DIPLOMOVÁ PRÁCE

PPOFIBUS DP slave s připojením přes USB

Eduard OBORNÍK

leden 2004

Katedra řídící techniky Školní rok 2001 / 2002

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Eduard Oborník

Obor: Technická kybernetika

Název tématu: Profibus DP slave s připojením přes USB

Zásady pro vypracování:

1. Seznamte se s komunikací Profibus DP a s možnostmi realizace zařízení Profibus DP slave.
2. Realizujte takové zařízení, navíc s rozhraním USB, v úvahu berte možnost redundance na Profibusu DP.
3. Vytvořte software pro PC, který bude s vaším zařízením komunikovat přes USB.

Seznam odborné literatury: Dodá vedoucí práce.

Vedoucí diplomové práce: Ing. Pavel Burget

Datum zadání diplomové práce: listopad 2001

Termín odevzdání diplomové práce: leden 2003

Doc. Ing. Jiří Bayer, CSc.
vedoucí katedry

Prof. Ing. Vladimír Kučera, DrSc.
děkan

V Praze dne 22.3.2002

Anotace

Tato diplomová práce se zabývá návrhem zařízení PROFIBUS DP slave vybavené rozhraním USB.
Teoretická část práce popisuje základní vlastnosti PROFIBUS DP a jeho rozšíření o slave redundanci. Dále jsou uvedeny možnosti implementace USB a rozbor vlastností tohoto připojení.
Praktická část je věnována popisu návrhu a realizaci daného zařízení. To je postaveno na základě obvodů Siemens DPC31 a FTDI FT245BM.
V příloze lze nalézt schémata zapojení hardwaru. Zdrojové kódy vytvořeného programového vybavení jsou uvedeny na přiloženém CD.

Annotation

This diploma thesis designs a PROFIBUS DP slave device equipped with a USB interface.
The theoretical part of the thesis describes key features of the PROFIBUS DP and a possibility of its extension of a slave redundancy. It also describes the possibilities of implementation of the USB interface.
The practical part of my work describes the implementation of the device, which is based on Siemens DPC31 and FTDI FT245BM circuits.
In the appendix, there are the schematics of the hardware. Source codes of programs can be found on the enclosed CD-ROM.

Prohlášení

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

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 Zákona č.121/2000 Sb. , o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne ……. …….

podpis

Poděkování

Na tomto místě chci poděkovat vedoucímu mé diplomové práce Ing. Pavlu Burgetovi a konzultantovi Ing. Petru Smolíkovi za vytyčení směru práce. Dále pak Ing. Romanu Bartosińskému a Ing. Přemyslu Šůchovi za rady a pomoc při oživování zařízení.

Největší dík ale patří mým rodičům, kteří mi poskytli zázemí v průběhu studia i při vypracovávání této diplomové práce.

OBSAH

1. Úvod

Průmyslové sběrnice postupně vytlačují většinu zastaralých analogových informačních rozvodů. Jsou totiž mnohem přehlednější a na kabeláž méně náročné.

Sběrnice PROFIBUS (PROcess Field BUS) se stala standardem mezi sběrnicemi používanými v průmyslových podmínkách a díky neustálému vývoji si svou pozici upevňuje. V současné době jsou používány 3 varianty této sběrnice – PROFIBUS FMS, PROFIBUS DP a PROFIBUS PA.

Právě sběrnice PROFIBUS DP se stala základem mé diplomové práce. Jejím cílem bylo vytvořit zařízení, které bude přes tuto sběrnici komunikovat s podobně připojenými zařízeními a bude sloužit jako univerzální jednotka vstupů a výstupů.

V souladu se zvyšujícími se požadavky na bezpečnost automatizačních systému měla být uvažována i možnost redundance.

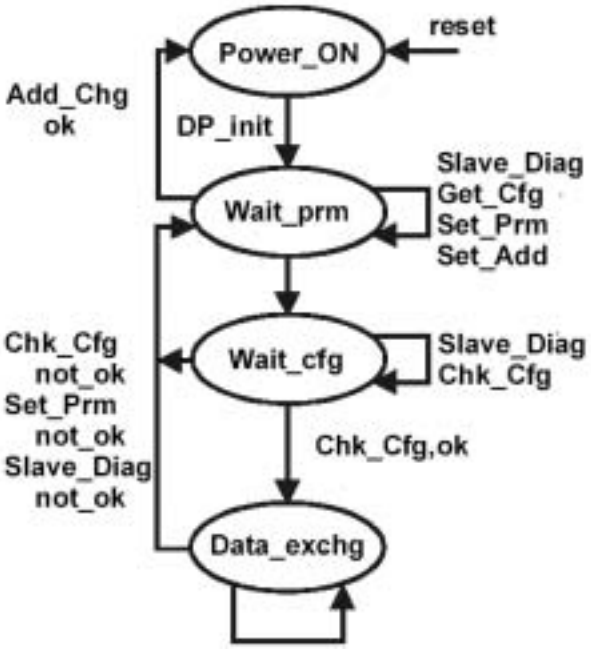
Pro snadnější ovládání a výměnu dat bude umět zařízení komunikovat přes sběrnici USB (Universal Serial Bus) s počítačem. Na počítači poběží program, který bude vizualizovat dění na sběrnici, potažmo na vytvořeném zařízení. Kromě toho bude program sloužit také jako ovládací terminál pro jednotku vstupů.

Diplomová práce může sloužit jako podklad pro další návrh obdobných zařízení, případně může být vytvořené zařízení nasazeno přímo do praxe. V tomto případě však doporučuji modifikaci programového vybavení podle specifických požadavků úlohy.

2. Profibus DP

S problematikou PRUFIBUSU DP jsem se seznámil v literatuře [1],[2],[4] a [5].

2.1. Komunikační model stanice DP slave



Obr. 1 Komunikační model PROFIBUS DP slave

2.2. Obvody pro implementaci protokolu Profibus DP slave

Je k dispozici široká škála zákaznických obvodů implementujících všechny, nebo některé funkce požadované od PROFIBUS slave zařízení. Mezi jejich nejvýznamnější výrobce patří firma Siemens. Nabízí obvody pro konstrukci nejjednodušších slave jednotek s funkcí vstupů/výstupů až po obvody integrující PROFIBUS rozhraní a díky integrovanému procesoru umožňující realizaci inteligentních slave jednotek.
Já jsem pro konstrukci zvolil obvod DPC31 od firmy Siemens. Tento obvod realizuje PROFIBUS protokol, obsahuje 6kB RAM a procesor C31 kompatibilní s řadou Intel x52.
Právě možnost využít vnitřní procesor byla jedním z hlavních důvodů k jeho volbě.

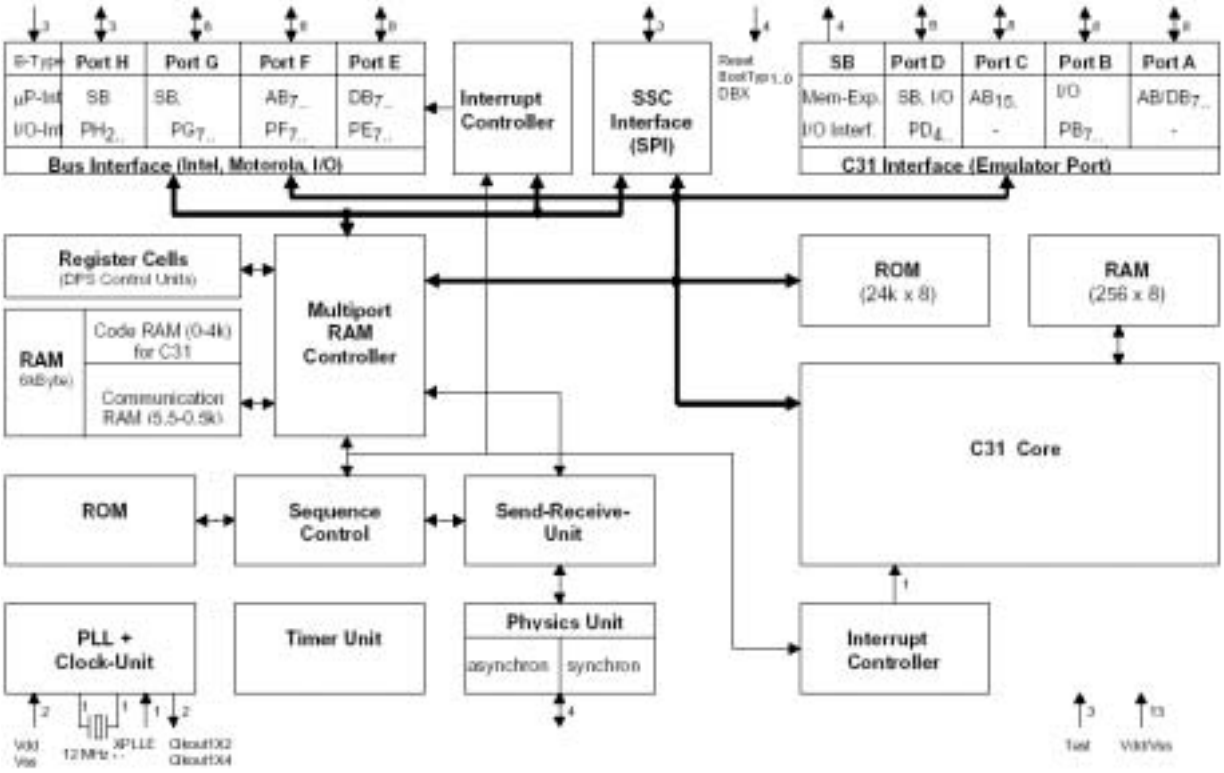
Je dodáván ve 100 vývodovém PQPF pouzdře.

Obr. 2 Obvod DPC31

2.2.1. DPC31

S obvodem Siemens DPC31 jsem se seznámil v [6]

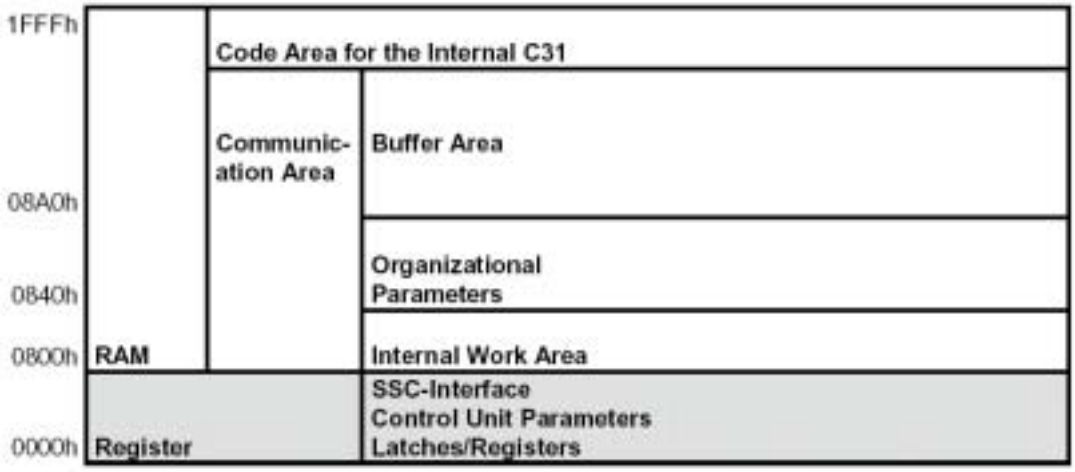
2.2.1.1 Bloková struktura



Obr. 3 Blokové schéma obvodu DPC31

2.2.1.2 Organizace paměti

Integrovaná 6kB paměť obvodu je rozdělena na několik částí.



Obr. 4 Rozdělení interní RAM DPC31

3. Slave Redundance

S rozšiřující se oblastí nasazování sběrnice PROFIBUS DP se zvyšují i požadavky na ni kladené. PROFIBUS DP byl k současnému datu rozšířen o dodatky DP/V1 a DP/V2, jenž obohacují PB DP o další funkce a tím i o další možnosti využití.

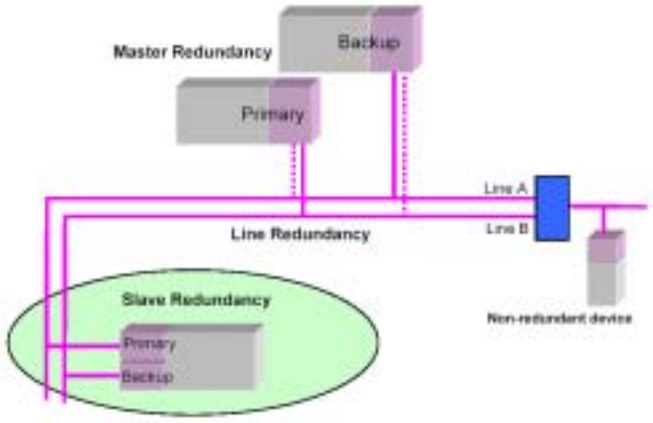
PROFIBUS DP/V1 znamenal rozšíření o

- Acyklické komunikace (MSAC_C1, MSAC_C2)
- Přenos alarmů
- FailSafe komunikaci (PROFIsafe)

PROFIBUS DP/V2 pak

- slave-to-slave komunikaci
- isochronní/synchronní výměnu dat
- časovou synchronizaci
- redundanci

Redundance koncových zařízení zvyšuje celkovou spolehlivost systému.



Obr. 5 Struktura redundantního systému

Na obrázku 5 je naznačena struktura kompletně redundantního systému. Jsou však přípustné libovolné kombinace redundance jednotek slave, master i linek. Každá úroveň redundance je na ostatních nezávislá.

Specifikace chování redundantních Slave je popsána v [3]. V dalším popisu budu pro přehlednost používat termíny tak, jak jsou uváděny v této literatuře.

Definice důležitých pojmů:

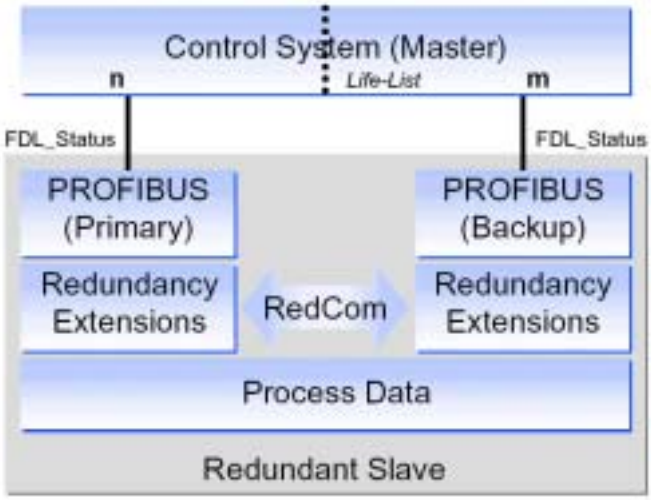
SIM (Slave Interface Module)	PROFIBUS slave jednotka s minimálně jedním připojením k PROFIBUSu
Redundatní slave	Pár SIM, kde jeden je Primary a druhý Backup Slave.
Primary slave	Slave modul, který vystupuje jako aktivní ve vztahu k PROFIBUSu a k procesu
Backup slave	Slave modul, který je pasivní ve vztahu k PROFIBUSu pro cyklickou výměnu dat (MS0) a acyklickou výměnu dat první úrovně (MS1) a k procesu a je připraven stát se primary v případě, že tento bude mít poruchu
RedCom	Komunikační kanál mezi dvěma SIM sloužící k výměně stavových informací
MS0	cyklická komunikace mezi DP-slave a DP-master třídy jedna (DPM1)
MS1	acyklická komunikace mezi DP-slave a DP-master třídy jedna
MS2	acyklická komunikace mezi DP-slave a DP-master třídy dvě (DPM2)

Redundantní Slave je definován jako jednotka, která obsahuje 2 nezávislé interface připojené k PROFIBUSu označené jako Primary a Backup. Může se jednat o 1 fyzický celek, nebo o 2 fyzické objekty nějakým způsobem propojené. Mezi těmito interface je vytvořen komunikační kanál Redundancy Communication (RedCom) nezávislý na PROFIBUSu.

