VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta elektrotechniky a komunikačních technologií
Ústav automatizace a měřicí techniky

DIPLOMOVÁ PRÁCE

Rozhraní přístroje A2000 ke sběrnici Ethernet

Brno 2002 **Michal Mikl**

P r o h l á š e n í

Prohlašuji, že jsem předloženou diplomovou práci zpracoval sám
s konzultační pomocí vedoucího projektu a uvedených konzultantů.
Použité literární prameny jsou uvedeny v literárních odkazech.

V Brně dne : 22.5.2002 Podpis:

P o d ě k o v á n í

Děkuji vedoucímu diplomové práce panu Doc. Ing. Františkovi Zezulkovi, CSc. za odbornou pomoc při vypracování projektu a poskytnutí odborné literatury a podkladů k mikroprocesoru Rabbit 2000. Dále pak firmě GMC – měřící technika, s.r.o. za poskytnutí podkladů k měřícímu přístroji A2000.

1. ÚVOD

Nabídka firmy GMC – měřící technika, s.r.o obsahuje v současné době i multifunkční wattmetr A2000. Tento měřící přístroj patří do kategorie moderních inteligentních přístrojů obsahujících mikroprocesor pro zpracování dat a komunikaci s okolím.

V dnešní době je komunikace stále se rozvíjející a nezbytnou součástí každodenního života a to i v oblasti měřící a řídící techniky. Rozvoj komunikačních forem pomocí síťových technologií jako je Internet pak přináší nový rozměr všech oblastí využití. V lokálních sítích je dnes nejvíce využívanou architekturou Ethernet. Většina firemních síťových architektur obsahuje brány k sítím Ethernet. Také samostatné přístroje, ale i aktory a čidla jsou již stále častěji vybavovány tímto rozhraním.

Firma GMC také vyvíjí různé způsoby připojení svých přístrojů k síti Internet. Jedním z nich je připojení přes místní síť typu Ethernet, což je tématem a náplní této diplomové práce. Rozhraní mezi sítí Ethernet a měřícím přístrojem A2000 má být realizováno jako samostatná jednotka připojitelná k vlastnímu měřícímu přístroji pomocí již implementovaného rozhraní RS 232C. Jádrem jednotky bude mikroprocesor Rabbit 2000, který bude sloužit ke komunikaci s přístrojem A2000 a komunikaci s uživatelem prostřednictvím http serveru a který umožní spouštění Java Appletu. Ke změně obsahu poskytovaného HTTP serverem bude sloužit FTP server.

Pro potřeby návrhu je k dispozici vývojové prostředí Rabbit TCP/IP Development Kit umožňující práci mikroprocesoru Rabbit připojeného k síti Ethernet. Z tohoto prostředí lze také čerpat motivaci při návrhu, resp. především v části styku mikroprocesoru Rabbit a rozhraní Ethernet.

2. PŘÍSTROJ A2000

obr. 1: Vzhled přístroje A2000

2.1 APLIKACE

A2000 je měřící přístroj určený pro analýzu a monitorování elektrických systémů napájených střídavým napětím. Může pracovat s vnitřními měřícími transformátory ve 3-fázových systémech do napětí 500 V, při použití vnějších měřících transformátorů může pracovat v systémech s větším napětím.

