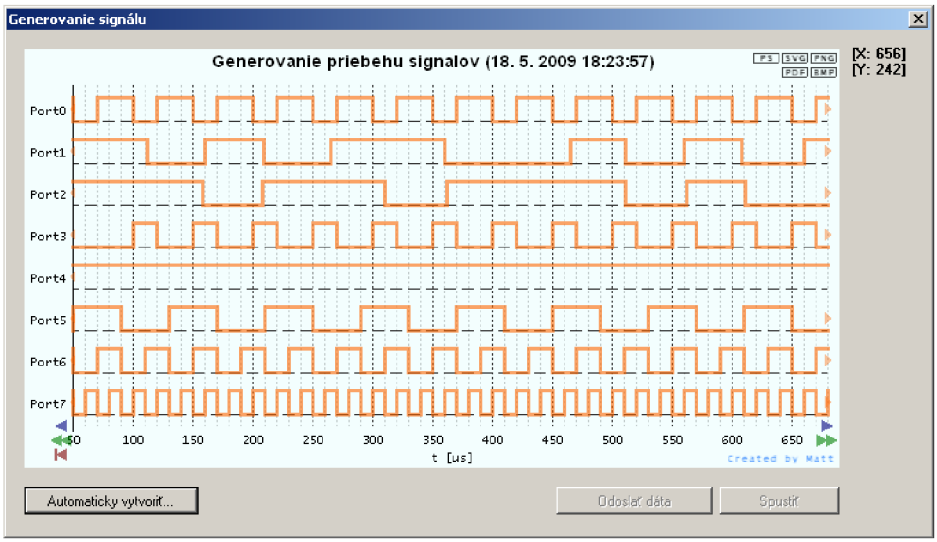
Poslednou možnou analýzou je analýza rozhrania UART. Aj v tomto prípade je možné pred samotnou analýzou nastaviť základné parametre. Užívateľ môže zvoliť rýchlosť prenosu dát na rozhraní a kľudový stav (polaritu) dátovej linky. Podporovaných je šesť rýchlostí prenosu dát. Sú to: 1.200 Bd, 2.400 Bd, 4.800 Bd, 9.600 Bd, 38.400 Bd a 115.200 Bd. Analýza sa spúšťa tlačidlom Start a rovnako, ako v predchádzajúcich prípadoch, môže byť ukončená automaticky pri zaplnení vnútornej vyrovnávacej pamäte v riadiacej jednotke, alebo ju môže zastaviť užívateľ stlačením rovnakého tlačidla Stop. Príklad analýzy rozhrania UART je znázornené na obr. 6.16. Získané dáta sú vpisované do textového poľa v strede okna, ktoré je taktiež meniteľné.



Obr. 6.16: Okno so zachytenými stavmi pri analýze rozhrania UART.

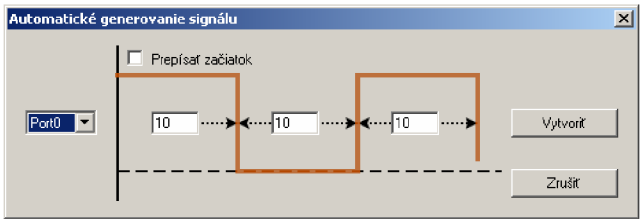
V záložke II. Konverzia je možné nastaviť prevod dát zo vstupnej zbernice I²C alebo SPI alebo vstupného rozhrania UART zvoleného v záložke I. Analýza na rovnaký alebo iný typ zbernice alebo rozhrania. Táto funkcia nie je v aktuálnej verzii aplikácie podporovaná a vytvorené formuláre a ovládacie prvky sú pripravené pre ďalšie rozširovanie aplikácie.

V aplikácii je možné vytvárať aj vlastný priebeh signálu. Cez menu Súbor > Nová > Generátor ponúkne aplikácia užívateľovi v novootvorenom okne (obr. 6.17) osem signálov v úrovni H pozdĺž celej časovej osi. Užívateľ si môže nastavovať zmenu signálu kliknutím na konkrétny signál v požadovanom časovom okamihu, čím v tomto mieste dôjde k zmene úrovne a tá bude trvať až po koniec generovaného signálu, prípadne po najbližšiu ďalšiu zmenu signálu. Kliknutím na os Y v mieste konkrétneho signálu je možné zmeniť polaritu celého tohto signálu. Pomocou šípok vľavo a vpravo pod grafom je s rôznou veľkosťou kroku umožnený užívateľovi pohyb pozdĺž celej časovej osi. Vygenerovaný signál je možné aj exportovať do niektorého zo zvolených formátov kliknutím na konkrétny typ vpravo v hornej časti grafu.



Obr. 6.17: Okno pre generovanie vlastného signálu.

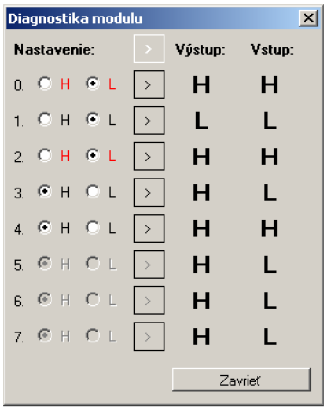
Vytváranie periodických signálov môže užívateľovi uľahčiť aj funkcia automatického generovania signálu (obr. 6.18), ktorá je prístupná kliknutím na tlačidlo Automaticky vytvoriť... Na obr. 6.17 je možné vidieť niekoľko takto vytvorených signálov. V okne pre automatické generovanie signálu užívateľ nastaví cieľový port, dĺžku úvodnej úrovne a dĺžky úrovní pre jednu periódu. V prípade, že je na zvolenom porte už vytvorený nejaký priebeh signálu, je možné zvoliť jeho prepísanie, alebo je možné pripojiť k nemu od nastaveného miesta nový aktuálne vytváraný priebeh signálu. Signál sa generuje s nastavenou periódou od nastaveného miesta pozdĺž celej časovej osi. Generovanie vlastného priebehu signálu nie je v aktuálnej verzii programu riadiacej jednotky podporované.



Obr. 6.18: Okno pre automatické generovanie periodického signálu.

Ak je ako zariadenie zvolený xBusAnalyzer (riadiaca jednotka), je možné po jeho pripojení vyvolať cez menu Nastavenie > Modul > Diagnostika okno pre testovanie jednotlivých liniek. Vzhľad okna je zachytený na obr. 6.19. Pre každú linku je vyhradená skupina ovládacích prvkov. Prepínač H/L umožňuje nastaviť požadovanú výstupnú úroveň linky a tlačidlo > slúži na aplikáciu tohto nastavenia. Aplikované nastavenie je zobrazené v stĺpci Výstup: a aktuálny stav linky v stĺpci Vstup:. Neaplikované nastavenie prepínača je

signalizované jeho červeným zafarbením. Najvrchnejšie tlačidlo > slúži na globálne aplikovanie zmenených nastavení jednotlivých liniek. V prípade, že nie je nastavená zmena na žiadnej linke, slúži toto tlačidlo na uzamknutie prenosu nastavenia, čo znamená, že nastavenia prepínačov sú aplikované okamžite po každej zmene. Tlačidlo Zavrieť slúži na zatvorenie okna.



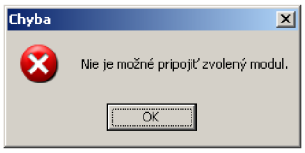
Obr. 6.19: Okno diagnostiky modulu.

Niektoré vytvorené ovládacie prvky v aplikácii sú síce viditeľné, avšak nie sú sprístupnené, pretože v aktuálnej verzii aplikácie nie sú vytvorené metódy na ich obsluhu, alebo tieto metódy nie sú dokončené. Patrí sem napríklad položka v menu Súbor > Uložiť, ktorá je určená pre ukladanie aktuálneho stavu nastavených parametrov, vytvorených a zachytených priebehov signálov a podobne. Položka v menu Súbor > Otvoriť slúži naopak na načítanie takto uloženého stavu aplikácie.

6.4 Problémy a možné riešenia

Pri testovaní funkčnosti celého analyzátora sa ako najväčší problém prejavila voľba hodnôt odporov rezistorov R7 až R16. Pri pripojených pull-up rezistoroch R7 až R11 na linky D0 až D4 dochádza pri úrovni H tokom prúdu cez tranzistory T7 až T11 kvôli príliš nízkej hodnote odporu rezistorov R12 až R16 k veľkému úbytku napätia na rezistoroch R7 až R11, čo spôsobuje, že celkové napätie na linkách D0 až D4 je pri úrovni H nízke (+3,24 V pri očakávaných +5 V a 2,22 V pri očakávaných +3,3 V). Nízka hodnota odporu rezistorov R12 až R16 spôsobuje pri analýze niektorých zberníc taktiež problém s rozpoznaním úrovne L, pretože spomínané tranzistory sa v niektorých prípadoch nezatvoria dostatočne rýchlo. Tento problém je možné vyriešiť nahradením rezistorov R12 až R16 rezistormi s odporom 120 kΩ až 150 kΩ.