Redundantní slave může pracovat na jednom i obou PROFIBUS připojeních.

Takovéto řešení redundance pro uživatele tedy znamená, že

- jediné zařízení lze provozovat při různých s různými strukturami redundantního systému.
- Redundance *Master*, *Slave* a komunikačních linek je na sobě vzájemně nezávislá
- Nejsou třeba žádné speciální konfigurační nástroje
- Je možné monitorovat stav obou částí Slave
- Není nijak ovlivněno zatížení sítě, tedy není ovlivněna dynamika odezvy PROFIBUSu



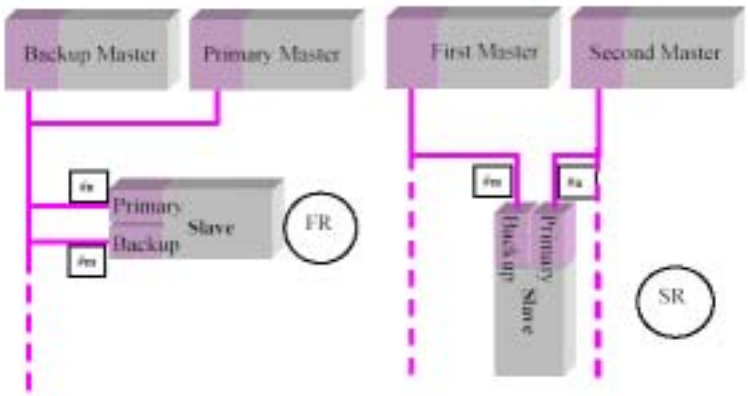
Obr. 6 Struktura jednotky slave

Redundantní slave tedy musí mít:

- 2 nezávislá připojení k PROFIBUSu
- 2 nezávislé PROFIBUS komunikační rozhraní
- komunikační kanál (RedCom) zajišťující propojení obou rozhraní nezávisle na PROFIBUSu

Redundantní slave je schopen pracovat s redundantním i neredundantním masterem. Slave redundance je z hlediska řídící aplikace transparentní a neklade na uživatele žádné nároky.

Redundantní slave lze provozovat ve 2 základních možných strukturách. Jsou označovány Flying redundancy (FR) a Systém redundancy (SR). Významně se liší v roli Backup slave a nastavení adres.



Obr. 7 Zapojení jednotky slave do systému

Pro FR platí, že

- PROFIBUS adresa primary slave #n je vždy odlišná od adresy backup slave #m
- Konfigurace je nezbytná jen pro jednotku primary
- Master nikdy nekomunikuje s jednotkou backup slave prostřednictvím MS0, či MS1

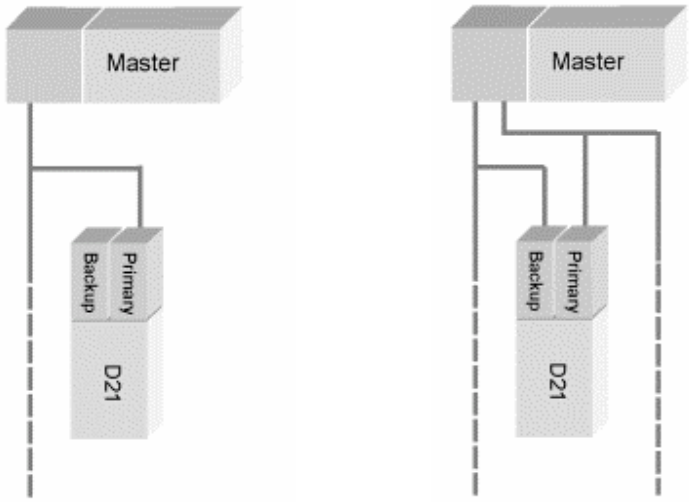
Pro SR pak platí, že

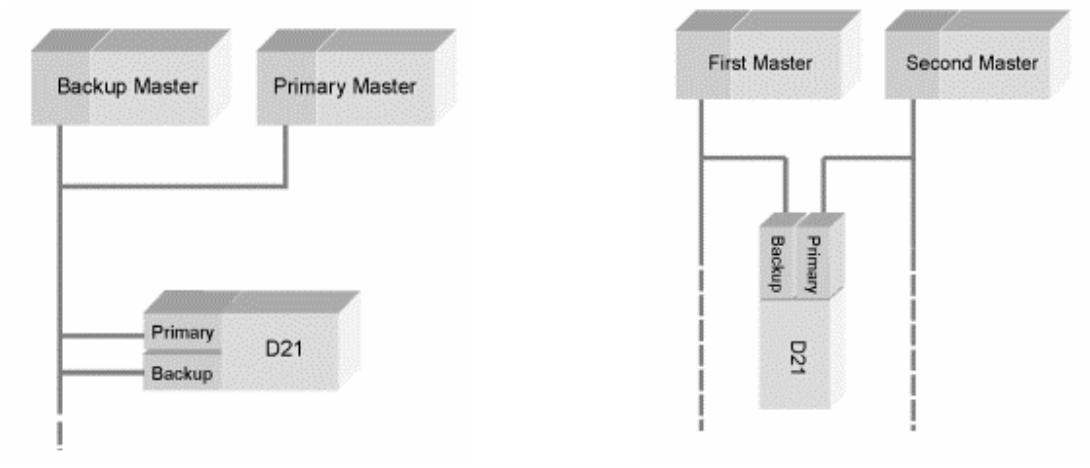
- PROFIBUS adresa primary #n se může shodovat s adresou slave #m
- Může proběhnout konfigurace obou jednotek slave
- Master může komunikovat s backup slave prostřednictvím MS0

Slave musí být schopen pracovat v obou strukturách, stejně jako pracovat s redundantním i neredundantním masterem.

Případná probíhající cyklická komunikace (MS0) s jednotkou backup nemá vliv na řízení procesu. Je použitelná redundantní jednotkou master jen k diagnostickým účelům.

Protože komunikace s backup není povinná, je třeba všechny diagnostické údaje jednotky backup přidat do diagnostiky jednotky primary.





Obr. 8 Používané kombinace zapojení Master-Slave

Blíže se budu věnovat chování redundantního slave zapojeného do struktury FR, protože v této konfiguraci měl být provozován slave v laboratoři katedry.

Komunikace (MS0, MS1) probíhá výhradně s Primary Slave. Jen tento Slave je zkonfigurován. Ve své rozšířené diagnostice však posílá i diagnostiku Backup Slave.

Dojde-li k poruše Primary Slave, Backup jednotka převezme jeho funkci. Výměna rolí *Primary-Backup* se nazývá *Redundancy Switchover*, nebo jen *Switchover*.

K *Redundancy Switchover* dochází za těchto podmínek:

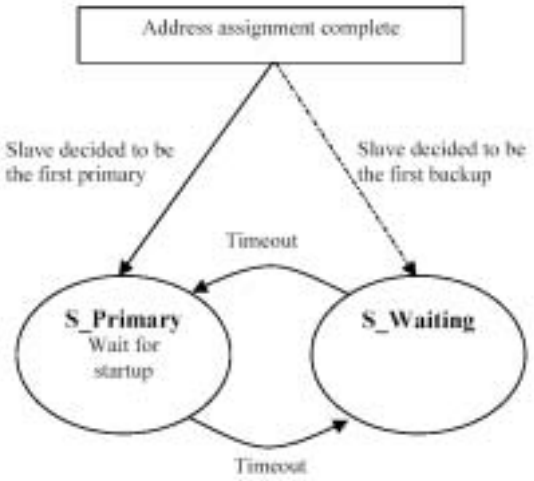
- *Primary Slave* sám o Switchover žádá (například když zjistí svou vlastní poruchu)
- *Backup* detekuje poruchu *Primary*
- *Backup* dostává příkaz k *Switchover od Mastera*

V případě Switchover je třeba dodržet určitá pravidla chování vstupních a výstupních dat. Během switchover nesmí v žádném případě dojít k jejich změně – to by mohlo vést k havárii řízeného systému. Backup slave tedy musí znát stavy vstupů a výstupů v okamžiku přepnutí jednotek a tak je nastavit. Další možností je, aby slave použil hodnoty získané při první cyklické výměně s masterem. V případě, že po přepnutí nedojde s jednotkou master k navázání komunikace, jsou hodnoty výstupů po určité době (OutputHoldTime) nastaveny na bezpečné (safe) hodnoty.

Podívejme se na příklad nejjednoduššího možného zapojení redundantního slave.

Mějme jednotku master, ke které je připojen redundantní slave. Po nastartování slave dojde k inicializaci jejích jednotek a následně k navázání vzájemé komunikace přes RedCom. V EEPROM paměti je uložena adresa primary jednotky označená primary_addres.. Na základě dohodnutých vnitřních pravidel se rozhodne, která z jednotek bude primary a která backup. Primary jednotka aktivuje své PROFIBUS rozhraní s adresou

primary5addres a přejde do stavu S_Primary a čeká na parametrizaci. Jednotka backup přechází do stavu S_Waiting.



Obr. 9 Chování slave po StartUp

Pokud jednotka primary do doby StartUpTime není úspěšně zparametrizována, dojde k výměně role primary a backup a proces se opakuje. StartUpTime začíná na 2 sekundách, s každým přepnutím jednotek je zdvojnásoben, dokud nedosáhne hodnoty 32sekund.

Jakmile je jednotka primary úspěšně zparametrizována, vstupuje do cyklické komunikace. O tomto přes RedCom informuje slave jednotku. Ta inicializuje své připojení k PROFIBUSu s adresou primary_addres+64 a je připravena pro acyklickou komunikaci s jednotkami MasterClass2. V těchto stavech jednotky setrvají do doby, než dojde k požadavku na Redundancy switchover.

Z výše uvedených pravidel pro chování redundantních slave vyplývá, že pokud bude zaručena možnost vytvořit komunikační propojení RedCom mezi dvěmi jednotkami slave, nemusí se po hardwarové stránce dále lišit od slave neredundantního.

3.1. RedCom

Komunikační kanál RedCom je klíčovou součástí redundantního Slave. Od jeho kapacity a spolehlivosti se odvíjí vlastnosti celého systému. Jsou na něj proto kladeny tyto požadavky

- Vysoká přenosová rychlost
- Vysoká bezpečnost

Specifikace neurčuje jeho fyzickou realizaci, ani použitý způsob komunikace.

Existuje několik možností jak RedCom realizovat a je plně na uvážení návrháře, kterou možnost zvolí.

Jako příklad lze uvést:

- Sériová rozhraní
  - linka RS232, RS422
  - sběrnice RS485
  - I2C sběrnice
- Paralelní rozhraní
- Dvoubránová RAM

Každé řešení má své výhody i nevýhody. Pro linku RS232 mluví jednoduchost implementace a častá podpora ze strany mikroprocesorů. Jako nevýhody lze považovat nízkou rychlost přenosu ve srovnání např. s dvoubránovou RAM. Ta zase naopak vyžaduje např. těsnou vazbu mezi jednotkami, je problematické galvanické oddělení, …

Já jsem se rozhodl RedCom realizovat sériovou linkou. Toto rozhraní je plně duplexní, lze ho zajistit paritou, lze ho snadno implementovat do řídícího programu. Je podporován hardwarem procesoru, takže jej již více nezatěžuje. Procesor C31 navíc přímo umožňuje jeho obsluhu přes přerušení, která jsou generována s každým odeslaným i přijatým znakem. Je též možno provoz na RedCom monitorovat z PC s využitím běžně dostupných aplikací jako je např. Hyperterminál (standardní součást Windows). V úvahu připadá i simulace jedné dílčí jednotky redundantního slave jiným zařízením.
Galvanické oddělení signálů linky RS232 nebylo nutné, a proto jsem od něho nakonec po poradě s konzultantem upustil.

Přenosová rychlost je dána frekvencí přetečení vnitřního čítače procesoru. Tento čítač je napojen na taktovací frekvenci procesoru, při přetečení je automaticky nastaven na zvolenou hodnotu.
Přenosová rychlost se odvíjí od taktovací frekvence procesoru. Pro generování standardních rychlostí jsou vhodné krystaly např. 11,0592MHz, nebo 3,6864MHz. Taktování 24MHz zajišťuje procesoru vysoký výkon, ale s přijatelnou chybou lze dosáhnout jen omezeného počtu přenosových rychlostí. Ty jsou 1200,2400,4800 a 9600Baud.

Maximální dosažitelná přenosová rychlost je 125 000Baud. Parametry použitého přenosu jsou uvedeny v následující tabulce.

- 1 start bit
- 8 datových bitů, první LSb
- 1 bit sudé parity
- 1 stop bit
- žádné řízení toku dat

Na jeden byte dat je třeba 11bitů, teoretická maximální přenosová rychlost je tedy 11,1 kB/s.

Monitorování provozu na obou linkách RedCom z PC je možné při použití tzv. Y-kabelu.
Na straně PC ale můžeme narazit na problém s nepodporovanou rychlostí. Nastavením rychlosti na 9,6kBaud sice dojde ke snížení výkonu RedComu, ale můžeme pro monitorování použít běžné programy pro práci s sériovou linkou.

3.2. Připojení PROFIBUS Slave přes USB

Jednotky PROFIBUS slave bývají navrhovány s pevně danou funkcí popřípadě jako modulová zařízení. Určení pak vyplývá z hardwarové konstrukce vlastní jednotky, nebo zásuvných modulů.

Univerzální rozhraní však možnosti vyžití značně rozšiřují. Například je možno stav zařízení monitorovat z PC a výsledky podrobit statistickému zpracování. Hlavní výhodu však vidím v použití takové jednotky jako převodníku mezi PROFIBUS DP a libovolným virtuálním zařízením v podobě aplikace spuštěné na PC. Je tak odstraněna závislost na hardwarovému určení zařízení. Změna funkce je realizována pouze změnou programů jednotky slave a PC aplikace.

4. USB – Universal Serial Bus

Rozhraní Universal Serial Bus (dále USB) je v dnešní době nejčastěji používaným rozhraním PC. Ve verzi 1.0 vzniklo v roce 1996. U jeho zrodu stálo konzorcium firem (Compaq, Hewlett-Packard, Intel, Lucent Technologies, Microsoft, NEC, Philips) sdružených do organizace USB Implementers Forum. Postupně prodělalo vývoj přes verzi 1.1 k dnešní 2.0.