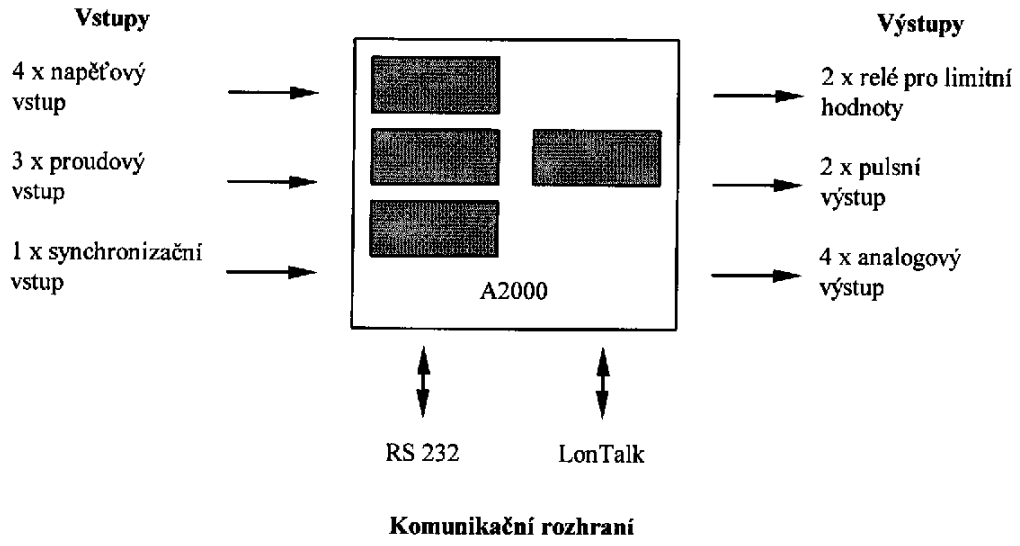
A2000 měří napětí, proud, frekvenci a fázový posun v jedno a 3-fázových systémech. Z těchto údajů počítá činný, jalový a zdánlivý výkon, činnou a jalovou energii a účiník.

Rozsahy vnějších měřících transformátorů mohou být zadány do přístroje, což umožňuje zobrazení primárních údajů přímo na přístroji. Maximální hodnoty mohou být uloženy v paměti pro každou měřenou nebo vypočtenou veličinu. A2000 umožňuje spínat relé při překročení nastavitelných limitních hodnot. Digitální a analogové vstupy slouží k napojení dalších přístrojů jako jsou například elektromechanické čítače energie, zápisníky, řídící smyčky.

Přístroj také může komunikovat s ostatními vnějšími systémy např. pomocí sériového rozhraní RS 232 nebo může být zapojen do sítě LonWorks.

2.2 POPIS

Na následujícím obrázku jsou zobrazena rozhraní přístroje A2000.



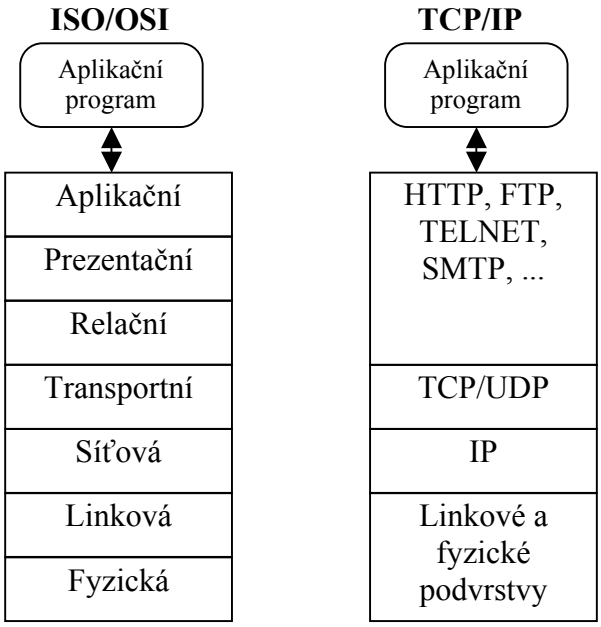
obr. 2: Rozhraní přístroje A2000

- **napěťové vstupy**: každý napěťový vstup má bezpečnou impedanci; měření v třífázových systémech do 500 V je možné bez použití vnějších transformátorů
- **proudové vstupy**: všechny proudové vstupy jsou vzájemně galvanicky odděleny; jestliže jsou použity vnější transformátory, jejich primární a sekundární rozsahy musí být zadány do přístroje, aby zobrazoval správné hodnoty
- **synchronizační vstup** je použit pro výběr intervalu pro výpočet 15 minutových hodnot; synchronizace může být také řízena programově samotným přístrojem
- **výstupní relé** mohou být spínána při překročení nastavené limitní hodnoty pro jakoukoliv měřenou nebo vypočtenou hodnotu
- **impulsní výstupy**: hodnoty měřené činné a jalové energie mohu být vysílány formou impulsů pro připojení elektromechanických počítadel