V užívateľskej aplikácii boli všetky zistené kolizné stavy ošetrené výnimkami. V týchto prípadoch je užívateľ vždy upozornený chybovým hlásením s popisom možnej príčiny (obr. 6.20). Môže sa však stať, že užívateľ pri používaní aplikácie narazí na nový neošetrený kolizný stav. Vo väčšine prípadov problém vyrieši reštartovanie aplikácie a odpojenie a znovu pripojenie hardvérového modulu xBusAnalyzer (riadiacej jednotky).



Obr. 6.20: Príklad chybového hlásenia užívateľskej aplikácie.

7 ZÁVER

Prvá časť práce bola venovaná popisu špecifikácií a základných vlastností veľmi často používaných sériových zberníc I²C, SMBus, PMBus, SPI a rozhrania UART. V jednotlivých kapitolách sú detailne graficky a slovne popísané najdôležitejšie stavy, priebehy signálov a správanie obvodov pripojených na tieto zbernice.

V druhej časti bol navrhnutý a zostrojený prototyp riadiacej jednotky univerzálneho analyzátora, ktorý dokáže analyzovať prevádzku na neznámej zbernici, na známych zberniciach I²C, SMBus, PMBus a SPI a na rozhraní UART. Riadiaca jednotka, ktorej hlavnú časť tvorí mikrokontrolér PIC24HJ128GP204 od spoločnosti Microchip, sama vykonáva čiastočnú analýzu zvolenej zbernice a získané dáta potom zasiela prostredníctvom USB portu do počítača PC vytvorenej užívateľskej aplikácií, kde je analýza dokončená. Výsledok celej analýzy je potom graficky v podobe priebehov signálov a slovne prezentovaný užívateľovi. Aplikácia umožňuje užívateľovi uložiť získané údaje do niektorého z podporovaných grafických a textových formátov a využiť ich tak pre ďalšie účely.

V aktuálnej verzii užívateľskej aplikácie a riadiaceho programu mikrokontroléra nebola z časových dôvodov dokončená funkcia konverzie, ktorá by umožnila preklad toku dát z jednej zbernice na rovnaký alebo iný typ zbernice. Pre tento účel boli v užívateľskej aplikácii vytvorené formuláre a základné nastavovacie a ovládacie prvky, ktoré je možné využiť v budúcnosti pri dopĺňaní analyzátora o túto funkciu.

LITERATÚRA

[1] *Serial communication from Wikipedia, the free encyclopedia* [online]. [2005] [cit. 2008-11-29]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Serial_bus>

[2] Philips Semiconductors. *The I²C-bus Specification* [online]. Version 1.2. January 2000 [cit. 2008-11-07]. 46 s. Dostupný z WWW: <http://www.nxp.com/acrobat_download/literature/9398/39340011.pdf>

[3] *I²C from Wikipedia, the free encyclopedia* [online]. [2005] [cit. 2008-11-07]. Dostupný z WWW: <http://en.wikipedia.org/wiki/I2C>

[4] SBS Implementers Forum. *System Management Bus (SMBus) Specification* [online]. Version 2.0, August 3, 2000 [cit. 2008-12-01]. 59 s. Dostupný z WWW: <http://www.smbus.org/specs/smbus20.pdf>

[5] *System Management Bus from Wikipedia, the free encyclopedia* [online]. [2005] [cit. 2008-12-01]. Dostupný z WWW: <http://en.wikipedia.org/wiki/System_Management_Bus>

[6] Hawlett-Packard Corporation, Intel Corporation, Microsoft Corporation, Phoenix Technologies Ltd., Toshiba Corporation. *Advanced Configuration and Power Interface Specification* [online]. Revision 3.0b. October 10, 2006 [cit. 2008-12-05]. 631 s. Dostupný z WWW: <http://www.acpi.info/DOWNLOADS/ACPIspec30b.pdf>

[7] *PMBus from Wikipedia, the free encyclopedia* [online]. [2005] [cit. 2008-12-07]. Dostupný z WWW: <http://en.wikipedia.org/wiki/PMBus>

[8] *Serial Peripheral Interface (SPI) from Wikipedia, the free encyclopedia* [online]. [2005] [cit. 2008-12-05]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus>

[9] *Microwire from Wikipedia, the free encyclopedia* [online]. [2005] [cit. 2008-12-05]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Microwire>

[10] *Universal Asynchronous receiver/transmitter from Wikipedia, the free encyclopedia* [online]. [2005] [cit. 2008-12-07]. Dostupný z WWW: <http://en.wikipedia.org/wiki/UART>

[11] Microchip. *PIC24HJ32GP302/304, PIC24HJ64GPX02/X04, and PIC24HJ128GPX02/X04 Data Sheet* [online]. Revision B. March 2008 [cit. 2009-05-

10]. 338 s. Dostupný z WWW:
<http://ww1.microchip.com/downloads/en/DeviceDoc/70293B.pdf>

[12] Microchip. *PIC24H Family Reference Manual : Section 7. Oscillator* [online]. Revision D. December 2008 [cit. 2009-05-10]. 33 s. Dostupný z WWW: <http://ww1.microchip.com/downloads/en/DeviceDoc/70227d.pdf>

[13] Microchip. *PIC24H Family Reference Manual : Section 10. I/O Ports* [online]. Revision C. September 2008 [cit. 2009-05-10]. 12 s. Dostupný z WWW: <http://ww1.microchip.com/downloads/en/DeviceDoc/70230C.pdf>

[14] Microchip. *PIC24H Family Reference Manual : Section 11. Timers* [online]. Revision B. September 2008 [cit. 2009-05-10]. 24 s. Dostupný z WWW: <http://ww1.microchip.com/downloads/en/DeviceDoc/70244B.pdf>

[15] Microchip. *PIC24H Family Reference Manual : Section 35. Parallel Master Port (PMP)* [online]. Revision B. March 2008 [cit. 2009-05-10]. 31 s. Dostupný z WWW: <http://ww1.microchip.com/downloads/en/DeviceDoc/70302B.pdf>

[16] Microchip. *PIC24H Family Reference Manual : Section 39. Oscillator (Part III)* [online]. Revision B. September 2008 [cit. 2009-05-10]. 34 s. Dostupný z WWW: <http://ww1.microchip.com/downloads/en/DeviceDoc/70308B.pdf>

[17] Microchip. *MPLAB® ASM30, MPLAB® LINK30 AND UTILITIES USER'S GUIDE* [online]. Version 1.2, November 2002 [cit. 2009-05-10]. 272 s. Dostupný z WWW: <http://ww1.microchip.com/downloads/en/DeviceDoc/Asm30_Link_Util_51317e.pdf >

[18] FTDI Chip. *FT245BL USB FIFO (USB Parallel) I.C. Datasheet* [online]. Version 1.6. February 2005 [cit. 2009-05-10]. 24 s. Dostupný z WWW: <http://www.ftdichip.com/Documents/DataSheets/DS_FT245BL.pdf>

[19] ASIX. *UMP2 USB modul FIFO (paralelní) : Uživatelská příručka* [online]. [2000] [cit. 2009-05-10]. 14 s. Dostupný z WWW: <http://www.asix.cz/download/usb/ump2/ump2_cz.pdf>