Význačné vlastnosti USB jsou:

- Přenos dat 1,5Mbit/s až 480Mbit/s
- Možnost připojit až 127 zařízení
- Napájení zařízení do odběru 0,5A po sběrnici
- Podpora Plug & Play

Díky těmto parametrům toto progresivní rozhraní rychle vytlačuje dříve používaný sériový port RS232 či paralelní rozhraní LPT.

4.1. Vývoj USB

USB verze 1.0 z roku 1996 nebyla podporována výrobci hardware ani ze strany operačních systémů a nedosáhla většího rozšíření.
V roce 1998 bylo USB povýšeno na verzi 1.1. Tato verze definuje dva druhy zařízení komunikující různou přenosovou rychlostí. Nízkorychlostní zařízení *LowSpeed* (dále LS) pro rychlost 1,5 Mbit/s a rychlá zařízení *FullSpeed* (dále FS) pro rychlost 12 Mbit/s.
V roce 2000 byla uvedena verze USB 2.0. To s sebou přineslo typ vysokorychlostního zařízení *HighSpeed* (dále HS) pro připojení rychlostí 480 Mbit/s.

4.2. Architektura

Jsou rozlišovány 2 druhy jednotek – rozdělovače a koncová zařízení. Koncová zařízení jsou například tiskárna, myš, mikrofon, reproduktory, atd. Rozdělovač (hub) je určen k připojování jednotlivých segmentů a koncových zařízení do víceúrovňové hvězdicové struktury. Lze připojit až 127 zařízení v celkem 7 úrovních .

4.3. Fyzická vrstva

Propojovací kabel a konektory jsou standardizovány. Komunikace probíhá diferenciálně po 2 vodičích, navíc jsou v kabelu 2 vodiče určené pro 5V napájení zařízení.

Přenosové rychlosti jsou určeny podle klidové napěťové úrovně datových vodičů jako

- Low speed 1,5 Mbit / s
- Full speed 12 Mbit / s
- High speed 480 Mbit / s

Důležitou vlastností USB je, že připojená zařízení mohou komunikovat vzájemně odlišnými rychlostmi.

4.4. Druhy přenosů

Na USB jsou definovány 4 druhy přenosu

- Řídící
- Izochronní (přenos ze zaručenou šířkou pásma a zpožděním)
- Přenos objemných dat (Bulk)
- Interupt (přenos malého objemu dat s definovaným zpožděním)

Řídící (control) přenos se používá pro systémovou komunikaci mezi USB zařízeními. Izochronní (isochronous) přenos je vhodný např. pro přenos obrazu nebo zvuku. Je pro něj možno vyhradit definovanou šířku přenosového pásma a zaručit rychlost přenosu, není však zaručena bezchybnost přenosu, protože příjem dat není cílovým zařízením potvrzován. Přenos objemných dat (též Bulk) slouží k přenosu objemných dat, přenos je zajištěn CRC a příjem je potvrzován. V případě chyby, je přenos opakován. Zpoždění dat není definováno. Interupt přenos je určen pro přenos malého množství dat (jednotky byte), přičemž dopravní zpoždění je určeno. Detaily časování komunikací na USB lze nalézt přímo v normě [12].

Každý z přenosů je vhodný pro určitou oblast použití podle toho, zda požadujeme garantovanou šířku pásma, zpoždění či bezpečnost přenosu.

Izochronní přenos bývá využíván např. při přenosu zvuku, *bulk* při komunikaci s tiskárnou nebo flash-disky, v *interupt* režimu pracuje myš.

4.5. Obvody pro USB

Způsobů, jak integrovat USB rozhraní do aplikace je několik.

- Implementovat celý protokol do firmware v procesoru
- Využít procesor USB rozhraním vybavený
- Využít speciálních jednoúčelových obvodů
- Využít univerzálních USB radičů

Podrobný rozbor obvodů vhodný pro konstrukci USB periferií je uveden v [12].

SW implementace do procesoru aplikace je řešení vývojově nejsložitější a obecně lze dosáhnout jen nízkých přenosových rychlostí. Procesor vybavený USB rozhraním nabízí vysoký komfort obsluhy komunikace, ale dostupnost těchto procesorů je problematická. Speciální jednoúčelové USB obvody nalézají využití při konstrukci zařízení jako např. USB MP3 dekodérů, převodníků USB na IrDA, USB A/D převodníků, atd.
USB univerzální řadiče zajišťují komunikaci na straně USB, parametry přenosu se nastavují obsluhou řídících a datových registrů. Komunikace s obvodem ze strany uživatele je např. po datové sběrnici a řídících signálech, nebo např. po sběrnici IIC.

Já jsem se pro implementaci USB rozhraní do vyvíjené aplikace rozhodl použít obvod firmy Future Technology Devices International Ltd. (dále FTDI).
Obvod svými parametry plně vyhovoval stanoveným požadavkům na přenosovou rychlost, způsob připojení k procesoru a jednoduchost obsluhy. Navíc je lehce dostupný v kusovém množství (v ceně cca 200Kč za kus) a výrobce ke svým produktům zajišťuje příkladnou podporu. Pro USB řadiče jsou k dispozici ovladače pro systémy MS Windows i Linux. Dále je možné využít i DLL knihovny funkcí.

4.6. Obvody FTDI řady FT

Firma FTDI má v současné době ve své nabídce 2 obvody pro koncová USB zařízení.

- FT232BM (konvertor UART <-> USB)
- FT245BM (8bit FIFO brána <-> USB)

Jde o druhou generaci těchto obvodů, oproti prvním verzím FT8UxxxAM došlo k některým vylepšením. Nově je podporováno USB 2.0, přibyli nové funkce a širší možnosti nastavení zvětšují oblasti použití. Navíc byl zmenšen počet potřebných externích součástek a zlepšeny elektrické parametry obvodů.

4.7. Vlastnosti obvodu FTDI FT245BM

- Protokol USB 2.0
- Konverze USB – 8bit FIFO
- Přenosová rychlost až 1Mbyte/s
- Řízení zápisu a čtení signály WR, RD
- Vyrovnávací vstupní paměť 384B, výstupní 128B
- Indikace stavu obvodu signály RxFull, TxEmpty
- Volitelně externí EEPROM s identifikátory VID a PID
- Bulk nebo izochronní přenos

Obr. 10 USB řadič FT245BM

Obvod podporuje full speed rychlost 12Mbit/s dle prototokolu USB2.0. Detaily komunikace jsou před uživatelem částečně skryty - jeví se jako 8bitový port s funkcí fronty FIFO, čtení a zápis na USB se provádí řídícími signály. Stav vyrovnávacích pamětí je indikován signály stavovými. K čipu lze volitelně připojit sériovou paměť EEPROM. Její obsah dále určuje volitelné parametry obvodu potřebné pro jeho enumeraci při připojení na sběrnici. Takto se tedy nastavuje identifikační číslo výrobce (*Vendor ID* – přiděluje *USB Implementers Forum.* Pro FTDI bylo přiděleno 0403.), identifikační číslo produktu PID (*Product ID*), textový řetězec identifikující výrobce (*Manufacturer*) a jméno (*Description*) zařízení pod jakým bude vystupovat v systému. Dále používaný režim přenosu dat, spotřeba, podpora Plug & Play, sériové číslo výrobku a další. Není-li paměť použita, nebo je prázdná, obvod pracuje se základním nastavením. Její použití je ale doporučeno a případě připojení dvou zařízení k jednomu PC je prakticky nezbytné pro jejich bezchybné rozlišení přes sériová čísla.

Údaje je možné do EEPROM nahrát v programátoru na přesně definované pozice, nebo využít k tomuto účelu výrobcem vytvořenou aplikaci *EEPROMutility.* Do EEPROM se pak zapisuje přímo přes USB.

Isochronní přenos není dodávanými ovladači podporován. Je třeba si od FTDI speciální ovladače vyžádat, nebo vytvořit ovladače vlastní.

Data určená k odeslání jsou postupně zapisována do bufferu velikosti 384 byte. Perioda odesílání dat je nastavitelná na 1 až 256 ms. Odeslána jsou při vypršení určené doby (nastavitelné na 1 až 256 ms) nebo při obdržení příkazu *SendImediate* na pinu SI/WU. Odeslání je vždy vázáno na Bulk-In požadavek z PC. Na USB1.1 tento přichází 1x za 1ms, teoretické maximální rychlost je tedy 384kB/s. Na USB 2.0 lze dosáhnout rychlosti až 1MB/s. Kapacita sběrnice je sdílená a proto tyto hodnoty platí pro jedno USB zařízení připojené na sběrnici.

4.8. Dosažitelné přenosové rychlosti

Přenosové rychlosti dosažitelné při řízení procesorem řady x52 závisí na jeho výpočetním výkonu daném taktovací frekvencí. Procesor musí zapsat data na vstup obvodu a potvrdit zápis na signálu WR.
Instrukční cyklus procesoru C31 taktovaným na 24MHz trvá 500ns. Většina instrukcí je jednotaktových, nebo dvoutaktových.

Teoretická maximální rychlost přenosu dat je 488kB/s. Jedná se o periodické vysílaní konstanty v nekončné smyčce (4 instrukce / odeslání 1B) a pro praktické využití nemá smysl.

Při běžné práci s daty lze tedy očekávat rychlosti výrazně nižší.
Jako reálný příklad uvedu odesílání textového řetezce ukončeného nulou uloženým v externí paměti.
Maximálně efektivní kód toto realizující dosáhne poměru 10 instrukčních cyklů na 1 odeslaný bajt, tedy přenosové rychlosti cca 195kB/s.
Při nezbytném rozšíření o testování, zde je vysílací buffer připraven přijmout další data, rychlost dále klesá na cca 160kB/s.

4.9. USB jako připojení k PROFIBUSU

Navrhované zařízení mělo plnit úlohu převodníku mezi PROFIBUS DP a USB. Konkrétně jsem jej uvažoval jako jednotku 8 logických vstupů (DI) a výstupů (DO). Mělo tedy jít o přenos 1 byte dat směrem do PC (výstup) a 1 byte dat směrem na PROFIBUS (vstup) v každém cyklu sběrnice PROFIBUS.
USB je sběrnice sdílená a její propustnost závisí na počtu a aktivitě jednotlivých zařízení. Jsou definována limitující pravidla, jak má řídící počítač přidělovat šířku pásma jednotlivým přenosům. 10% kapacity (pro HS zařízení 20%) je vyhrazeno pro řídící přenosy, zbylých 90% (pro HS zařízení 80%) je rozdělováno mezi periodické (*interupt* a *izochronní*) přenosy. Zbývá-li nějaká volná kapacita, je přidělena *bulk* přenosu objemných dat. Konkrétní způsob distribuce kapacity však záleží na operačním systému a není striktně určen.
Dle konkrétního požadavku na funkci převodníku a tedy i na druh přenášených dat je třeba zvolit vhodný druh přenosu dle specifikace USB a tuto volbu podrobit zevrubné analýze.

Já jsem od USB primárně požadoval vysokou přenosovou rychlost a garanci bezchybného přenosu dat. Tento požadavek splňoval *bulk* režim přenosu. Skutečnost, že na sběrnici v tomto přenosu není definováno zpoždění jsem při návrhu zařízení opoměl. Tím je omezena možnost využít tento interface pro real-time aplikace, je-li USB zatíženo dalšími zařízeními.

5. Návrh hardware

5.1. Programátor EEPROG

5.1.1. Požadavky kladené na programátor EEPROG

Nechtěl jsem být závislý na školním EEPROM programátoru, a proto jsem se rozhodl postavit programátor vlastní. Navíc jsem jej uvažoval použít jako podpůrný prostředek při vývoji desky USBslave. Kladl jsem na něj tyto požadavky:

- Programování EEPROM použitých v USBslave
- Rozhraní RS232 pro připojení k PC
- Procesor řady x52 pro kompatibilitu s procesorem obvodu DPC31
- Snadnou změnu programu procesoru
- Ověření zapojení a funkcí USB řadiče FT245BM
- Ověření zapojení a funkcí LCD

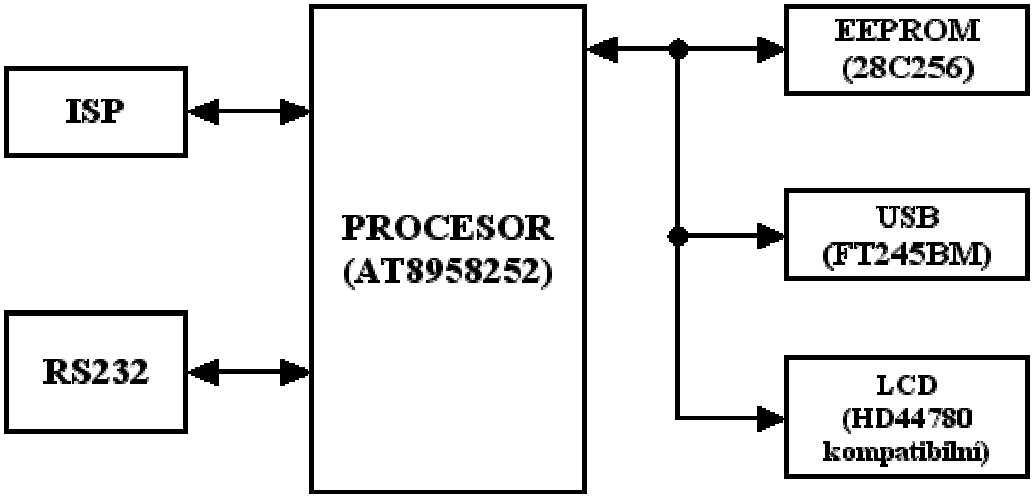
Ověření zapojení LCD a zejména USB řadiče jsem považoval za nezbytné před návrhem zapojení jednotky PROFIBUS slave.

5.1.2. Koncepce programátoru

Zařízení jsem pojmenoval EEprog, přestože programování EEPROM typu 28C256 a kompatibilních nebyla jeho jediná funkce.
Dále sloužil k ověření správnosti zapojení a funkce USB řadiče a LCD displeje a naprogramování rutin pro jejich obsluhu. Plánoval jsem jej využít i při vývoji aplikace pro komunikaci s PROFIBUS slave na PC.

Protože jsem neměl zkušenosti s programováním mikroprocesorů v jazyce C nezbytné pro implementaci funkce PROFIBUS slave v další části práce, chtěl jsem si používání kompilátoru uVision firmy Keil osvojit při programování tohoto programátoru.
Programátor je postaven na základě procesoru Atmel AT89S8252. Tento procesor umožňuje tzv. In System Programming (dále ISP) – tedy změnu jeho programu bez vyjímání z aplikace.
Pro programování procesoru slouží ISP rozhraní připojované na paralelní port PC. Tuto funkci jsem vyžadoval pro pohodlnou práci při testování programů.

Datové propojení s PC pro účely programování externí EEPROM, je realizováno sériovou linkou RS232. Programovaná paměť se vkládá do precizní patice. Nahrávaný program ve formátu Intel HEX se odesílá jako textový soubor na sériový port PC.
Po příslušné změně programu programátoru je přípustný i přenos po USB. Tím může být dosaženo výrazně kratších přenosových časů, ale získaná časová úspora by v tomto konkrétním případě nebyla adekvátní vynaloženému úsilí na změnu algoritmu.



Obr. 11 Blokové schéma EEprog

5.1.3. Řídící procesor

Jádrem programátoru je procesor Atmel AT89S8252 kompatibilní s řadou x52. Pro seznámení se s procesory řady x52 jsem použil skriptum [15] a internetové stránky [17]. Konkrétní informace vázané na procesor AT89S8252 jsem použil přímo z katalogového listu obvodu [16].