- **analogové výstupy**: jakákoliv měřené nebo vypočtená hodnota může být vysílána prostřednictvím analogového výstupu; jednotlivé analogové výstupy mohou být konfigurovány jako napěťové nebo proudové
- **sériové rozhraní RS 232** slouží pro přenos naměřených hodnot do osobního počítače a ke konfiguraci přístroje z osobního počítače. Přes toto rozhraní bude přístroj A2000 komunikovat s navrhovanou kartou, která bude dále zprostředkovávat styk s rozhraním Ethernet
- **sériové rozhraní RS 485**
- **rozhraní Lon Talk** je určeno pro připojení do sítě Lon Works

3. PROSTŘEDKY SÍŤOVÉ KOMUNIKACE

Komunikace v síťovém prostředí je komplexní problém, který byl pro jednoduchost rozdělen na několik dílčích problémů. Toto rozdělení vedlo ke vzniku referenčního vrstvového ISO/OSI modelu. Pomocí něj je možné komunikaci rozdělit na několik jednotlivých částí. Existuje řada síťových protokolů, které víceméně z modelu ISO/OSI vychází nebo se s ním srovnávají. V prostředí Internetu je jedním z nejdůležitějších protokol TCP/IP. Jeho srovnání s modelem ISO/OSI je vidět na obrázku č. 4.



Obr. 4: Vrstvový model ISO/OSI a TCP/IP

Srovnáním modelů je vidět, že funkce nejvyšších vrstev zastávají u modelu TCP/IP jednotlivé aplikační protokoly jako je FTP, Telnet apod. Funkci transportní vrstvy odpovídají protokoly TCP a UDP, síťové vrstvě protokol IP. Linková a fyzická vrstva nejsou jednoznačně definovány.

3.1 KOMUNIKAČNÍ SLUŽBY PROTOKOLU TCP/IP

Jak již vyplývá z výše popsaného modelu, TCP/IP není jednoznačně definovaným protokolem, ale spíše skupinou sdružující protokoly komunikačních služeb Internetu. V následující tabulce je uveden přehled nejdůležitějších služeb.

Protokol	Popis
HTTP	World Wide Web
NMTP	Diskusní skupiny
SMTP	Elektronická pošta
POP3	Čtení elektronických zpráv uložených na serveru
FTP	Přenos souborů mezi počítači
TELNET	Vzdálený přístup k počítači

3.2 HTTP PROTOKOL

Tento protokol slouží zejména pro distribuci hypertextových a dalších dokumentů, nebo-li zprostředkování dobře známých www stránek. Využívá portu 80. Je to jednoduchý protokol, který se nestará o to, zda budou data doručena, ale předpokládá, že to co se objeví na jedné straně, dostane se na druhou stranu v nezměněné podobě. Komunikace probíhá formou dotazů, kdy klient vyšle dotaz a server vystaví odpověď. Pro tuto komunikaci definuje protokol HTTP určité metody. Nejznámější metody přenosu požadavků na server jsou POST a GET. Metoda GET zasílá požadavky serveru prostřednictvím URL dotazu, tedy požadovaná data jsou součástí adresy. Velikost těchto dat je omezená. Metoda POST je univerzálnější a nemá omezenou velikost.

3.2.1 CGI skripty

Vhodným prostředkem pro zpracování dat na straně HTTP serveru jsou CGI skripty. Tyto rozšiřují schopnosti serveru o komunikaci s dalšími programy nebo dalším hardwarem. CGI skripty jsou spouštěny zadáním dotazu od klienta včetně parametrů prostřednictvím metod GET a POST. Výstupem CGI skriptu může být např. vytvoření HTML stránky, práce s databází, vykonání jiného programu na serveru atd.

Vlastní CGI skript je program vytvořený ve formě spouštěcího souboru v libovolném programovacím jazyku (např. C/C++).