ZOZNAM POUŽITÝCH SKRATIEK A ZNAČIEK

ACPI	Advanced Configuration and Power Interface; rozhranie pre rozšírenú správu napájania
ADC	Analog/Digital Convertor; Analógovo-Digitálny prevodník
ARA	Alert Response Address; špeciálna adresa na zbernici SMBus
ARP	Address Resolution Protocol; protokol používaný na prideľovanie dynamických adries na zbernici SMBus
Bd	Baud; jednotka modulačnej rýchlosti
BMP	BitMaP; nekomprimovaný grafický formát
C-BUS	Komunikačný protokol určený na kontrolu a riadenie
CRC	Cyclic Redundancy Check; cyklický zabezpečovací kód
DDA	Device Default Address; implicitná adresa zariadenia na zbernici SMBus
DLL	Dynamic-Link Library; dynamicky linkovaná knižnica
DMA	Direct Memory Access; priamy prístup do pamäte
EEPROM	Electrically Erasable Programmable Read-Only Memory; pamäť elektronicky vymazateľná a programovateľná
EIA	Electronic Industries Association; názov spoločnosti
FIFO	First-In, First-Out; vyrovnávacia pamäť typu prvý-dnu, prvý-von
FRC	Fast RC oscillator; rýchly RC oscilátor
I²C	Inter-IC Control; 2-vodičová sériová zbernica
ICD	In-Circuit Debugger; odlaďovací nástroj od spoločnosti MICROCHIP pre vývoj aplikácií na báze PIC
ICSP	In-Circuit Serial Programming; vnútorné sériové programovanie mikrokontrolérov PIC prostredníctvom dvoch sériových liniek
IDE	Integrated Development Environment; integrované vývojové prostredie
IrDA	Infrared Data Association; názov spoločnosti

JTAG	Joint Test Action Group; názov pre štandard rozhrania
LED	Light-Emitting Diode; elektronická súčiastka generujúca viditeľné svetlo
MISO	Master Input, Slave Output; vstupná sériová linka zbernice SPI
MOSI	Master Output, Slave Input; výstupná sériová linka zbernice SPI
NPN	Typ bipolárneho tranzistora
PC	Personal Computer; osobný počítač
PDF	Portable Document Format; dokumentový formát vytvorený spoločnosťou Adobe
PEC	Packet Error Checking; mechanizmus kontroly chýb paketov
PLL	Phase-Locked Loop; slučka fázového závesu
PMBus	Power Management Bus; 2-vodičová sériová zbernica
PMP	Parallel Master Port; paralelné rozhranie mikrokontroléra
PNG	Portable Network Graphics; grafický formát pre bezstratovú kompresiu
PS	Post Script; príkazový jazyk pre tlačiarne
RAM	Random Access Memory; pamäť s náhodným prístupom (časté označenie operačnej pamäti)
RTCC	Real Time Calendar/Clock; obvod reálneho času
RTSP	Run-Time Self-Programming; programovanie pamäte programu vlastným riadiacim programom mikrokontroléra
RxD	Received Data; sériová linka pre príjem dát na UART
SCL	Serial Clock; sériová taktovacia linka zbernice I^2C
SCLK	Serial ClocK; sériová taktovacia linka zbernice SPI
SDA	Serial Data; sériová dátová linka zbernice I^2C
SGPIO	Serial General Purpose Input/Output; 4-vodičová sériová zbernica
SMBCLK	SMBus ClocK; sériová taktovacia linka zbernice SMBus
SMBDAT	SMBus DATa; sériová dátová linka zbernice SMBus

SMBus	System Management Bus; 2-vodičová sériová zbernica
SMD	Surface Mounted Device; súčiastka pre povrchovú montáž
SMT	Surface Mounted Technology; technológia povrchovej montáže
SPI	Serial Peripheral Interface; 4-vodičová sériová zbernica
SS	Slave Select; linka pre výber zariadenia *slave* na zbernici SPI
SVG	Scalable Vector Graphics; jazyk určený na opis 2-D vektorovej grafiky
TTL	Transistor-Transistor Logic; tranzistorovo-tranzistorová logika
TxD	Transmitted Data; sériová linka pre odosielanie dát na UART
UART	Universal Asynchronous Receiver/Transmitter; univerzálna asynchrónna sériová linka
USB	Universal Serial Bus; univerzálna sériová zbernica

PRÍLOHY

Príloha A.1: Schéma zapojenia súčiastok riadiacej jednotky.

Príloha A.2: Masky plošného spoja riadiacej jednotky.

Príloha A.3: Osadzovacie plány riadiacej jednotky.

Príloha A.4: Zoznam použitých súčiastok.

Príloha B.1: Podrobný vývojový diagram programu riadiacej jednotky (strana 1 z 9).

Príloha B.2: Podrobný vývojový diagram programu riadiacej jednotky (strana 2 z 9).

Príloha B.3: Podrobný vývojový diagram programu riadiacej jednotky (strana 3 z 9).

Príloha B.4: Podrobný vývojový diagram programu riadiacej jednotky (strana 4 z 9).

Príloha B.5: Podrobný vývojový diagram programu riadiacej jednotky (strana 5 z 9).

Príloha B.6: Podrobný vývojový diagram programu riadiacej jednotky (strana 6 z 9).

Príloha B.7: Podrobný vývojový diagram programu riadiacej jednotky (strana 7 z 9).

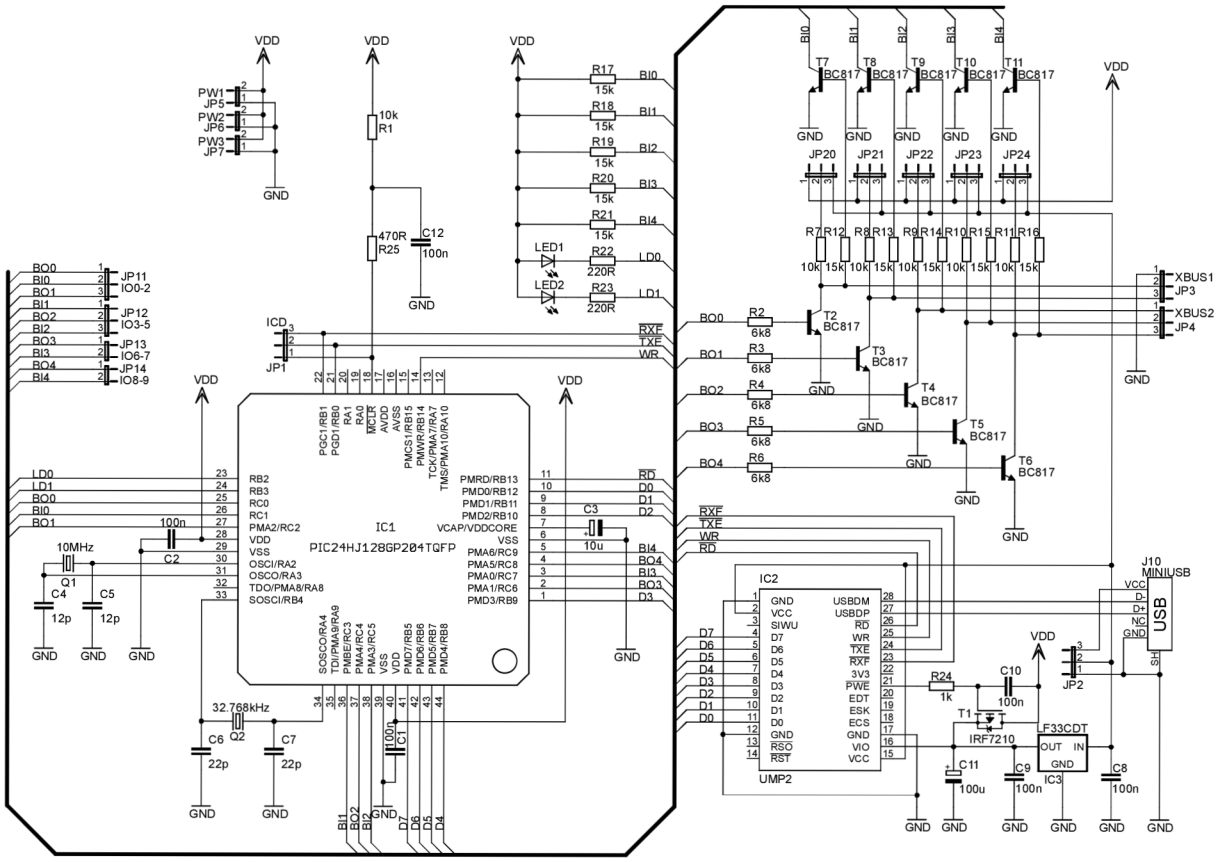
Príloha B.8: Podrobný vývojový diagram programu riadiacej jednotky (strana 8 z 9).

Príloha B.9: Podrobný vývojový diagram programu riadiacej jednotky (strana 9 z 9).

Príloha C: Stromová štruktúra priečinkov priloženého CD.