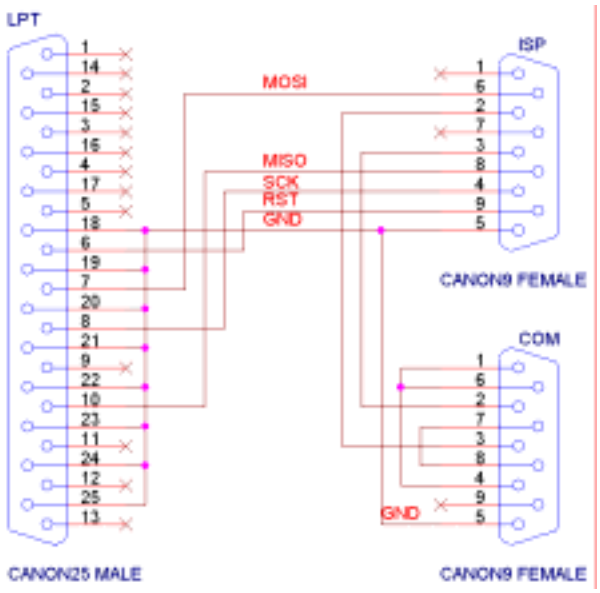
Procesor je vybaven vnitřní 8 kB FLASH pamětí programu. Tuto paměť lze programovat bez vyjímání procesoru z aplikace. Jedná se o tzv. ISP (In System Programming) programování přes sběrnici SPI. Jeho kompatibilita s řadou x52 a tím i s procesorem DPC31 použitým v jednotce SLAVE umožňuje vývoj částí aplikací tímto pohodlným způsobem a až po jejich odladění přenos na procesor DPC31.

Procesor je nyní taktovaný na 24 MHz. Původně byl použit krystal 11,0592 MHz, ale později jsem přešel na stejnou taktovací frekvenci, která je použita pro procesor DPC31 jednotky PBSL. Frekvence 24 MHz sice není příliš vhodná pro generování standardních přenosových rychlostí integrovaného UARTu, ale s původním krystalem nebylo možné ladění aplikací pro DPC31 se shodným časováním.

5.1.3.1 ISP rozhraní

Programování procesoru řídí aplikace z PC. Generuje programovací signály na paralelním portu (dále LPT), kam je propojovacím kabelem (viz. obr. 12) připojen EEPROG. Připojení na LPT je nejjednodušší způsob propojení. Není potřeba žádné další úpravy signálu, jako např. při připojení na port sériový. Signály ISP rozhraní jsou společně se sériovou linkou vyvedeny na 9pinový cannon konektor.
Pro zajištění možnosti resetu procesoru z PC i resetovacím tlačítkem na desce bylo na ISP nutno osadit diodu (D7), která zabraňuje přetížení paralelního portu PC napětím 5V z EEprogu.

Za řídící aplikaci jsem zvolil zdarma šířený program AEC_ISP 3.0 firmy AEC Electronics.



Obr. 12 Propojovací kabel ISP + RS232 rozhraní

5.1.4. LCD

Zvolil jsem displej kompatibilní s řadičem HD44780. Tyto textové displeje se vyrábějí v několika variantách provedení, lišících se počtem zobrazovaných řádků a počtem znaků. Jsou k dispozici ve verzích např. 8x2, 16x1, 16x2, 16x4, 20x2, 20x4, 40x2 znaků. Volit lze i mezi druhy podsvícení a technologií výroby. Já jsem zvolil cenově dostupný typ 16x2 znaků s LED podsvícením. Displej umožňuje doplnit pevně danou znakovou sadu až o 8 uživatelsky definovaných znaků.
Komunikace s displejem probíhá volitelně po 4 nebo 8 bitové datové sběrnici a 3 řídících linkách. Kontrast displeje lze nastavit velikostí napětí na vývodu ovládání kontrastu. Způsob komunikace s displeji kompatibilními s HD44780 je popsán v [18].

5.1.5. Mapování pamětí

Programovaná EEPROM paměť je vkládána do precizní patice připojené na externí datovou sběrnici procesoru. Pro zápis programu je k ní přistupováno jako k externí datové paměti.
Požadoval jsem, aby po nahrání programu do EEPROM jej bylo možné v programátoru i spustit a ověřit tak jeho funkčnost. Procesor řady x52 je harwardské koncepce, tzn. že paměť programu a dat je rozdělena a tento režim práce vylučuje. Zapojil jsem proto paměť dle koncepce Von Neumana, to znamená, že jsem paměťové prostory sloučil.

EEPROM je vybrána stále, čtení je iniciováno příkazem RD při přístupu do datové paměti, nebo PSEN při přístupu do paměti programu. Zápis je řízen signálem WR při zápisu do datové paměti.

Výběr mezi interní a externí pamětí programu se provádí zkratovací propojkou (JP1).
Na LCD ani USB řadič není možno přistupovat, jestliže procesor provozujeme s programem v externí paměti. Jejich využití je možné jen při běhu programu z interní FLASH EPROM procesoru.

5.1.6. Připojení periferií

5.1.6.1 Připojení LCD

Na port P2 procesoru (volně využitelný port, nebo multiplexovanou sběrnici dat a dolní poloviny adresy pro přístup do externích pamětí) je připojena i datová sběrnice LCD. LCD se připojuje přes konektor, konstrast se nastavuje odporovým trimrem, intenzitu podsvícení LED (je-li podsvícení displejem podporováno) je možno nastavit ve 2 úrovních zkratovací propojkou (JP2). Displej lze ovládat po 4 nebo 8 bitové datové sběrnici a ověřit tak oba přípustné režimy komunikace.

5.1.6.2 Připojení USB

Na port P0 procesoru (volně využitelný port, nebo horní polovina adresy pro přístup do externích pamětí) je připojena datová sběrnice USB řadiče FT245BM. Jde o konfiguraci USB zařízení napájeného z aplikace. Je použita EEPROM paměť pro nastavení volitelných parametrů obvodu. Popis obvodu lze nalézt v katalogovém listu [7]. Popis zapojení vychází z aplikačních poznámkách k obvodu [8].

5.1.6.3 Rozhraní RS232

Signály RxD a TxD integrovaného UARTu procesoru jsou budičem ICL232 převedeny na standard RS232 a vyvedeny na standardních pozicích 9 pinového konektoru canon („samec“). Propojení s PC lze tedy provést běžným tzv. kříženým null-modem kabelem.
Probíhající komunikace je na straně TTL úrovní signálů indikována LED. Aktivitu (nulovou úroveň) na RxD přijímací lince indikuje zelená a aktivitu na TxD vysílací lince oranžová LED.

5.2. PROFIBUS slave PBSL

5.2.1. Požadavky kladené na PBSL

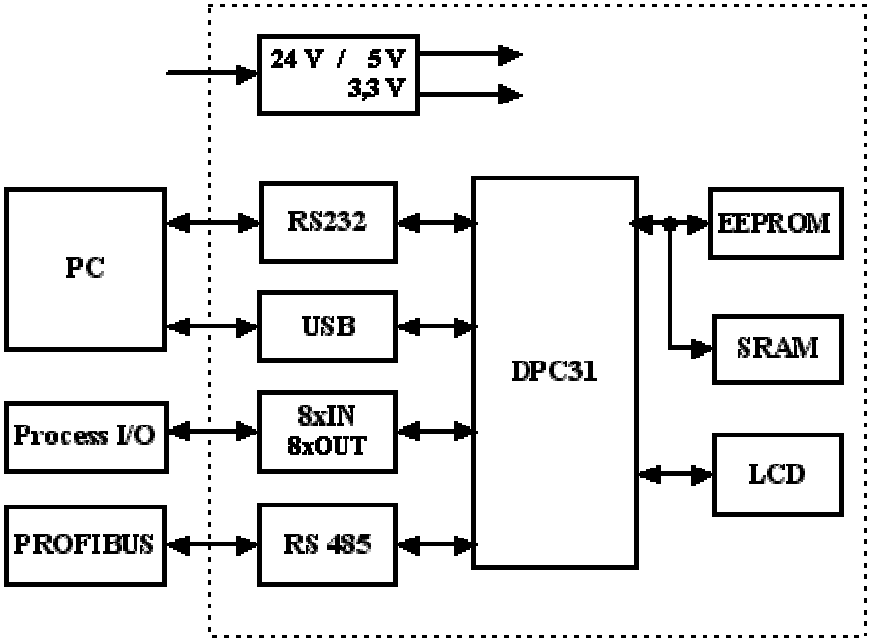
Na USB_SLAVE jsem kladl, z důvodu zabezpečení maximální funkčnosti při minimálních nákladech, dostupnosti součástek, universálnosti použití, jako i podmínek návrhu a výroby, následující požadavky:

- Jako jádro zařízení využít ASIC DPC31, použít jeho interní procesor C31
- Napájení step-down měničem
- Stav 8 logických vstupů a 8 logických výstupů indikovat LED
- Indikace stavů zařízení na LCD
- Rozhraní RS232 pro komunikaci s PC či vytvoření kanálu RedCom redundantního slave
- USB rozhraní pro komunikaci s PC
- Pro připojení k PROFIBUS DP užít galvanicky oddělené rozhraní RS485
- Paměť programu EEPROM 32kB
- Paměť dat SRAM 64kB
- Zálohování napájení SRAM baterií
- Možnost spouštět program z paměti SRAM
- Mapování adresových prostorů pamětí programu a dat obvodem GAL

Pro konstrukci zařízení jsem stanovil tyto požadavky:

- Použít 5V obvody logiky pro jejich snadnou dostupnost
- Použít obvody v DIL pouzdrech osazených do patic pro snadné oživování zařízení

5.2.2. Blokové schéma



Obr. 13 Blokové schéma zařízení PBSL

5.2.3. Organizace paměti

Kompilací vzorové knihovny V1SL k obvodu DPC31 jsem zjistil, že potřebná minimální paměť pro program je 10kB. Z tohoto důvodu připadalo v úvahu použít paměť 32, nebo 64kB. Z finančních důvodů jsem se rozhodl použít paměť velikosti 32kB. Přesto zvolená kapacita zajišťovala dostatečný prostor pro rozšíření o další funkce a odladění aplikace. Navíc jsem při návrhu uvažoval pro ladění používat tzv. BootLoader s funkcí mapování paměti programu do paměti dat, takže v případě potřeby by bylo možno spouštět programy až do velikosti 64kB.

Nároky na velikost datové paměti SRAM jsem nedokázal dopředu odhadnout, a tak jsem se rozhodl použít paměť 64kB. To je maximální velikost, jakou procesor dokáže adresovat. Dostupné byly ale jen paměti o velikosti 32 nebo 128kB. Volil jsem tedy 128kB, horních 64kB zůstává nevyužito. Cenový rozdíl v době návrhu byl 100Kč. Paměť 32kB stála 120Kč, paměť 128kB 220Kč. Dostatečná rezerva kapacity paměti však vyšší cenu zcela ospravedlňovala.

Dále jsem předpokládal, že v průběhu vývoje aplikace bude třeba poměrně často měnit vlastní program procesoru. Chtěl jsem se vyhnout neustálému vyjímání FLASH EEPROM z aplikace a

jejímu programování v programátoru (časté vyjímání by mohlo vést k jejímu poškození a kromě toho tento postup zdržuje). Proto jsem se rozhodl implementovat tzv. funkci BootLoader, což jekrátký zaváděcí program umístěný v EEPROM, který zajišťuje nahrání hlavního programu do SRAM a jeho spuštění.

Tento způsob práce s pamětí však vylučuje využití řídích signálů pro výběr pamětí tak, jak je generuje procesor C31. Z tohoto důvodu je generování řídících signálů třeba dělat dodatečně. Já jsem se rozhodl pro realizaci dodatečného generování těchto řídících signálů obvodem GAL. Provedení z diskrétních logických obvodů jsem zamítl pro jeho obvodovou složitost a hlavně kvůli nemožnosti případné změny výběrových funkcí.

Problematické volání obslužných rutin přerušení a příliš složitá konfiguraci procesu kompilace však nepřinášela žádnou časovou úsporu oproti externímu programování EEPROM. Problémy vyplynuli z nemožnosti umísťovat program do dolních 6kB RAM, které interně využívá DPC31.

Nakonec jsem tedy od používání BootLoaderu upustil. Program GALu byl tedy následně pozměněn a nyní realizuje přímý výběr pamětí, tak jak jej generuje DPC31.

5.2.4. Napájecí zdroj

Standardní průmyslový napájecí rozvod je 24 V. Při odhadnutém odběru 0,5A by z lineárního 5V stabilizátoru bylo třeba odvést ztrátový tepelný výkon téměř 10W.

$$(24-5)*0{,}5=9{,}5W$$

Pro stabilizaci napětí z 24V na 5V jsem proto použil integrovaný step-down měnič LM2474 s deklarovanou účinností 95%.

Měření stejnosměrného odběru, při oživování zařízení, je řešeno tak, že lze napájecí vedení těsně za měničem přerušit a vřadit do něj ampérmetr.

Napětí 3,3V pro obvod DPC31 dodává lowdrop stabilizátor napájený přímo z 5V.

5.2.4.1 Napájení SRAM

Jak jsem uvedl výše, uvažoval jsem využití paměti SRAM i pro uložení programu. Musel jsem tedy vyřešit zálohování napájení pro případ výpadku externího napájení obvodu.

SRAM je nyní zálohována 3,6V NiCd baterií.
Napájecí napětí je monitorováno napěťovým watchdog obvodem. Jeho výstup ruší výběr paměti při jeho poklesu a zabraňuje tak náhodnému zápisu. Automaticky je tím snížen odběr paměti z 50mA na 50µA.
Funkce obvodu je výrobcem garantována od okamžiku překročení hodnoty 3,6 V při náběhu napájení obvodu, do okamžiku poklesu napětí napájení pod 2 V. Abych zaručil bezchybnou funkci mimo tento interval, je zálohováno i napájení vlastního dohlížecího obvodu.

Společně s výběrem paměti SRAM je řízen i reset obvodu DPC31. Připojení úrovně 3,6V při vypnutém napájení je dle dokumentace pro DPC31 nepřípustné, a proto jsem toto napětí snížil děličem tak, aby nikdy nemohlo bezpečnou hodnotu překročit.

Při předpokládaném dlouhodobém odstavení zařízení je vhodné baterii odpojit rozpojením propojky JP2. Zabrání se tak úplnému vybití zálohovací baterie.

5.2.5. Rozhraní USB, LCD, RS232

Zapojení USB řadiče jsem převzal tak, jak jsem jej ověřil při konstrukci EEPROGu.

Pro komunikaci s LCD je použito zapojení se 4bitovou sběrnicí, jak jsem jej ověřil při konstrukci EEPROGu.

Rozhraní RS232 je opět v standardním zapojení. Stav linek TxD a RxD na straně procesoru je indikován LED.

5.2.6. TTL vstupy a výstupy

Volně použitelné 8x logické vstupy a 8x logické výstupy signálu jsou vyvedeny v TTL úrovních. Jejich stav je indikován LED.
Pro případný požadavek na dodatečné galvanické oddělení je na připojovací konektory přivedeno i napájecí napětí 5V.
Výstupy lze uvést do stavu vysoké impedance. Takto jsou nastaveny i v případě resetu procesoru aplikace.