Při použití metody GET jsou data předávána jako parametry na příkazové řádce. Příklad volání skriptu metodou GET:

GET /cgi-bin/název_skriptu?param1=hodnota1&param2=hodnota2 HTTP/1.0,

kde znak ? je oddělovač mezi názvem skriptu a jeho parametry, znak & je oddělovač jednotlivých parametrů. HTTP/1.0 označuje použitou verzi protokolu.

Při použití metody POST jsou data předána zápisem na standardní výstup klienta, tj. na standardní vstup na straně CGI skriptu. Příklad volání skriptu metodou POST:

POST /cgi-bin/název_skriptu HTTP/1.0
Content-type: typ_přenášeného_obsahu/podtyp_přenášeného_obsahu
Content-length: délka_přenášených_dat_v_bytech
prázdný_řádek
param1=hodnota1&param2=hodnota2&…,

kde proměnná *Content-type* určuje typ přenášeného obsahu (např. text/plain, text/html, image/gif). Proměnná *Content-length* obsahuje velikost přenášených dat. Po těchto informacích následuje prázdný řádek a pak již vlastní data, která mají stejný tvar jako u metody GET .

3.3 FTP PROTOKOL

Jedná se o protokol podporující přenos souborů mezi uzly v síti. Pro počáteční přihlášení a získání oprávnění přístupu na FTP server je třeba zadat uživatelské jméno, které je přenášeno sítí v nezašifrované podobě. FTP využívá spolehlivé transportní služby se spojením protokolu TCP. Číslo portu používané

protokolem FTP je 21. Tento port slouží pro kontrolní příkazy FTP. Pro přenos dat se implicitně využívá port 20.

O správu souborů v daném uzlu sítě a autentizaci přihlášení se potom stará program zvaný FTP server. Ten podporuje dle implementace jak anonymní přihlášení (většinou přihlašovací jméno *anonymous*, *host* nebo *quest*), tak i přihlášení s vyžádáním uživatelského jména a hesla. Po přihlášení lze využívat základní příkazy, které umožňují výpis souborů a adresářové struktury, upload nebo download souborů a jejich mazání. Pro každý soubor nebo adresář lze většinou odděleně nastavit úroveň práv a vlastníka.

Pro přístup na FTP server slouží program zvaný FTP klient, který je spouštěn na klientské stanici. Ten překládá uživatelské příkazy do požadavků odesílaných na server a zprostředkovává příjem dat ze serveru.