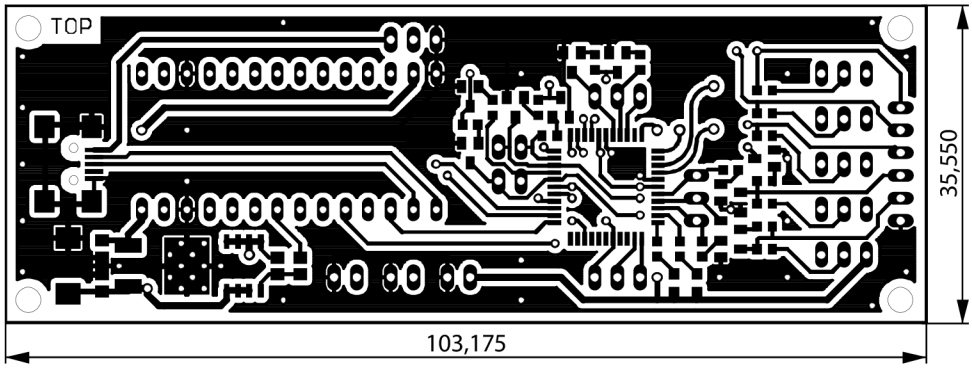
Príloha A.1

Schéma zapojenia súčiastok riadiacej jednotky.

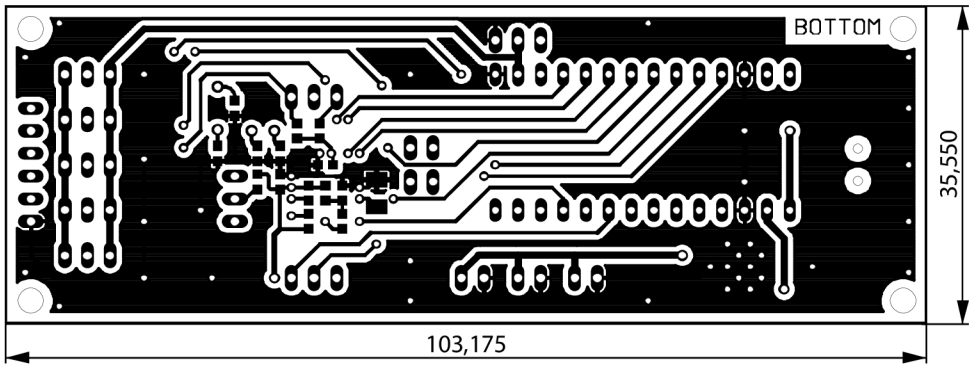


Príloha A.2

Vrchná strana masky plošného spoja analyzátora.

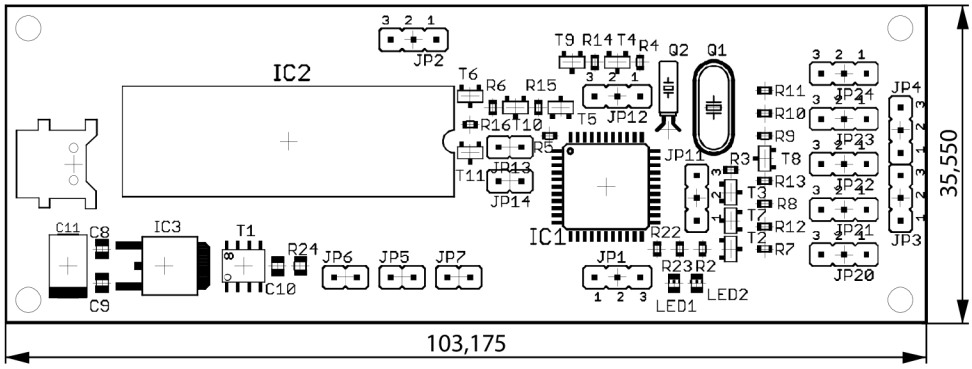


Spodná strana masky plošného spoja riadiacej jednotky.

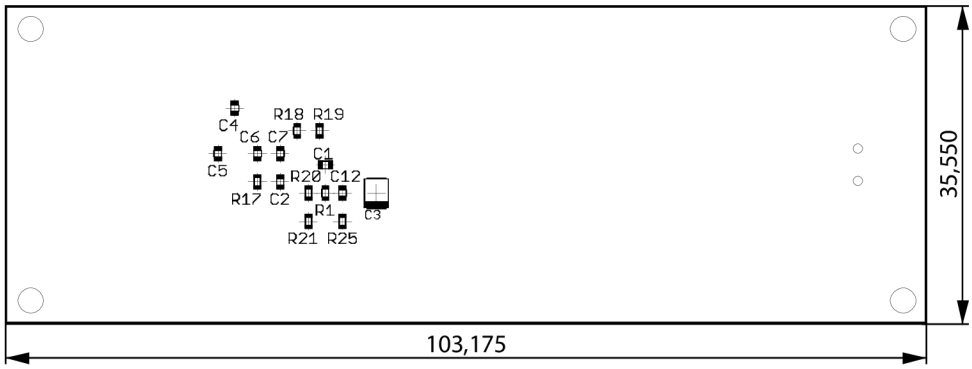


Príloha A.3

Osadzovací plán riadiacej jednotky z vrchnej strany plošného spoja.



Osadzovací plán riadiacej jednotky zo spodnej strany plošného spoja.



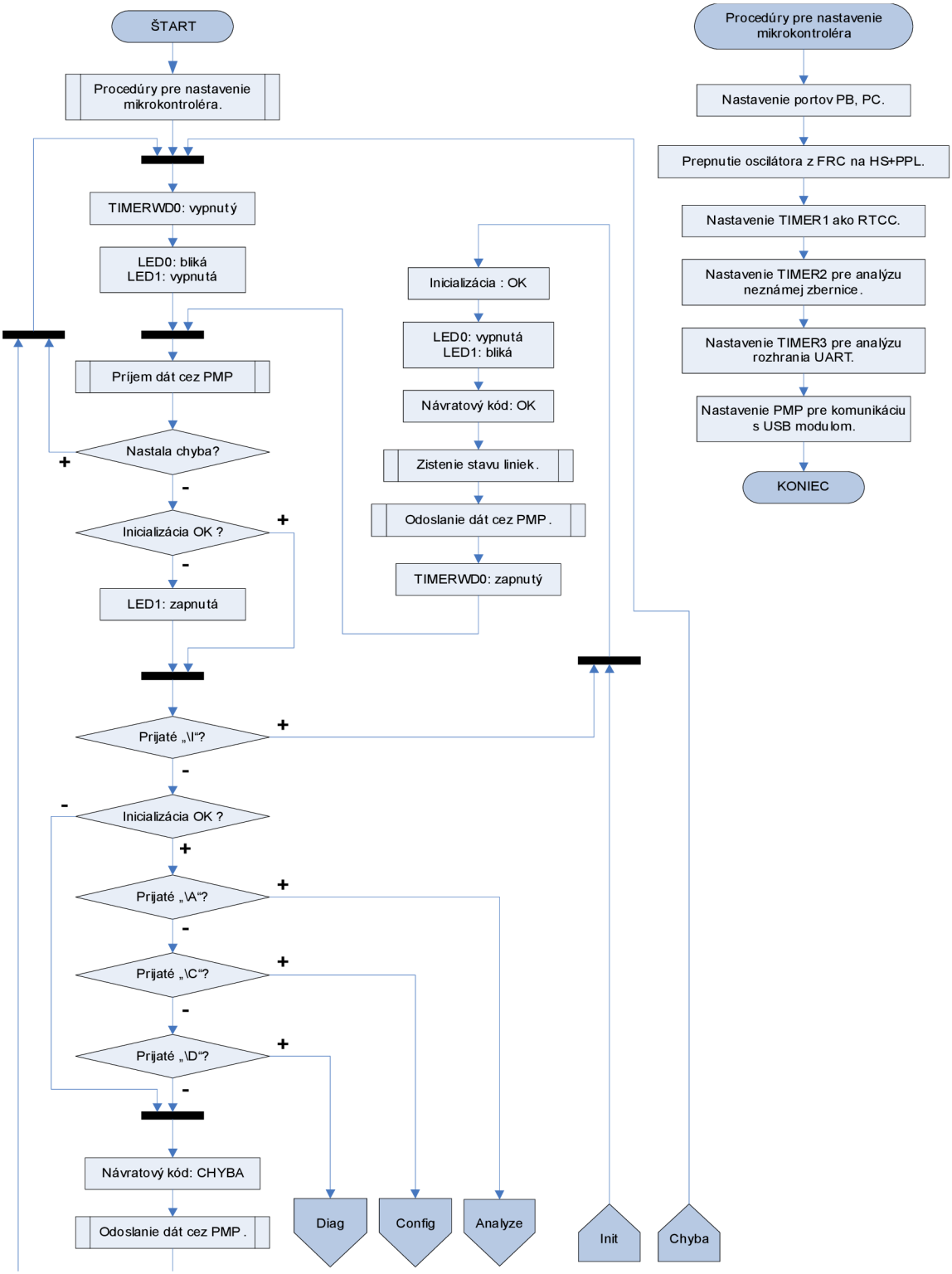
Príloha A.4

Zoznam použitých súčiastok.