5.2.7. PROFIBUS rozhraní

Při návrhu PROFIBUS budiče jsem vycházel z doporučeného zapojení uvedeného v [6]. Je dodrženo úplné galvanické oddělení jednotky slave od sběrnice.
Nepodařilo se mi však získat galvanické oddělovače pro rychlosti 12Mbit/s. Použil jsem běžné optočleny 6N137 vyhovující do přenosových rychlostí 10Mbit/s. Maximální dosažitelná komunikační rychlost na PROFIBUSu je tedy 6Mbit/s.

Navíc jsem byl momentální nedostupností v maloobchodě nucen jako budič RS485 osadit pomalejší typ DS485 namísto plánovaného SN75176. Obvod vyhovuje do rychlosti 2,5Mbit/s. Současná komunikační rychlost je tím omezena na 1,5Mbit/s.

Vysílání na sběrnici se strany PBSL jsem se rozhodl indikovat žlutou LED.

Na konektor RS485 je vyvedeno napětí 5V. Maximální přípustné proudové zatížení je 150mA.

6. Výroba a osazení plošných spojů

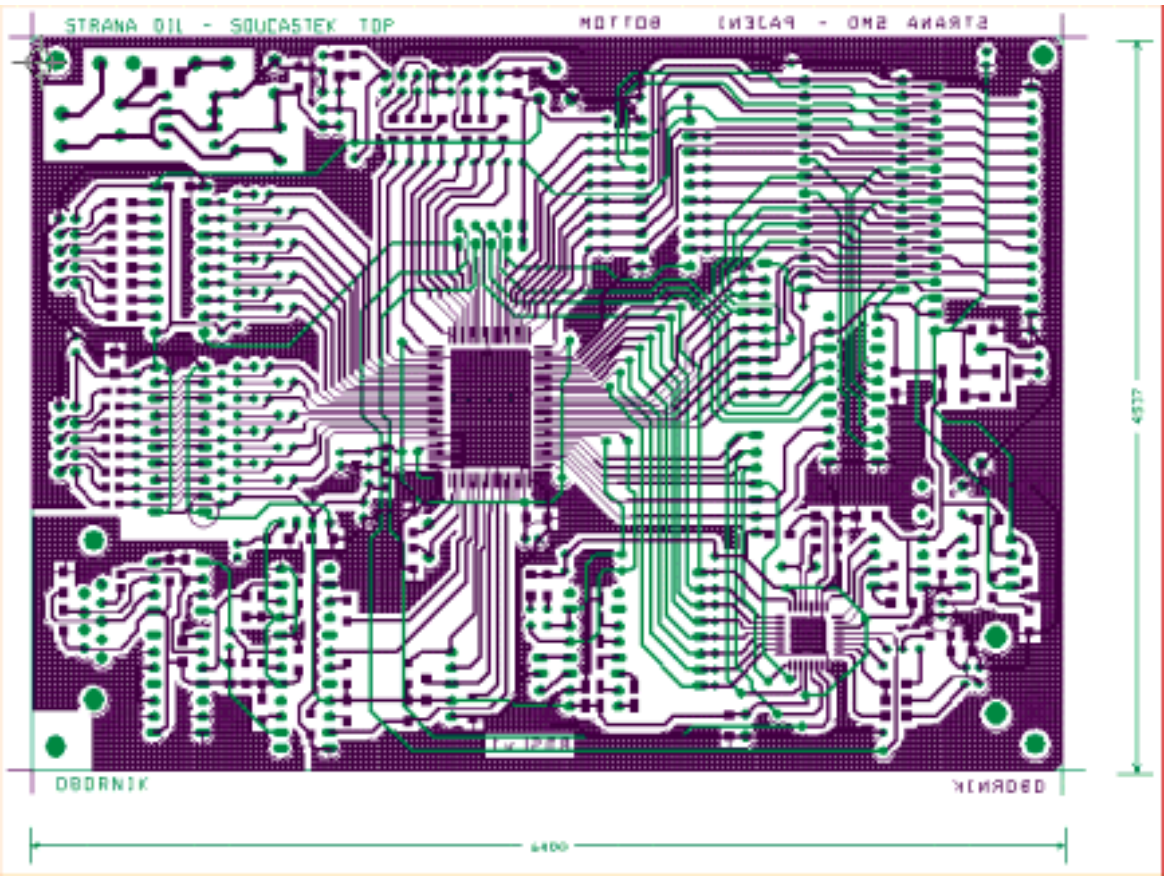
Plošné spoje jsem se rozhodl navrhovat v programu Layout programového balíku OrCad 9.11. Programátor EEprog jsem realizoval na jednovrstvé desce, USBslave pak na desce dvouvrstvé. Při návrhu jsem se snažil respektovat pravidla pro rozmísťování a propojování součástek tak, aby byla dodržena kritéria kladená na zařízení z hlediska elektromagnetické kompatibility (EMC). Čerpal jsem z [14].
Pro realizaci jsem si stanovil použít součástky klasické montáže pro všechny integrované obvody (IO) a povrchové montáže (SMT) pro pasivní součástky. Obvod DPC31 a USB řadič FT245BM byl dostupný jen v SMD provedení. Zde jsem musel od svého požadavku ustoupit.
Použití IO v DIL pouzdrech osazovaných do patic významně zjednodušilo postupné oživování zařízení a umožnilo pohodlné měření osciloskopem. Pozdější snadná výměna součástek, např. při podezření na jejich nesprávnou funkci, nebo vlivu na okolní součástky by v případě SMD nebyla myslitelná.

Jedním z mých cílů bylo dosáhnout nízkých výrobních nákladů desek a možnosti ručního osazení součástek. Plošné spoje jsem tedy navrhl ve IV. třídě přesnosti s minimem prokovů, robustními vodiči a dostatkem prostoru mezi součástkami. Všechny klasické součástky jsou osazeny z jedné a SMD z druhé strany plošného spoje. Je však přípustné i strojové osazení a jednoprůchodové pájení na cínové vlně.

Výrobu plošných spojů jsem svěřil firmě „Spoj“(www.volny.cz/plspoj), na kterou jsem měl dobré reference.
S kvalitou nejdříve vyrobeného plošného spoje zařízení EEprog jsem byl spokojen a rozhodl jsem se firmě zadat i zhotovení desky pro jednotku PBSL. Náročnost této zakázky se ukázala být na hranici technologických možností firmy a před osazením desky jsem na ní musel provést drobné korekce.

Vlastní osazení součástek jsem provedl sám.

Pro ilustraci zde uvádím obrazec spojů plošného spoje PBSL. Skutečná velikost je 16x11 cm



Obr. 14 Ukázka plošbého spoje PBSL

7. Oživení a vývoj softwaru

7.1. Oživení EEPROG

Oživení programátoru EEprog a implementace všech jeho funkcí byla těsně vázána na vývoj softwarového vybavení.

Po osazení součástek s průběžnou kontrolou odběru a jejím porovnáváním s katalogovými údaji jsem se pustil do ověření funkce jednotlivých periferií.

7.1.1. Oživení procesoru

Programy pro ISP programování procesorů generují řídící signály na paralelním či sériovém portu PC. Já jsem zvolil paralelní port. Mohl jsem tak EEPROG připojit k PC kabelem přímo bez úprav úrovní signálů.
Původně jsem chtěl využívat volně šířený (**www.lancos.com**) program PonyProg. Měl jsem na něj dobré reference, je často používán pro programování procesorů MicroChip PIC, Atmel AVR a sérivých EEPROM pamětí. Zjistil jsem však, že jeho podpora procesoru AT89S8252 je nedokonalá. Procesor bylo možno naprogramovat, ale pro spuštění programu pak bylo nezbytné odpojit programovací kabel. PonyProg ve verzi 2.06a totiž nemá vhodně ošetřené ovládání signálu RST programovaného procesoru, kdy je tento stále udržován aktivní.
Používal jsem proto jiný freeware program AEC-ISP (**www.aec-electronics.com**) vytvořený firmou AEC electrononics.

7.1.2. Ověření funkce sériového rozhraní

Na PC jsem spustil freeware program Serial. Ten umožňuje široké nastavení parametrů portu a pohodlné vysílání a příjem dat. Pro EEPROG jsem napsal aplikaci RS232loop, které po UART přijatá data odesílala zpět. Tak vznikla komunikační smyčka mezi PC a EEPROGem, kterou jsem ověřil správnou funkci sériové linky. Tento komunikační kanál jsem poté hojně využíval pro ladění dalších funkcí EEPROGu.

7.1.3. Oživení USB řadiče

Po prověření funkce sériového rozhraní jsem přistoupil k ověření funkce USB řadiče. Připojil jsem EEPROG USB kabelem k PC. Operační systém detekoval nové zařízení a vyžádal si příslušné ovladače. Dalším krokem bylo nastavení parametrů USB řadiče dle mých požadavků. To se děje přímo přes USB utilitou dodanou výrobcem čipu FT245B. Ta slouží k naprogramování sériové EEPROM paměti připojené k čipu. Význam a způsob programování volitelných parametrů EEPROM je popsán v [10].
Komunikace však opakovaně selhávala. Nakonec jsem zjistil, že je na vině nefunkční paměť EEPROM. Po její výměně se mi konečně podařilo nastavit požadované parametry přenosu dat, identifikátoru číslo výrobce VID (Vendor Identification Number), produktu PID (Product Identification Number) a název (EEPROG) koncového USB zařízení.

Poté jsem do procesoru naprogramoval aplikaci, která přijatá data z USB odesílala na UART a opačně. Na PC jsem v C++ Builder naprogramoval aplikaci USBloop, která odesílala data na USB a přijímaná data zobrazovala. Spolu s programem Serial, deskou EEPROG a programem USBloop vznikla komunikační smyčka, kdy z programu Serial odeslaná data se zobrazila v aplikaci USBloop a opačně.

7.1.4. Oživení LCD

Dále jsem pracoval na oživení LCD. Napsal jsem rutiny pro komunikaci, podařilo se mi implementovat všechny funkce displeje včetně podpory české znakové sady. Zde však chci upozornit na nutnost striktního dodržování výrobcem požadovaných průběhů řídících i datových signálů tak, jak jsou popsány v [18]. Jakákoliv odchylka, zejména při použitých vysokých přenosových rychlostech, může způsobit chybu komunikace.

7.1.5. Zápis do externí paměti

Poté jsme přistoupil k otestování funkce externí paměti. Naprogramoval jsem jednoduchý program, který měl blikat diodou na výstupu UART. Tento program jsem univerzálním školním programátorem naprogramoval do paměti a tuto vložil do desky. Procesor byl konfigurován na běh programu z externí paměti. Program se však neprováděl.
Pokusil jsme se tedy otestovat funkci paměti zapisováním a zpětným vyčítáním hodnot. Do interní paměti procesoru jsem nahrál program, zapisující konstantu do paměti, kterou opět čte a vysílá na terminál na PC.
Zjistil jsem, že opakovaným čtením z jedné pozice dostávám různá data. Hodnoty čteného bytu se vždy lišili v předposledním v bitu, 6. Věděl jsem, že takto je pamětí indikován probíhající zápis. Při návrhu jsem totiž udělal chybu a prohodil vodiče adresové sběrnice A14 a řídícího signálu WR. Paměť tak byla trvale v režimu zápisu. K chybě došlo při kreslení schématu. Místo schématické značky paměti 28C256 jsem použil značku paměti 29C256. Tyto paměti se liší právě umístěním A14 a WR.
Tuto chybu jsem na již vyrobené desce opravil drátovými propojkami.
Po této úpravě již bylo zařízení po hardwarové stránce plně funkční.

7.1.6. Vytvoření programu pro programování EEPROM paměti

Nakonec zbývalo vytvořit aplikaci zajišťující vlastní programování EEPROM pamětí.
Rozhodl jsem se pro toto řešení: Na PC vygenerovaný HEX soubor programu je do EEPROGu odesílán po sériové lince. Pro toto lze využít širokou škálu terminálových aplikací, nebo odeslat soubor přímo z příkazové řádky operačního systému příkazem COPY. Já jsem využíval již zmíněný Serial.

Parametry přenosu jsou:

- Rychlost 2400Baud
- Formát dat 8bit
- 1 stop bit, 1 start bit
- Parita není
- Řízení toku dat není

Parita při přenosu není použita. Každý záznam HEX souboru je již zabezpečen kontrolním součtem (CHKSUM). Do EEPROM zapisovaná data jsou následně kontrolována zpětným čtením.

Aplikace programátoru z postupně přicházejících dat dekóduje vlastní program a zajistí jeho uložení v EEPROM paměti.
O výsledku programování či případných chybách (chybný kontrolní součet, chyba ve struktuře HEX souboru) je uživatel informován výpisem na terminálu.

Programátor má i explicitní funkci verifikace dat. Je-li první znak HEX souboru vykřičník (!) namísto standardní dvojtečky (:), přijímaná data nejsou do EEPROM zapisována, ale je prováděno jejich porovnání s obsahem EEPROM.

Naprogramování programu o velikosti cca 10kB trvá zhruba 3minuty. Rychlost by však bylo možné zvýšit. Například posílat do programátoru přímo BIN soubor programu(oproti HEX formátu je úspornější) a zapisovat data do EEPROM po celých stránkách. Po této úpravě by naprogramování původních 10kB programu trvalo cca 10 vteřin.
Další variantou by bylo posílat data po USB a dosáhnout časů ještě nižších.

Korektnost zapsaného programu lze otestovat i jeho přímým spuštěním v programátoru.

Dokončením tohoto programu jsem završil vývoj programátoru EEPROG a tím i první části vývoje hardwaru diplomové práce.

7.2. Oživení PBSL

7.2.1. Porucha procesoru C31

Oživení zařízení PBSL jsem se rozhodl provést stejným způsobem, jako v případě zařízení EEprog.

Nejdříve jsem osadil stabilizátory napětí a ověřil jejich funkčnost. Poté jsem s průběžnou kontrolou odběru osadil všechny pasivní součástky a integrované obvody v pouzdrech pro povrchovou montáž – tedy DPC31 a FT245BM.
Naprogramoval a osadil jsem obvod GAL pro výběr pamětí a do paměti EEPROM nahrál jednoduchý program. Ten měl blikáním diody potvrdit funkčnost obvodů GAL, DPC31 i vlastní EEPROM v jednotce PBSL. Jeho funkci jsem zkontroloval spuštěním přímo v programátoru.

V PBSL se program ale nevykonával.

Osciloskopem jsem tedy zkontroloval běh oscilátoru a funkci fázového závěsu procesoru. Pracovali korektně. Následovala dlouhá řada experimentů a dalších měření.

Nepřijatelné interference mezi jednotlivými vodiči sběrnic pamětí se podařilo eliminovat odstraněním pull-up rezistorů.
Má-li být tedy DPC31 provozován s interním procesorem, zřejmě nesmí být žádné pull-up rezistory na sběrnicích zapojeny. Tuto domněnku jsem si ale ani dodatečně v dokumentaci [6] k obvodu nepotvrdil.

Nakonec jsem došel k závěru, že procesor je z blíže neurčeného důvodu poškozen a není schopen správně pracovat.
Po poradě s konzultantem Ing. Petrem Smolíkem jsem se rozhodl postavit nový exemplář zařízení.