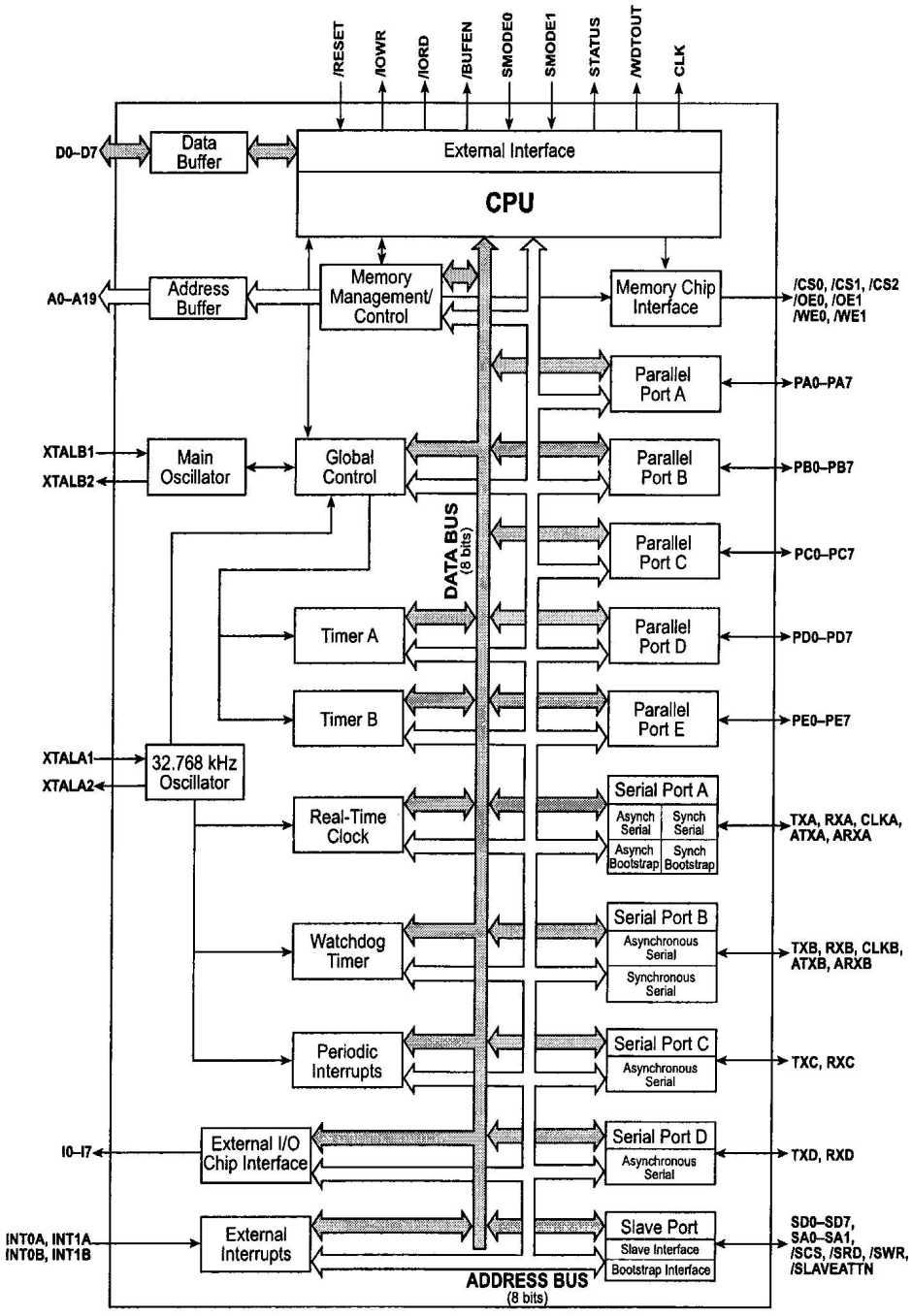
4. RABBIT 2000

4.1 ÚVOD

Mikroprocesor Rabbit 2000 (dále jen Rabbit) vychází z koncepcí mikroprocesorů Z80, Z180 a HD64180 při použití v malých řídících aplikacích. Rabbit má podobnou architekturu a vysoký stupeň kompatibility s těmito mikroprocesory, ale je značně vylepšen.

4.2 OBECNÁ SPECIFIKACE

Rabbit je usazen ve 100 vývodovém pouzdru PQFP určeném pro povrchovou montáž na desku plošného spoje. Pracovní napětí je od 2,7 V do 5 V. Hodinový kmitočet může být až 30 MHz. Jeho datová sběrnice je 8-bitová, stejně jako celá koncepce mikroprocesoru. Instrukce pracující s adresou mají 16-bitový operand a tento se převádí v paměťové mapovací jednotce na 20-bitovou adresu pro vlastní paměťový čip. Tím mikroprocesor dokáže adresovat až 1MB paměťového prostoru. Má 40 paralelních vstupně/výstupních linek (sdílených se sériovými porty), které jsou rozčleněny do bran PA až PE. Jednotlivé sériové či paralelní porty mohou pracovat v různých módech činnosti. Bránu PA lze například konfigurovat jako „Slave port“ pro komunikaci s dalším procesorem (v tomto zapojení jsou využity ještě některé bity z dalších bran). Mikroprocesor má čtyři úrovně priority přerušení a rychlou odezvu na přerušovací signál pro praktické aplikace. Umožňuje programování přes PC pomocí vývojového systému jazyka C (Dynamic C). Do paměti o velikosti 1 MB je možné v jazyku C uložit až 50 000 řádků kódu. Přístup k vstupně/výstupním zařízením je řešen použitím instrukcí přístupu do paměti s I/O prefixem.



obr. 3: Vnitřní struktura mikroprocesoru Rabbit 2000

4.3 VSTUPNĚ VÝSTUPNÍ ROZHRANÍ MIKROPROCESORU

4.3.1 Paralelní porty

Rabbit má pět osmi-bitových paralelních portů označených jako A, B, C, D a E. Vývody použité pro paralelní porty jsou často sdílené s dalšími vstupně výstupními funkcemi mikroprocesoru viz [1] . Nejdůležitější vlastnosti jsou zde shrnuty.