Položka	Hodnota	Prevedenie
R1	10k	SMD 0603
R2–R6	6k8	SMD 0603
R7–R11	10k	SMD 0603
R12–R21	15k	SMD 0603
R22–R23	220R	SMD 0603
R24	1k	SMD 0603
R25	470R	SMD 0603
C1-C2, C8-C10, C12	100n	SMD 0603
C3	10µ/6,3V	SMC B
C4-C5	12p	SMD 0603
C6-C7	22p	SMD 0603
C11	100u/6,3V	SMC D
LED1	LED-RED	SMD 0805
LED2	LED-GREEN	SMD 0805
T1	IRF7210	SO8
T2-T11	BC817-40	SOT23
IC1	PIC24HJ128GP204	TQFP44
IC2	UMP2	DIP28
IC3	LF33CDT	DPAK
Q1	10MHz	HC49/S
Q2	32,768kHz	TC26H
JP1-JP4, JP11-JP12, JP20-JP24	3-pin JUMPER	JUMPER
JP5-JP7, JP13-JP14	2-pin JUMPER	JUMPER
J10	Mini-USB	MINI USB

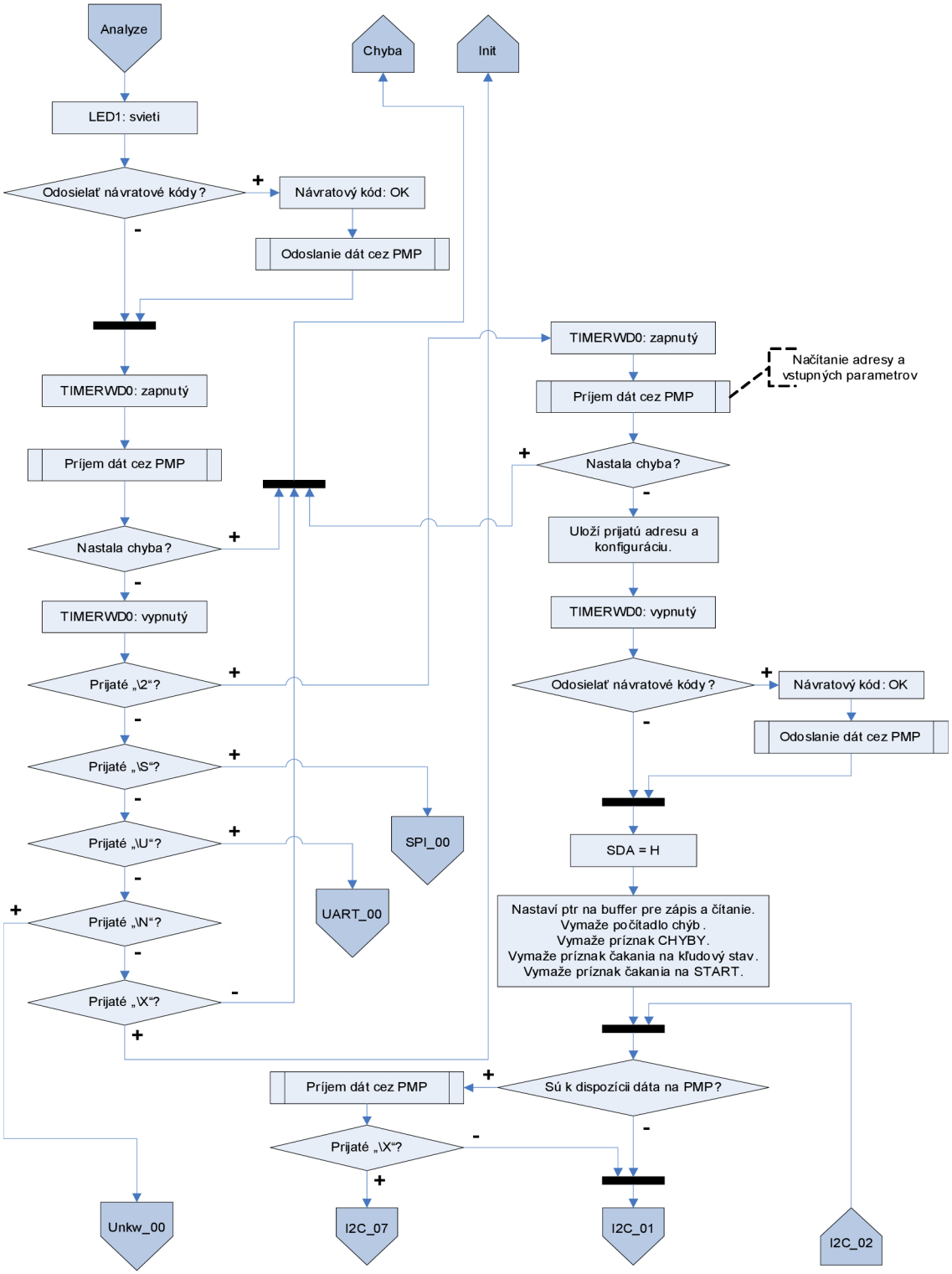
Príloha B.1

Podrobný vývojový diagram programu riadiacej jednotky (strana 1 z 9).



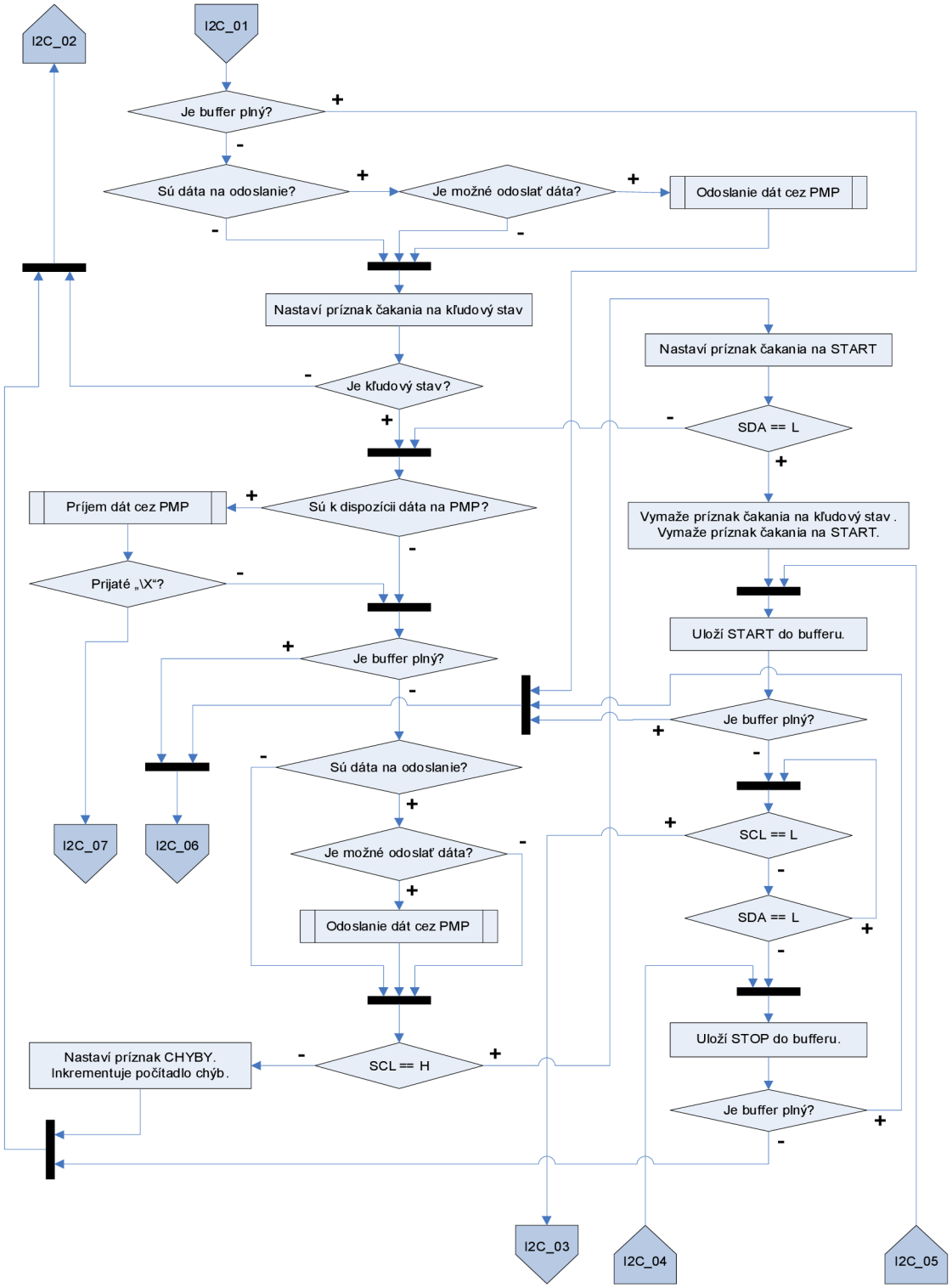
Príloha B.2

Podrobný vývojový diagram programu riadiacej jednotky (strana 2 z 9).