7.2.2. Oživení druhé verze

Při osazování a oživování druhé verze PBSL jsem postupoval obdobně jako v případě verze první. Jen pull-up rezistory na paměťových sběrnicích jsem v plošném spoji neosadil.
Také funkci procesoru jsem zkoušel stejným programem.
Procesor napoprvé pracoval správně.

Pokusil jsem se ještě vzájemným porovnáním měření na obou deskách zjistit důvod nefunkčnosti prvního exempláře PBSL. Shledával jsem ale parametry obou desek jako stejné. Z tohoto důvodu jsem definitivně usoudil na vnitřní poškození prvního procesoru.

Postupně jsem ověřil funkci RS232, USB i displeje. Využíval jsem k tomu aplikací vytvořených již dříve při oživování EEprogu.
Problémy jsem měl jen s ovládáním portů PE až PH obvodu DPC31 po jednotlivých bitech. Pro každý bit je vyhrazen v interní datové paměti jeden byte. Zápis hodnoty 0FFh způsobí nastavení

příslušného pinu do logické jedničky, jakákoliv jiná hodnota znamená stav logické nuly. V dokumentaci k obvodu toto není uvedeno.

Funkci paměti dat SRAM jsem úspěšně ověřil postupným zápisem a čtením všech paměťových buněk.

7.2.3. Ověření funkce zařízení na PROFIBUSu

Potom jsem se seznámil s knihovnou funkcí V1SL dodávanou k obvodu DPC31. Čerpal jsem z [20],[21] a [22].

Na základě nabytých znalostí jsem vytvořil program realizující připojení k PROFIBUSU.

Jednotku PBSL jsem připojil jako další jednotku do výukového systému PROFIBUS Demonstrátor v laboratoři katedry a pokusil se ověřit funkci jednotky.
Pro analýzu chování jednotky na sběrnici jsem použil PC s PROFIBUSkomunikační kartou a software PROTest.

Při oživování rozhraní RS485 jsem zjistil, že DC/DC měnič galvanického oddělení nevykazuje definované parametry. Hodnota výstupního napětí byla mimo toleranci. Nejde o ojedinělou závadu konkrétního obvodu, zvýšené výstupní napětí na cca 5,5V jsem naměřil i na měniči osazeném v jiné aplikaci.
Dále jsem musel drátovými propojkami opravit prohození datových vodičů PROFIBUS sběrnice na výstupním konektoru.

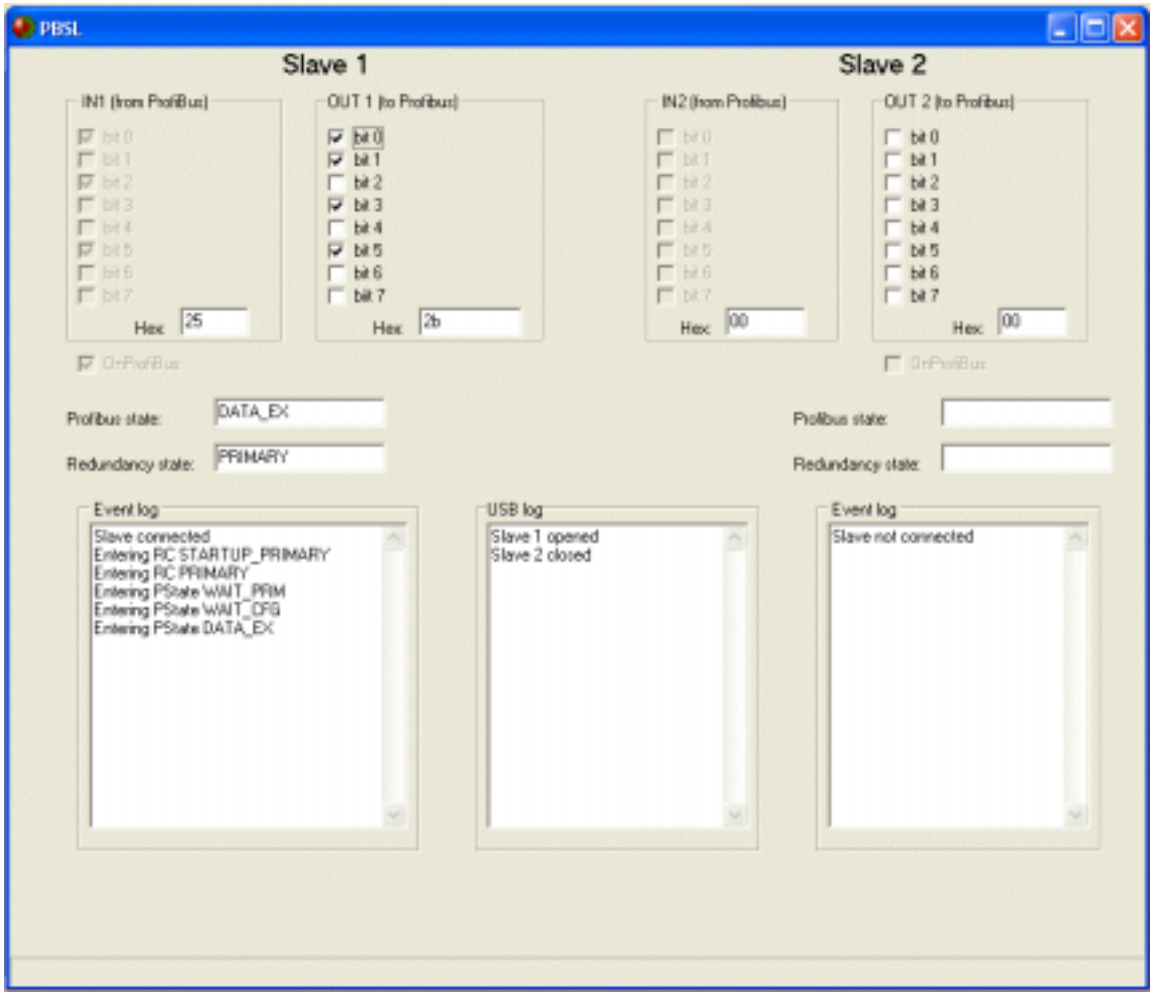
Nakonec se mi zařízení povedlo zprovoznit a tato jednotka byla jednotkou Master přiřazena na tzv. LiveList zařízení.

Tím se mi podařilo prokázat plnou funkčnost jednotky v systému PROFIBUS a ukončil jsem tím proces jejího oživování.

8. Aplikace na PC

Aplikace na PC zajišťuje komunikaci s PBSL po USB. Naprogramoval jsem ji v C++ Builderu verze 5.0. V tomto směru se ukázala volba obvodu FT245BM od FTDI jako správná. Dostupnost ovladačů významně vývoj této aplikace zjednodušila .
Aplikace PBSLmonitor plní jednak funkci vizualizační – je zobrazován stav připojeného slave a stav výstupů, tak ovládací.

Aplikace PCSlave zajišťuje obousměrnou komunikaci se zařízením PBSL. Umožňuje připojit až dvě tyto jednotky. Je koncipována jako virtuální jednotka 8 logických vstupů a výstupů.



Obr. 15 Okno aplikace PBSL

Po spuštění program automaticky detekuje připojená USB zařízení a naváže s nimi komunikaci. Ta probíhá podle tohoto protokolu:

Stanice PBSL cyklicky vysílá rámec dat do PC a naopak. Rámec má délku 6 byte a tuto strukturu:

Byte	0	1	3	4	5
Význam	StartByte	PROFIBUS status	RedCom status	OutHi4	OutLo4

Každý rámec začíná tzv. StartByte. Jeho hodnota je unikátní a zajišťuje v proudu dat synchronizaci.
Dále je vyhrazen 1byte PROFIBUSstatus pro indikaci stavu PBSL z hlediska jeho statusu na PROFIBUSu, 1 byte RedComStatus pro indikaci stavu z hlediska Redundance.
Pro přenos 8bitů hodnoty logických výstupů jsou vyhrazeny 2 byte. Přenos v 1 byte není možná, protože by mohl být porušen požadavek na jedinečnost unikátnost výskytu hodnoty vyhrazené pro StartByte. Nejdříve jsou tedy přeneseny horní 4 a poté dolní 4bity dat.

Tento je nezbytný pro nalezení začátku bloku dat.

Aplikace vysílá pro PBSL 1 byte dat, který reprezentuje hodnotu vstupů. Odeslání dat je provedeno s každým přijatým rámcem ze strany PBSL, nebo každou 1s. Stavy bufferů koncových zařízení jsou monitorovány.

Aplikace automaticky detekuje odpojení i opětovné připojení koncového zařízení na USB portu bez potřeby znovuspuštění. Správně detekuje i výpadky a chyby při přenosu.

9. Závěr

V průběhu diplomové práce jsem se seznámil se sběrnicí PROFIBUS a USB v rozsahu nutném pro návrh a konstrukci zařízení PROFIBUS DP slave a zařízení jsem fyzicky realizoval. Funkci jednotky jsem úspěšně ověřil a potvrdil tak správnost návrhu.

Vytvořené zařízení je schopno plnit úlohu 8 logických vstupů a 8 logických výstupů a je plně slučitelné s ostatními zařízeními PROFIBUS. Implementovaná rozhraní RS232 a USB navíc umožňují jeho připojení k PC. Linka RS 232 může též plnit úlohu komunikačního kanálu RedCom mezi dvěma jednotkami slave a lze implementovat rozšíření PROFIBUSu o takzvanou slave redundanci. Chování redundantních slave je věnována samostatná kapitola v teoretické části práce.

Navrhl a sestrojil jsem i další zařízení – programátor EEPROM. Tato jednotka byla navíc též vybavena rozhraním RS232 a USB a sloužila jako podpůrný vývojový nástroj. Vyvedením vybraných vývodů řídícího procesoru na konektory připadá v úvahu i jiné alternativní využití tohoto obvodu.

Při vývoji a návrhu zařízení jsem se seznámil s příslušnou součástkovou základnou a vývojovými nástroji nutnými pro realizaci projektu od vypracování návrhu zapojení až po uvedení prototypu do výroby.

Osvojil jsem si programování na nejnižší úrovni HW v případě programování řídících mikroprocesorů v jazyku assembler a C. Využití prostředí Borland C++Builder při programování aplikace na PC bylo naopak zkušeností s vyšším programovacím jazykem.

Značné úsilí jsem byl nucen vynaložit při oživování systému. Potýkal jsem se s nezdokumentovaným chováním obvodu DPC31 jenž byl jádrem celé práce. Protože jsem byl nucen konstatovat poškození tohoto prvku, plně oživit se povedlo až nově vyrobený exemplář jednotky PBSL.

Zařízení je navrženo jako značně univerzální. Uvítal bych, kdyby v jeho vývoji bylo i nadále pokračováno a nalezlo uplatnění v laboratořích katedry kybernetiky.

10. Literatura

[1] *PROFIBUS Specification*, EN 50 170, 1996.

[2] Popp, M. *The Rapid Way to Profibus-DP*. PROFIBUS Nutzerorganisation, Karlsruhe, 1997.

[3] *PROFIBUS – Specification Slave Redundancy*. PROFIBUS Nutzerorganisation, Karlsruhe, 2002.

[4] *PROFIBUS Technical Description*. PROFIBUS Nutzerorganisation, Karlsruhe, 1999.

[5] Líčko, M. *Průmyslová sběrnice Profibus*. Diplomová práce. Praha: České vysoké učení technické, elektrotechnická fakulta, katedra kybernetiky, 2000.

[6] *DPC31 – Hardware description*. Verze 1.1. SIMATIC NET, 2002. <**www.sea.siemens.com**>

[7] *FT245BM USB FIFO*. Verze 1.0. Future Technology Devices International, 2002. <**www.ftdichip.com**>

[8] *FT245BM Designers Guide*. Verze 1.1. Future Technology Devices International, 2002. <**www.ftdichip.com**>

[9] *FTD2XX Programmer's Guide*. Verze 2.01. Future Technology Devices International, 2002. <**www.ftdichip.com**>

[10] *FTD2XXST User's Guide*. Verze 1.0. Future Technology Devices International, 2002. <**www.ftdichip.com**>

[11] *Universal Serial Bus Specification Revision 2.0*. USB Implementers Forum, 2000. <**www.usb.org**>

[12] Bartosiński, R. *Implementace USB Interface pro počítačové periferie*. Diplomová práce. Praha: České vysoké učení technické, elektrotechnická fakulta, katedra kybernetiky, 2003.

[13] Katalog GM Electronic 2000. GM electronic, Praha, 2000. 460 s.

[14] Záhlava, V. *Metodika návrhu plošných spojů*. Vydavatelství ČVUT, Praha, 2000.

[15] Podlešák, J. *Přístrojové aplikace mikroprocesorů*. Vydavatelství ČVUT, Praha, 1999.

[16] *AT89S8252*. Katalogový list. Atmel Corporation, 2000. <**www.atmel.com**>

[17] Fuksa, M. *Alespoň něco o 8051*...[online]. <**www.volny.cz/fuksam**>

[18] *HD44780 Dot Matrix Liquid Crystal Display Controller*. Hitachi, 1999.

[19] *Cx51 Compiler User's Guide*. Keil software, 2000.

[20] Smolík, P. *Průmyslová sběrnice Profibus DP Extended*. Diplomová práce. Praha: České vysoké učení technické, elektrotechnická fakulta, katedra kybernetiky, 2001.