Port A - je sdílen se „Slave data interface“ portem

Port B - je sdílen s řídícími signály pro Slave port a vývodem hodinového kmitočtu pro použití sériového portu A či B jako synchronního

Port C - sdílen s vývody sériových portů A až D

Port D - vyšší čtyři bity sdíleny s alternativními vývody pro sériový port A a B. Nižší čtyři bity nejsou sdíleny. Má výstupní registry časově řízené umožňující generování pulsů.

Port E - všechny bity portu E mohou být konfigurovány jako vstupně výstupní strobované vývody. Čtyři bity mohou být použity jako vstupy externích přerušení. Jeden bit je sdílen se signálem „Slave port chip select“. Dále obsahuje port E také výstupní registry kaskádně řazené a časově řízené umožňující generování pulsů.

4.3.2 Sériové porty

Mikroprocesor obsahuje na čipu integrované čtyři sériové porty označené A, B, C a D sdílené s vývody paralelního portu C. Všechny umožňují vysokorychlostní asynchronní komunikaci. Navíc mohou být porty A a B jako synchronní. Pro tento případ jsou vyvedeny hodinové impulsy CLKA a CLKB sloužící k jejich časování. Porty A a B mají ještě další rozšiřující možnost a to přepnutí mezi standardními a alternativními vývody (sdíleny s paralelním portem D). Port A má speciální možnost být použit ke „studenému“ startu systému. Provádí se přes něj nahrávání programu do paměti a následné ladění.

4.4 NÁVRHOVÉ STANDARDY

K dosažení stejné funkčnosti při práci s mikroprocesorem Rabbit může být použito několika způsobů. Připravením a publikováním návrhových standardů firmou Rabbit Semiconductor je umožněna snadnější hardwarová a softwarová podpora.

4.4.1 Programovací port

Jedná se o specifikaci propojení Rabbita kabelem se sériovým portem PC sloužící jako standardní programovací interface. Ve specifikaci je uvedeno zapojení tohoto kabelu a rozhraní RS232/TTL logiky a toto propojovací rozhraní bývá součástí jednotlivých „kitů“ nabízených výrobcem. Také standardní návrhové rozhraní Dynamic C distribuované s mikroprocesorem předpokládá toto komunikační rozhraní k nahrávání a ladění programů. Fyzicky je na straně Rabbita programovací port připojen k sériovému portu A, startovacím vývodům, signálu reset a programovatelnému výstupnímu vývodu k posílání signálů do PC.

4.4.2 BIOS

Firma Rabbit Semiconductor poskytuje standardní BIOS pro mikroprocesor Rabbit. BIOS je softwarový program, který zajišťuje spuštění a ukončení činnosti Rabbita a základní služby pro běh softwaru na Rabbitu.

4.5 PROGRAMOVÉ VYBAVENÍ

4.5.1 Dynamic C

Dynamic C je interaktivní vývojový systém odvozený od jazyka C. Je určen pro operační systémy Windows 9x a NT. Nabízí kombinaci překladače, editoru a ladícího prostředí. Nahrávání a ladění softwaru pro Rabbit se děje přes programovací kabel připojený na programovací port mikroprocesoru. Jednotlivé knihovny Dynamic C nabízejí výkonné softwarové prostředky k ovládání Rabbita. Tyto zahrnují ovladače, utility, matematické rutiny a odlaďovací BIOS

pro Dynamic C. Dále zahrnují některé verze Dynamic C pro Rabbit upravený operační systém reálného času uC/OS-II.