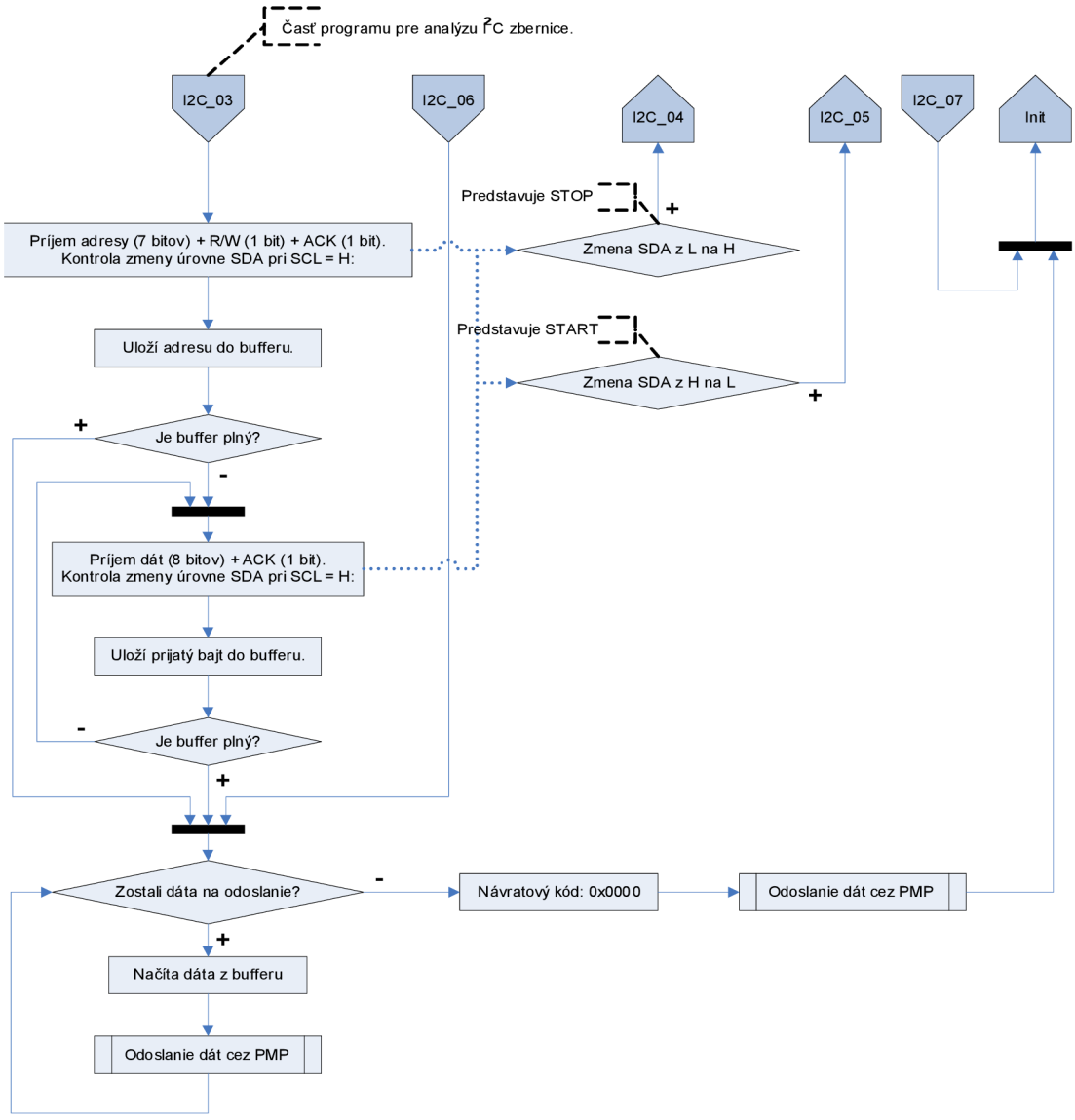
Príloha B.3

Podrobný vývojový diagram programu riadiacej jednotky (strana 3 z 9).



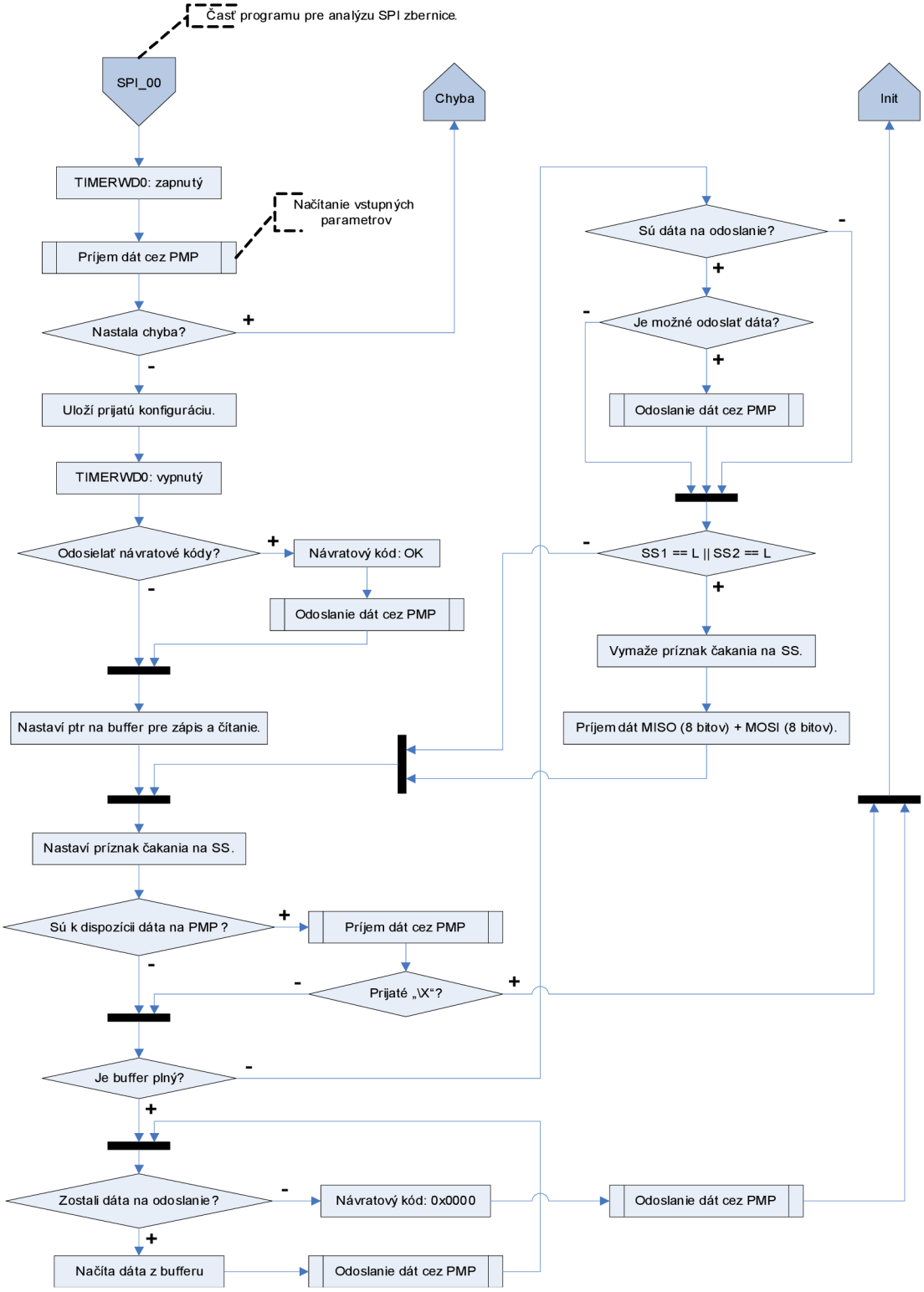
Príloha B.4

Podrobný vývojový diagram programu riadiacej jednotky (strana 4 z 9).