[21] *V1SL user description - Firmware for Siemens ASIC DPC31 DPV1*. Verze 1.3. SIMATIC NET, 2001. <**www.sea.siemens.com**>

[22] *V1SL getting started - DPC31 Slave Firmware*. Verze 1.1. SIMATIC NET, 2001. <**www.sea.siemens.com**>

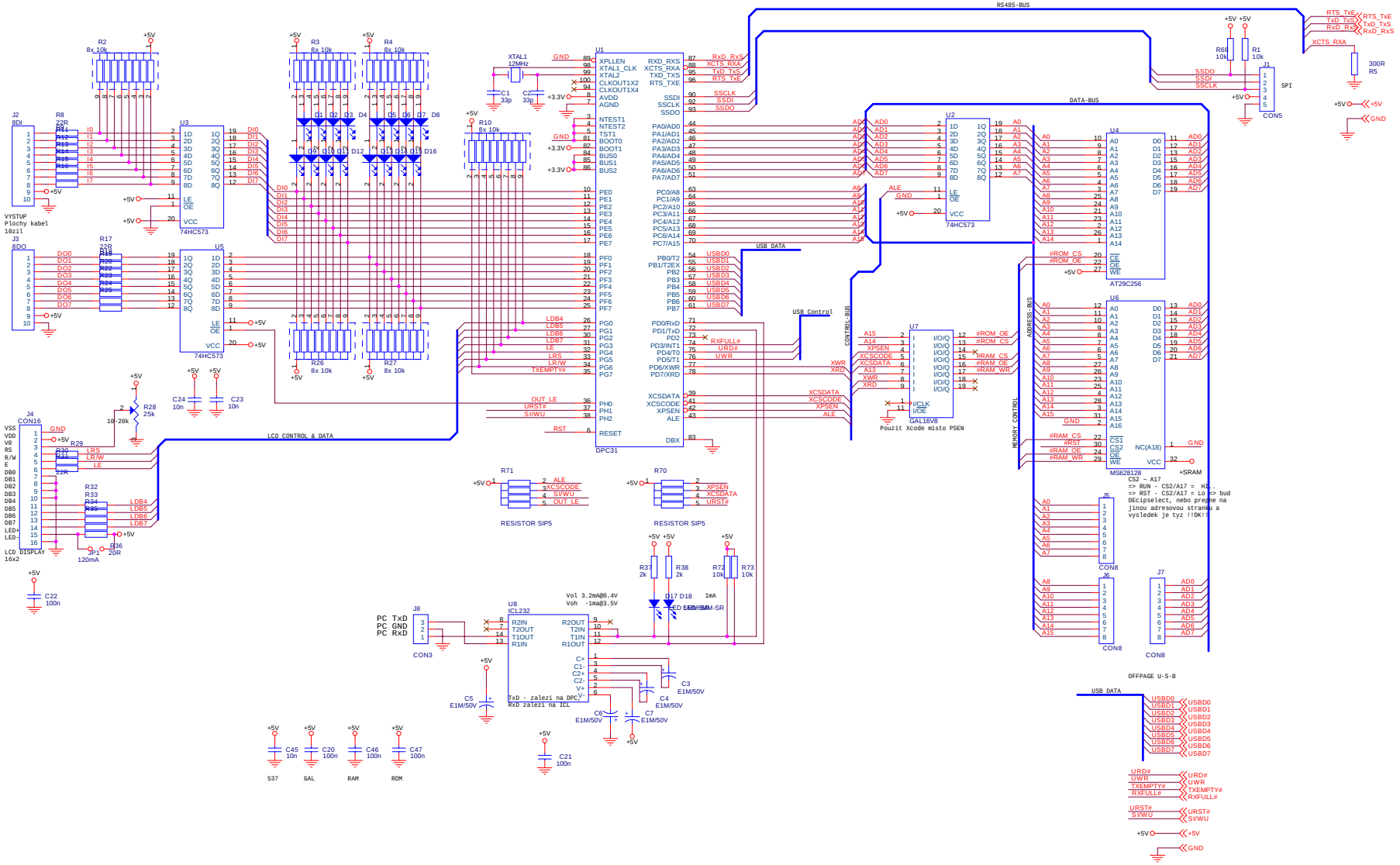
[23] *Device Description Data Files - GSD*. Verze 2.2. Siemens, 2003

11. Obsah přiloženého CD

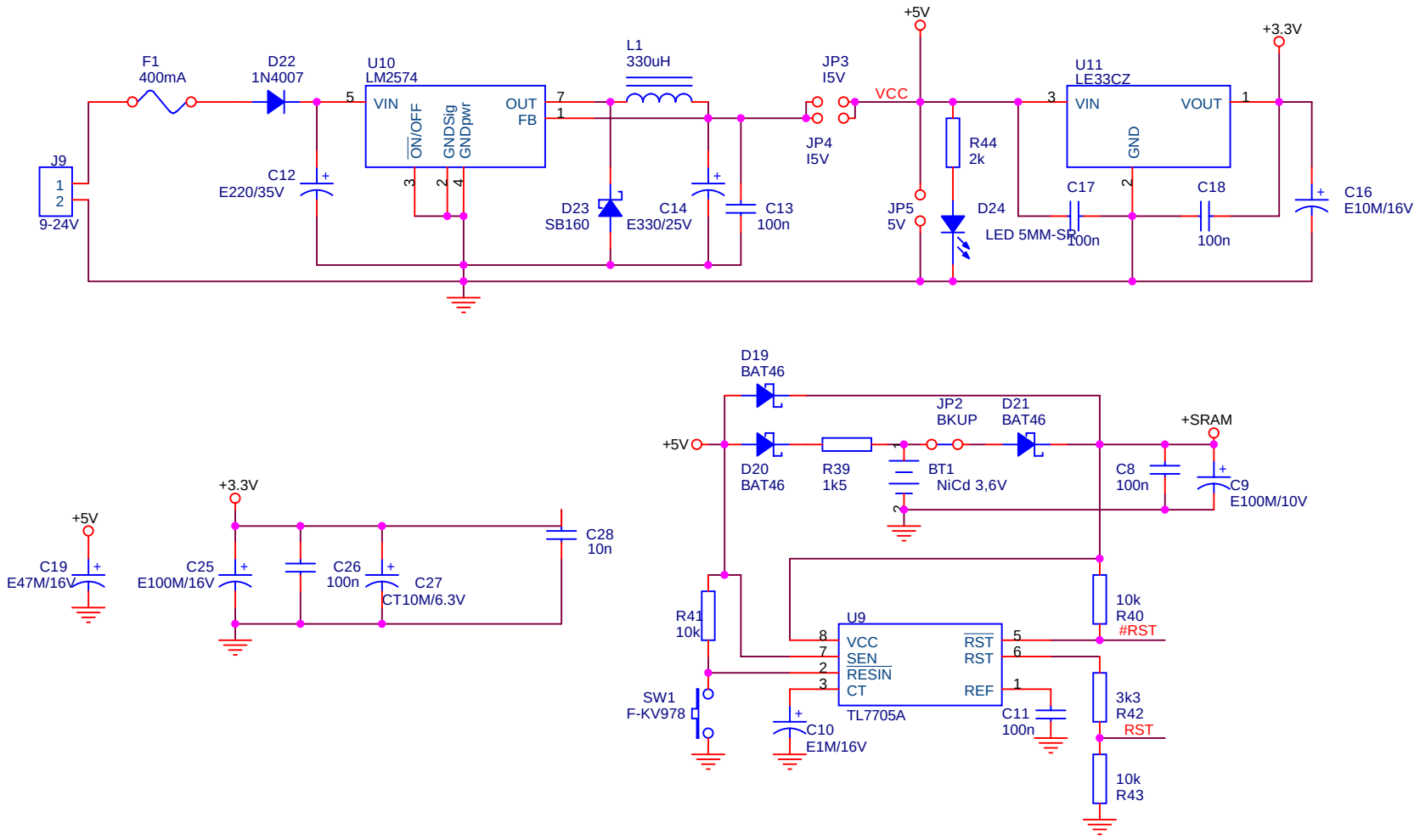
Adresář	Obsah adresáře
\DataSheet	Katalogové listy použitých součástek
\Foto	Fotografie hotových plošných spojů
\GAL	Podklady pro výrobu programovatelného pole GAL
\Manuals	Část dokumentace v elektronické podobě (použitá literatura)
\ORCAD	Schémata navržených plošných spojů
\Software_52	Programové vybavení procesorů použitých na plošných spojích
\Software_PC	Programové vybavení PC
.\sources	Zdrojové kódy v C++Builderu
.\PBSL	Hlavní aplikace komunikující prostřednictvím navrženého zařízení se sběrnicí PROFIBUS
.\USBloop	Aplikace pro testování USB komunikace PC s navrženým zařízením
.\USBspeed	Aplikace pro měření přenosové rychlosti při komunikaci navrženého zařízení s PC přes USB
.\binaries	Zkompilované spustitelné soubory
.\drivers	Ovladače pro USB komunikaci PC s plošnými spoji
\Text_DP	Text diplomové práce v elektronické podobě
\Tools	Používaný volně šířitelný software

12. Přílohy

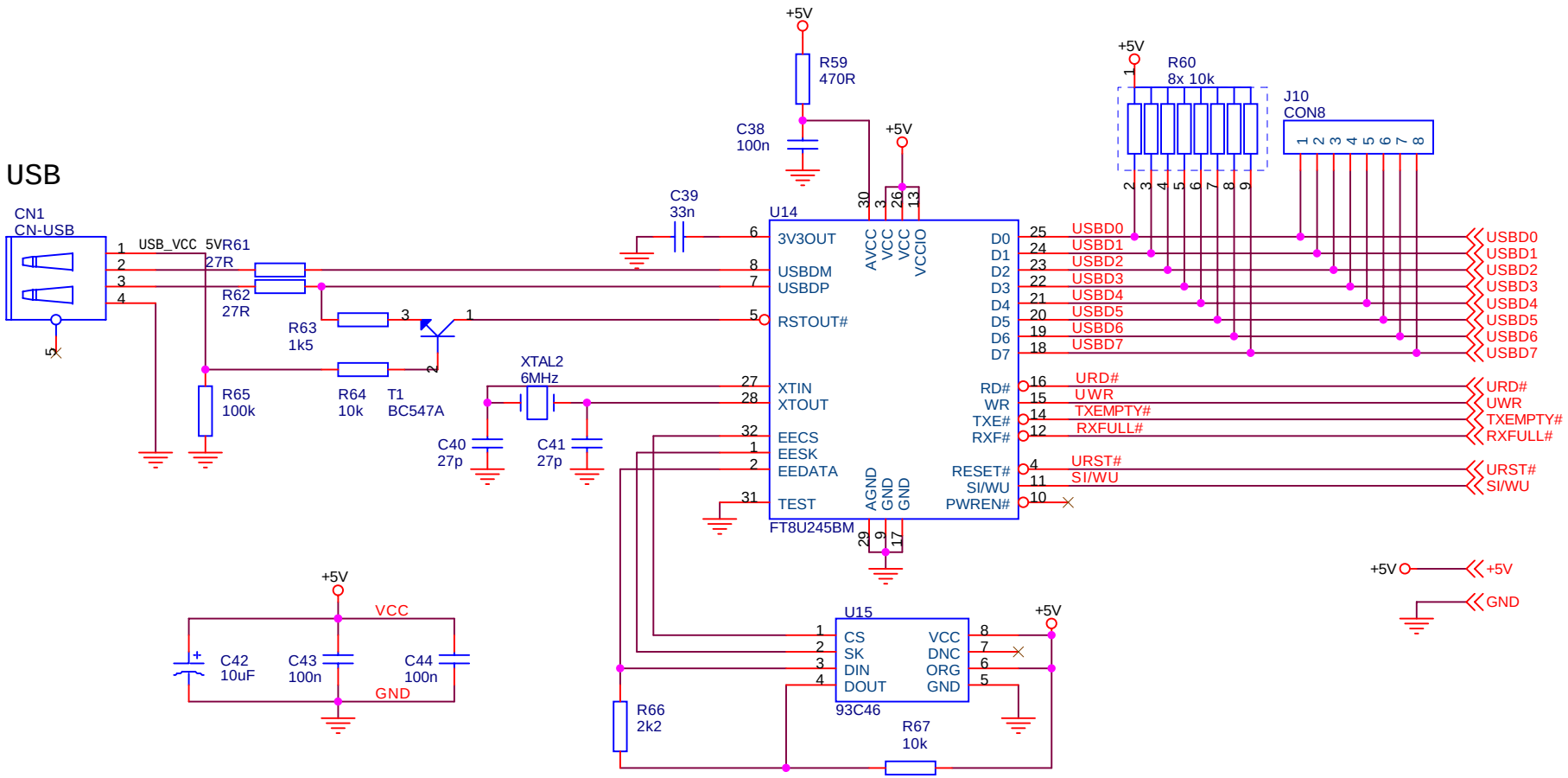
I. Schéma zapojení PROFIBUS Slave PBSL
II. Schéma zapojení programátoru EEPROG
III. PLD soubor programovalelného pole GAL
IV. GSD soubor RDSL
V. Fotografie zařízení



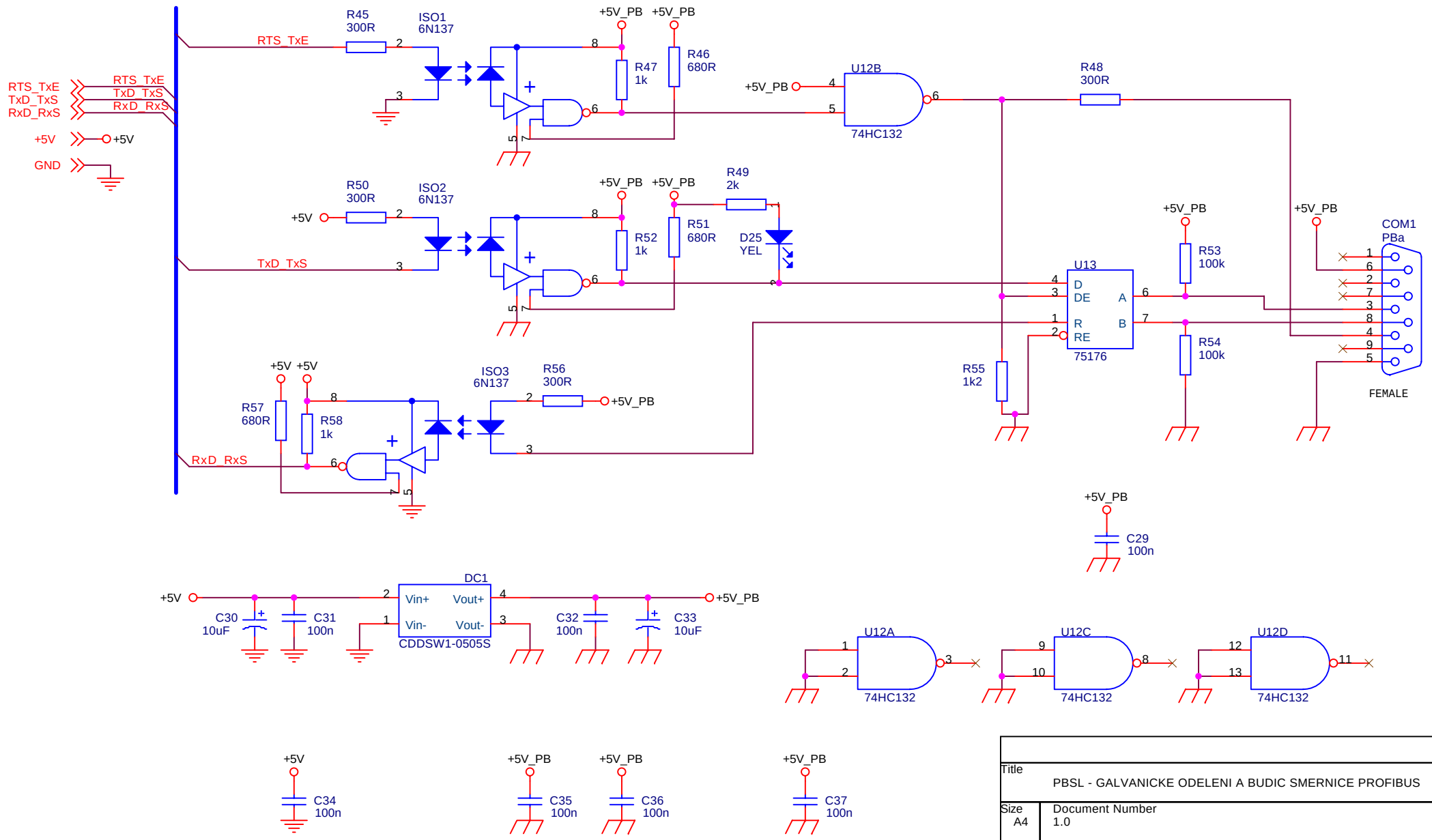
Title	DPC31
Size A2	Document Number 1.0
Rev	1.0
Date:	Tuesday, January 27, 2004
Sheet	1 of 4