4.5.2 Základní skupiny funkcí

Dynamic C obsahuje množství základních knihoven funkcí. Některé knihovny nesou stejné označení jako knihovny běžných implementací jazyka C pro PC a obsahují také úplně či téměř stejné funkce. Nachází se zde ale také velké množství funkcí, které jsou určeny specificky pro mikroprocesor Rabbit a využívají plně jeho možností. Funkce lze rozčlenit do skupin dle jejich použití. Jedná se o funkce aritmetické, manipulující s bity, pro práci se znaky, rozšířenou pamětí, pro rychlou FFT transformaci, přerušení, multitasking a mnoho dalších.

4.5.3 Rozšíření funkcí – protokol TCP/IP

Jazyk Dynamic C má implementovány ještě další skupiny funkcí, které ovšem nejsou zahrnuty do standardních, protože k jejich uplatnění je potřeba širší hardwarový základ než jen samotný mikroprocesor s nejnutnějšími obvody. Tyto rozšiřující možnosti využijeme například při práci s rozhraním Ethernet, kdy přichází v úvahu funkce pro práci s protokolem TCP/IP. K tomuto účelu je ovšem nutné mít Rabbit připojený k ethernetovému řadiči přes výrobcem definované vstupy. Takovéto zapojení je použito i ve vývojové sadě „TCP/IP Development Kit“ a použito pro tuto práci. Jedině za takovýchto předpokladů je možné použít jednotlivé funkce v nezměněné podobě. Při zapojení Ethernetového řadiče na jiné vývody mikroprocesoru by bylo nutné vytvořit vlastní funkce například tím, že se předefinují odkazy na použité propojovací vývody.

Funkce a konstanty pro práci s protokolem TCP/IP se dají opět přehledně shrnout do skupin dle významových kategorií. Jedná se o konfigurace, inicializace, nastavení bufferů TCP/IP protokolů, přenos dat, standardní vstupně výstupní funkce, datové konverze a další.

HTTP server (také web server) slouží ke zpřístupnění dokumentů HTML a dalších různých dokumentů klientům pomocí prohlížeče webu. Tento server je implementován v knihovně HTTP.LIB. Knihovna obsahuje čtyři datové struktury.

- HttpSpec – obsahuje všechny soubory, proměnné a funkce, které zpřístupňují web server
- HttpType – provádí asociace přípon souborů s typem MIME (Multipurpose Internet Mail Extension) a funkcí, která ovládá typ MIME
- HttpRealm – stará se o uživatelská ID a hesla zpřístupňující určité chráněné oblasti na serveru
- HttpState – použití této struktury je nutné pro CGI funkce

CGI (Common Gateway Interface) slouží jako standardní rozhraní mezi externími aplikacemi a HTTP serverem. Pokaždé, když si klient vyžádá URL odpovídající CGI programu, server tento CGI program vykoná v reálném čase. V programu pro Rabbit v Dynamic C jsou CGI funkce normálními funkcemi v jazyce C na něž je proveden odkaz přes odpovídající nadefinování struktur HTTP serveru. HTTP server potom zprostředkovává napojení na tyto funkce (jejich vykonání) při požadavku klienta (např. volba odkazu z prohlížené www stránky).