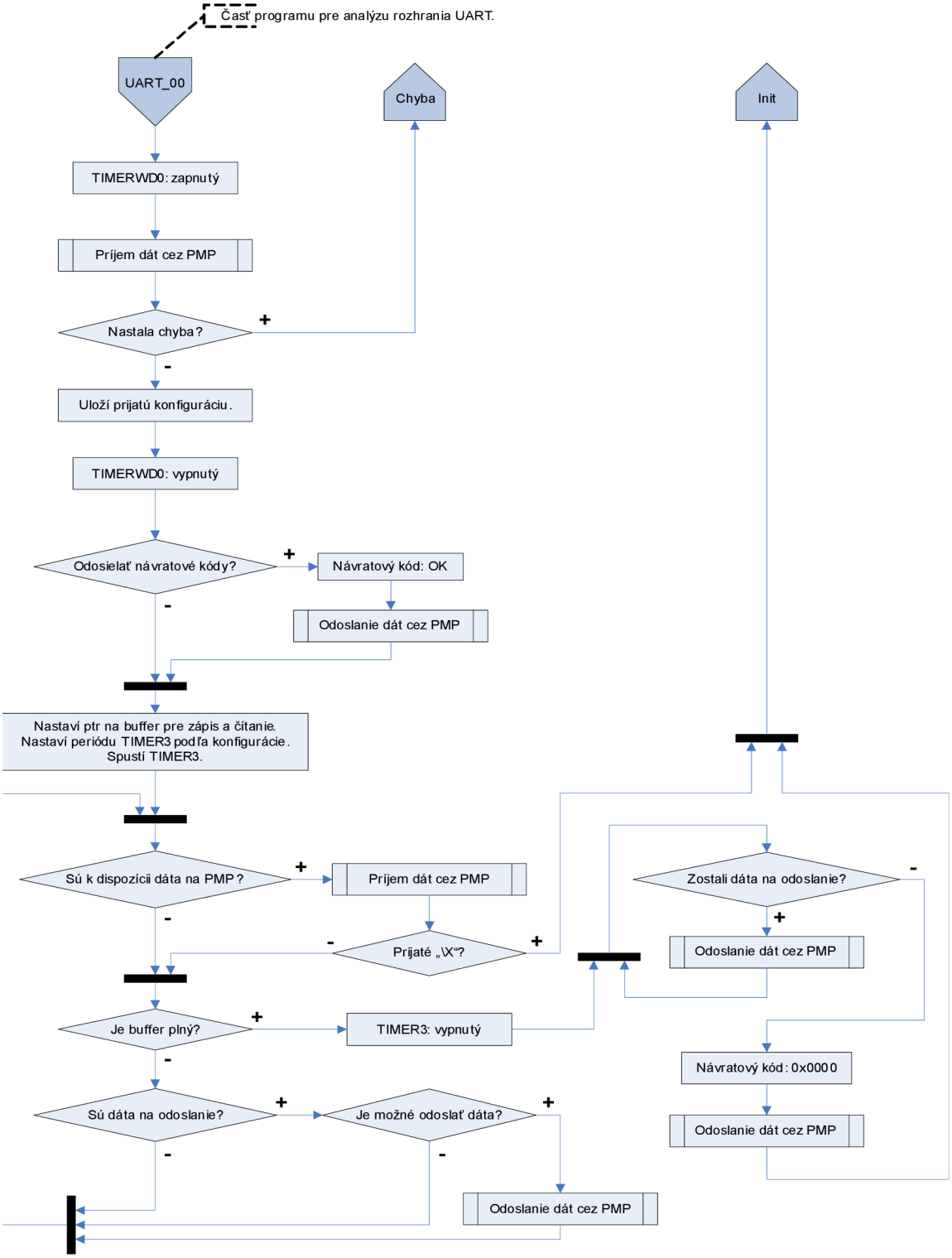
Príloha B.5

Podrobný vývojový diagram programu riadiacej jednotky (strana 5 z 9).



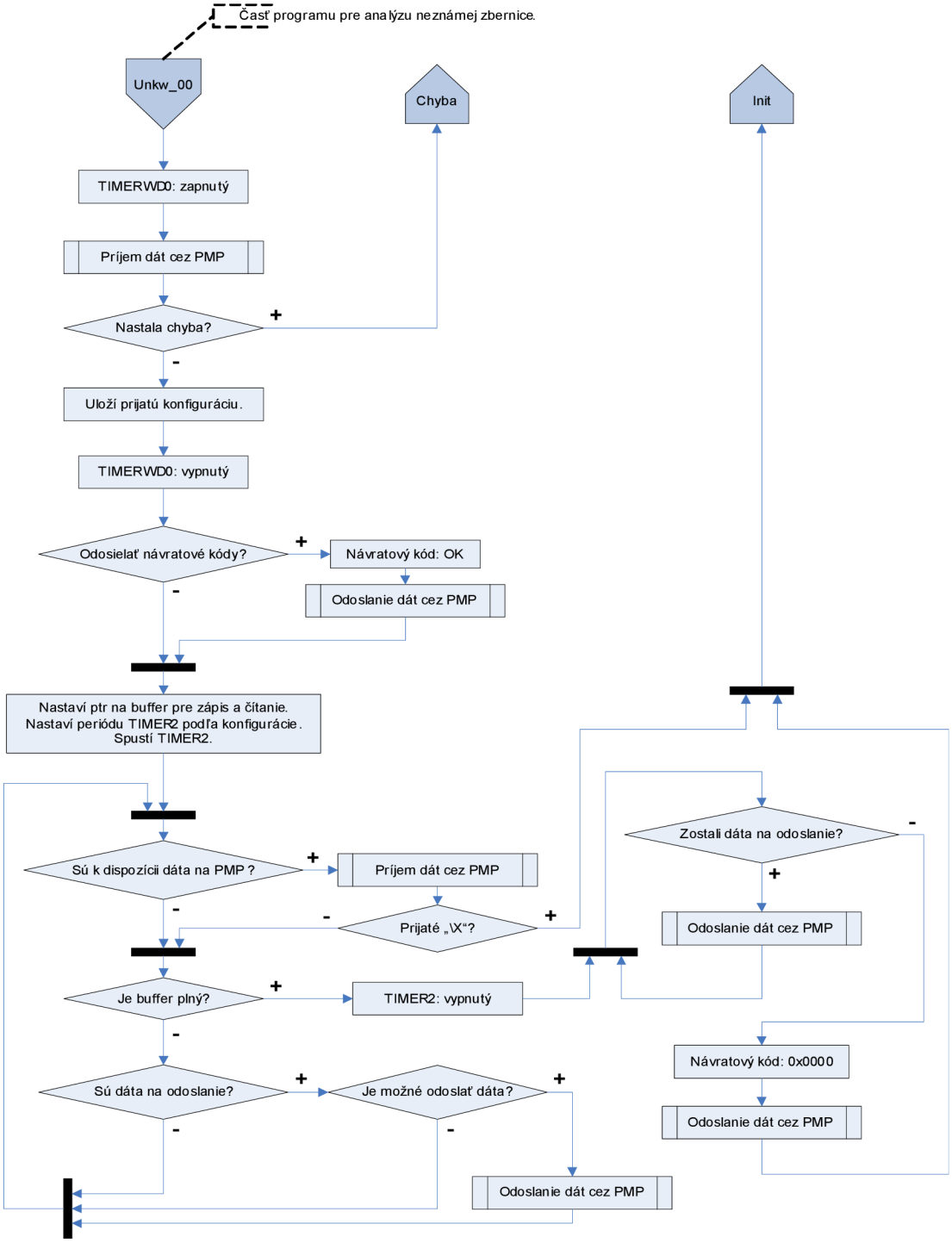
Príloha B.6

Podrobný vývojový diagram programu riadiacej jednotky (strana 6 z 9).