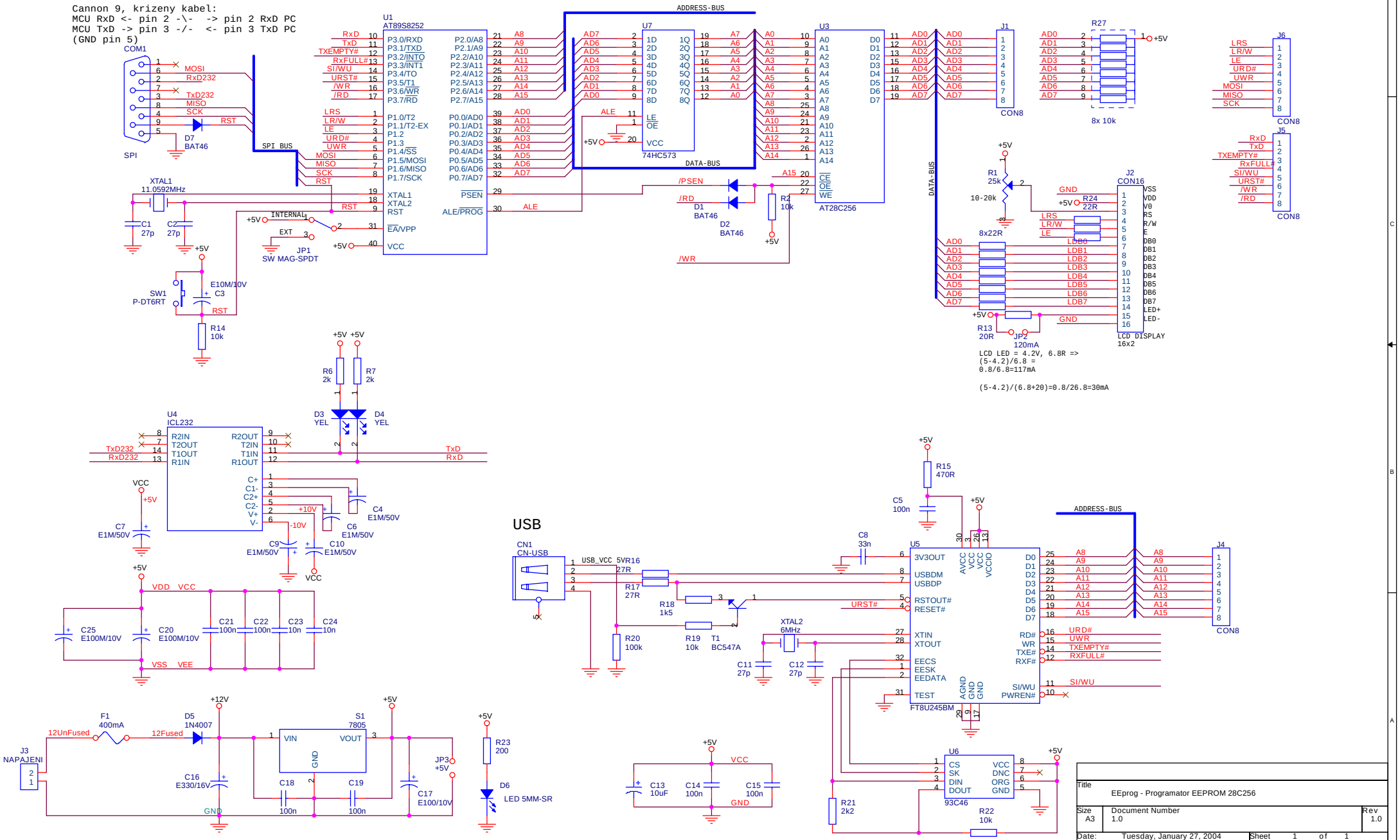
Title	PBSL - Zdrojova cast	
Size A4	Document Number 1,0	Rev 1,0
Date: Tuesday, January 27, 2004	Sheet 2 of 4	



Title	PBSL - USB rozhrani	
Size A4	Document Number 1,0	Rev 1,0
Date: Tuesday, January 27, 2004	Sheet 3 of 4	



Title	PBSL - GALVANICKE ODELENI A BUDIC SMERNICE PROFIBUS	
Size A4	Document Number 1.0	Rev 1.0
Date: Tuesday, January 27, 2004	Sheet 4 of 3	


Cannon 9, krizeny kabel:
MCU RxD <- pin 2 -\- -> pin 2 RxD PC
MCU TxD -> pin 3 -/- <- pin 3 TxD PC
(GND pin 5)
COM1
SPI
D7
BAT46
SPI BUS
U1
AT89S8252
XTAL1
11.0592MHz
C1
27p
C2
27p
INTERNAL
EXT
JP1
SW MAG-SPDT
SW1
P-DT6RT
E10M/10V
C3
R14
10k
ADDRESS-BUS
U7
74HC573
DATA-BUS
U3
AT28C256
D1
BAT46
D2
BAT46
R2
10k
J1
CON8
R27
8x 10k
J6
CON8
J5
CON8
R1
25k
10-20k
R24
22R
8x22R
J2
CON16
LCD DISPLAY
16x2
R13
20R
JP2
120mA
LCD LED = 4.2V, 6.8R =>
(5-4.2)/6.8 =
0.8/6.8=117mA
(5-4.2)/(6.8+20)=0.8/26.8=30mA
R6
2k
R7
2k
D3
YEL
D4
YEL
U4
ICL232
C4
E1M/50V
C6
E1M/50V
C7
E1M/50V
C9
E1M/50V
C10
E1M/50V
C25
E100M/10V
C20
E100M/10V
C21
100n
C22
100n
C23
10n
C24
10n
USB
CN1
CN-USB
R16
27R
R17
27R
R18
1k5
R19
10k
T1
BC547A
R20
100k
R15
470R
C5
100n
C8
33n
U5
FT8U245BM
XTAL2
6MHz
C11
27p
C12
27p
J4
CON8
U6
93C46
R21
2k2
R22
10k
C13
10uF
C14
100n
C15
100n
J3
NAPAJENI
F1
400mA
D5
1N4007
S1
7805
C16
E330/16V
C18
100n
C19
100n
C17
E100/10V
JP3
R23
200
D6
LED 5MM-SR
Title
EEprog - Programator EEPROM 28C256
Size
A3
Document Number
1.0
Rev
1.0
Date:
Tuesday, January 27, 2004
Sheet
1
of
1

Příloha III.

|GAL16V8
definice typu obvodu

Typ vyvodu (In/Out) nastaven automaticky dle cisla vyvodu.
Prirazeni signalu:
|2:A15,
|3:A14,
|4:XPSEN,
|5:XCSCODE,
|6:XCSDATA,
|7:A13,
|8:XWR,
|9:XRD,
|11:OE,
^vstupni signaly piny 2..9, 1~hodiny, 11~ /OE

XRD	cteni z externi pameti DATA
XWR	zapis do externi pameti DATA
XPSEN	cteni z programove pameti (eXternal Program ENabled)
XCSCODE	vyber externi pameti CODE
XCSDATA	vyber externi pameti DATA
A[13..15]	adresove bity A13..A15
OE	OutputEnable GALu

|12:ROM_OE,
|13:ROM_CS,
|15:RAM_CS,
|16:RAM_OE,
|17:RAM_WR
^vystupni signaly piny 12..19

ROM_CS	vyber pameti FLASH
ROM_OE	OutputEnable (cteni) pameti FLASH
RAM_CS	vyber pameti SRAM
RAM_OE	OutputEnable (cteni) pameti SRAM
RAM_WR	zapis do pameti SRAM

|Title: "Vyberova logika FLASH/SRAM pro RSDL...CODE-FLASH 0~8kB, CODE-SRAM 8~64kB, DATA-SRAM 2kB~64kB"

ROM_CS = A15 # A14 # A13 # XCSCODE
nastaven pri pristupu do XCODE na adresy <= 8kB
(jakakoliv "I" ve vyrazu zabrani CS)
|ROM_CS = XCSCODE

|ROM_OE = XPSEN
nastaven pri pozadavku na cteni programu

RAM_CS = XCSDATA & ((A15 # A14 # A13) & XCSCODE')'
nasteven pri pozadavku na cteni dat z XDATA nebo
cteni programu z adres >8kB
xcsdata=0 => cs
((A15 # A14 # A13) & XCSCODE')' musi byt 0
((A15 # A14 # A13) & XCSCODE') musi byt 1
  (A15 # A14 # A13) musi byt 1 a XCSCODE musi byt 0
|RAM_CS = XCSDATA

RAM_OE = XRD & XPSEN
nastaven pri pozadavku na cteni dat/programu
RAM_OE = XRD
|RAM_WR = XWR
nastaven pri pozadavku na zapis dat
_____
Test v programu Vectors:
|VECTORS:
|{
_____
_____vyber pameti FLASH_____

|Display XCSCODE,A15, A14, A13, "->", ROM_CS
|Set OE
|Test XCSCODE,A15,A14,A13
_____
____OutputEnable pameti FLASH (cteni)_____
..netestovano..
_____
_____Vyber pameti SRAM_____

|Display XCSCODE,XCSDATA, A15, A14, A13, "->", RAM_CS
|Set OE

|Set XCSCODE,XCSDATA
|Test A15,A14,A13

|Clear XCSCODE
|Test A15,A14,A13

|Set XCSCODE
|Clear XCSDATA
|Test A15,A14,A13
_____
_____OutputEnable pameti SRAM (cteni)_____
..netestovano..
_____
_____Zapis do pameti SRAM_____
..netestovano..
|End
}

Příloha IV.

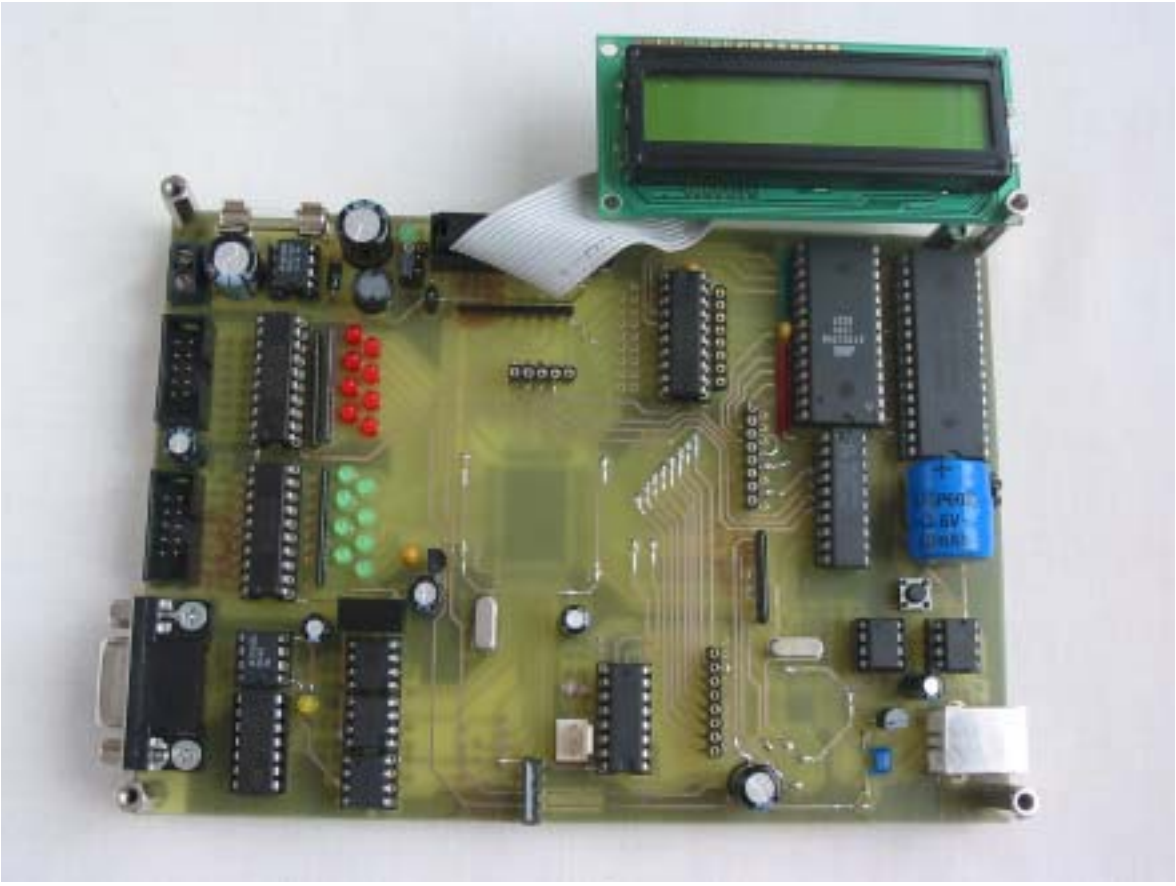
```
;_____________________________
;        GSD soubor pro RDSL
;
;RDSL ~ Redundantni Profibus DP slave
;Profibus DP ASIC DPC31, USB FT245BM, RedCom RS232
;
;
;Autor:Eduard Obornik
;Datum:1.12.2003
;_____________________________

#Profibus_DP

GSD_Revision= 1                ;GSD revize
;
Vendor_Name="Eduard Obornik"
Model_Name  ="RDSL-Redundantni USB DP slave"
Revision        ="V1.0"
Ident_Number=0x1234
Protocol_Ident          =0      ;only DP supported
Station_type    =0      ;DP slave
FMS_supp        =0      ;FMS is not supported
Hardware_Release="HW1.0"
Software_Release="SW1.0"
;
;Bitmap_Device          ="bmpDev.dib"           ;Bitmaps ~ *.DIB, 16colors, 70x40
;Bitmap_Diag ="bmpDiag.dib"
;Bitmap_SF      ="bmpSF.dib"
;
Slave_Family            =3@TdF@K336
;_____________________________
;podporovane prenosove rychlosti
9.6_supp        =1
19.2_supp       =1
93.75_supp      =1
187.5_supp      =1
500_supp        =1
1.5M_supp       =1
3M_supp         =0
6M_supp         =0
12M_supp        =0
;
MaxTsdr_9.6  =60      ;max. response time using
MaxTsdr_19.2            =60     ;different transmition rates
MaxTsdr_93.75           =60
MaxTsdr_187.5           =60
MaxTsdr_500 =100
MaxTsdr_1.5M            =150
```

```
;MaxTsdr_3M=250
;MaxTsdr_6M=450
;MaxTsdr_12M        =800
;_____________________________
;Hw specifikace
Redundancy   =0      ;not supported
Repeater_Ctrl_Sig    =2      ;RTS Signal with TTL level
24V_Pins      =0
Implementation_type ="DPC31"
;_____________________________
;DP specifikace
Freeze_Mode_supp   =0
Sync_Mode_supp     =0
Fail_Safe          =0      ;FailSafeMode not Supported
;
Auto_Baud_supp     =1      ;auto baud rate control
Set_Slave_Add_supp =0      ;nepodporuje
Min_Slave_Interval =100    ;nasobky 100us
;
Max_Diag_Data_Len =7       ;zadna uzivatelska diagnostika
;_____________________________
;Moduly
Modular_Station    =0
Modul_Offset       =1
;
Module             ="Redundantni USB DP slave" 0x3F
EndModule
;
```

Příloha V.



Jednotka PROFIBUS DP slave s rozhraním USB

Programátor EEPROM