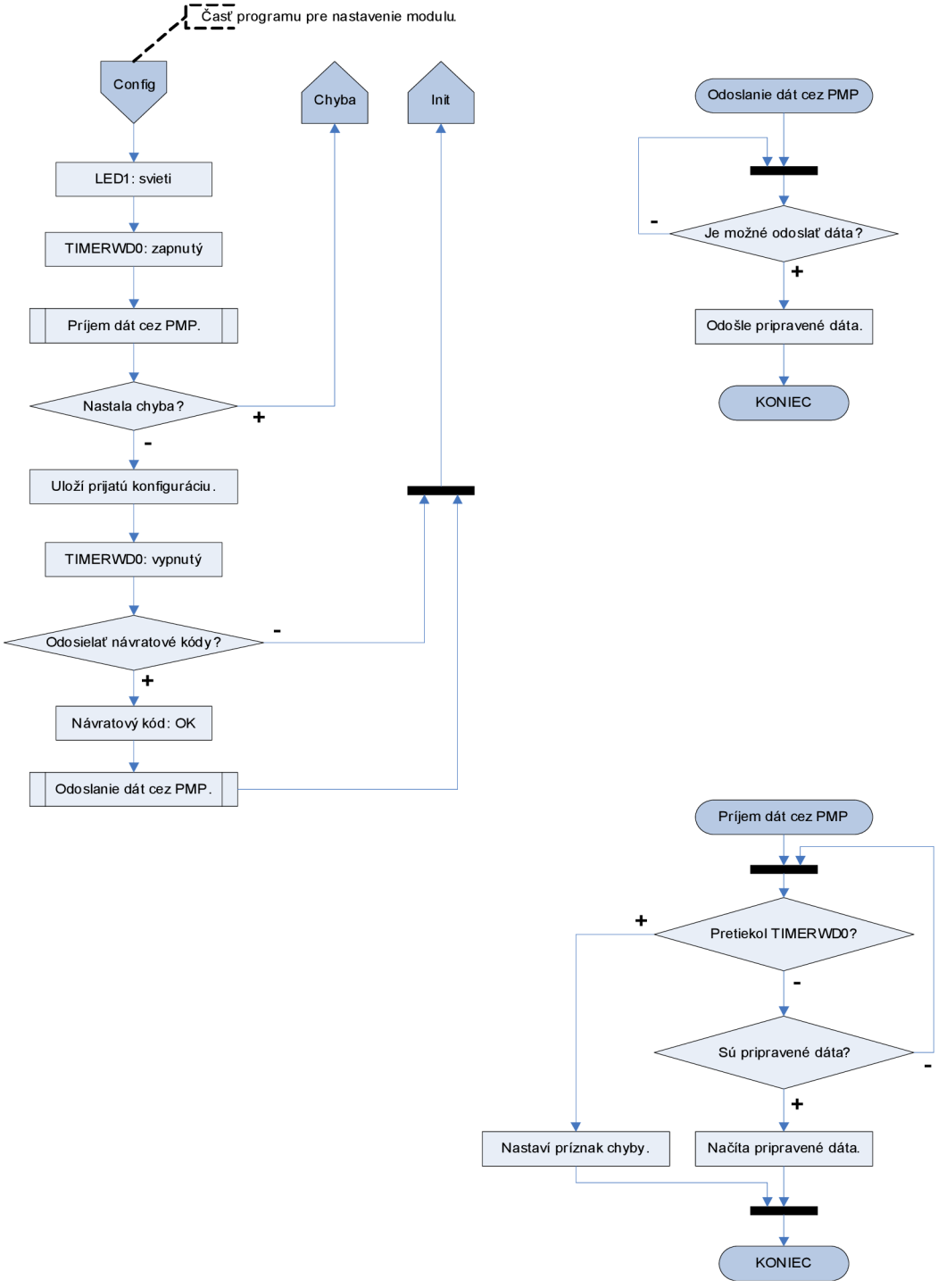
Príloha B.7

Podrobný vývojový diagram programu riadiacej jednotky (strana 7 z 9).



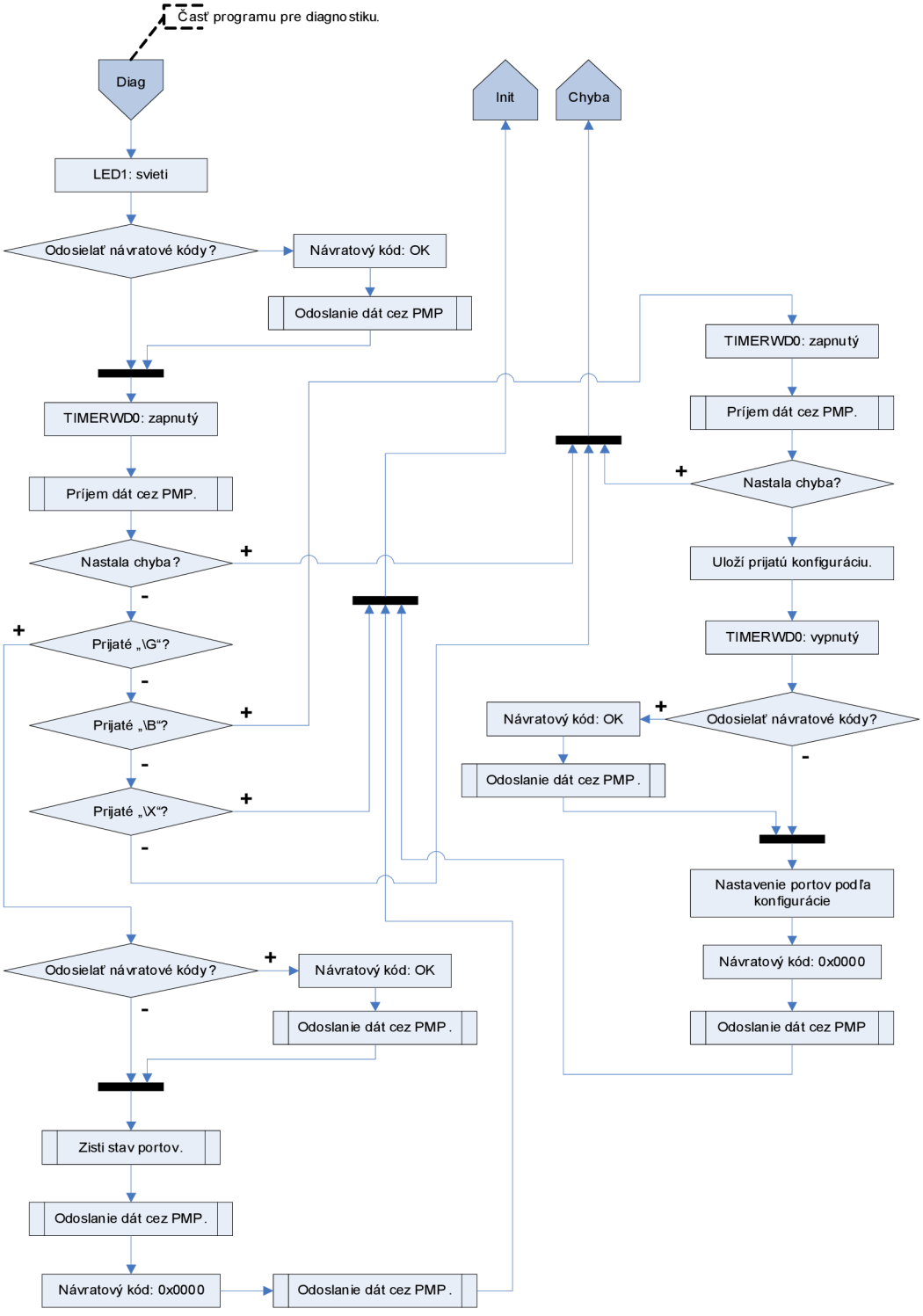
Príloha B.8

Podrobný vývojový diagram programu riadiacej jednotky (strana 8 z 9).



Príloha B.9

Podrobný vývojový diagram programu riadiacej jednotky (strana 9 z 9).



Príloha C

Stromová štruktúra priečinkov priloženého CD.

`\Documents`	- dokumenty a práca vo formáte PDF
`\Drivers`	- ovládače pre modul UMP2
`\Install`	
`\AdobeReader`	- prehliadač PDF súborov
`\dotNET`	- balík Microsoft .NET Framework
`\Eagle`	- návrhový systém pre tvorbu plošných spojov
`\FoxitReader`	- alternatívny prehliadač PDF súborov
`\MPLab`	- vývojové prostredie od spoločnosti Microchip
`\Program`	- skompilovaná užívateľská aplikácia
`\Sources`	
`\sBusAnalyzer`	- zdrojové súbory užívateľskej aplikácie
`\xBusAnalyzer`	- zdrojové súbory riadiacej jednotky