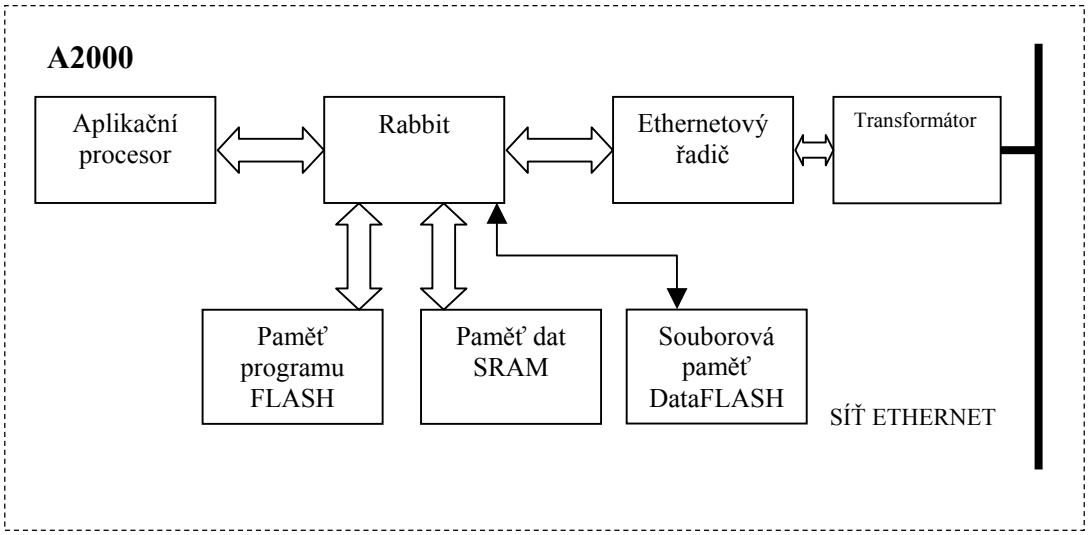
FTP (file transef protocol) server implementuje funkce pro přenos souborů. K tomuto účelu používá dvou TCP/IP spojení. Na otevřeném portu 21 FTP server pasivně očekává připojení klienta. Pro Dynamic C implementovaná verze FTP serveru podporuje také anonymní přihlášení.

Dynamic C také implementuje knihovnu (ZSERVER.LIB), která obsahuje struktury, funkce a konstanty umožňující FTP serveru a HTTP serveru sdílet data a uživatelskou autentifikaci. Obsahuje následující struktury:

- ServerSpec - umožňuje přístup FTP serveru k seznamu funkcí, souborů a proměnných
- ServerAuth - definuje globální pole se seznamem uživatelských jmen a hesel
- FormVar – reprezentuje proměnné v HTML formuláři

5. APLIKAČNÍ ROZHRANÍ

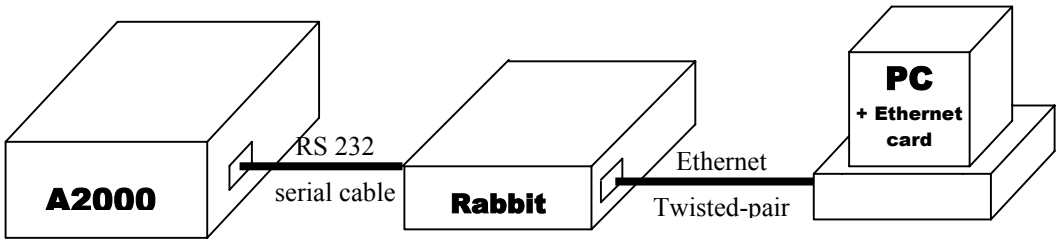
5.1 ZAPOJENÍ ROZHRANÍ ETHERNET



obr. 5: Blokové schéma rozhraní

Návrh rozhraní vychází z využití stávajících možností přístroje A2000. Ten má již zaimplementováno sériové rozhraní a rozhraní Lon Talk. Mikroprocesor Rabbit slouží k zajištění komunikace s aplikačním procesorem přístoje A2000. Dále na něm běží programové vybavení vytvářející HTTP server a FTP server pro přístup ze sítě Ethernet. Protože Rabbit nebsahuje žádnou vnitřní paměť, je k němu nutné připojit paměť programu typu FLASH, paměť dat typu SRAM a volitelně sériovou paměť DataFLASH sloužící k uložení souborů pro HTTP a FTP server. Jako rozhraní mezi sítí Ethernet a Rabbitem slouží ethernetový řadič Realtek RTL8019AS.

Celé zapojení by mělo být realizované jako samostatná „rozšiřující karta“, respektive modul, na jedné straně připojitelný k měřícímu přístroji přes jeho stávající rozhraní a na druhé straně připojitelný na ethernetové rozhraní pomocí standardu 10Base-T (twisted-pair). Názorný náhled je vidět na obrázku č. 6. Na obrázku č. 5 je principiální blokové schéma komunikace.

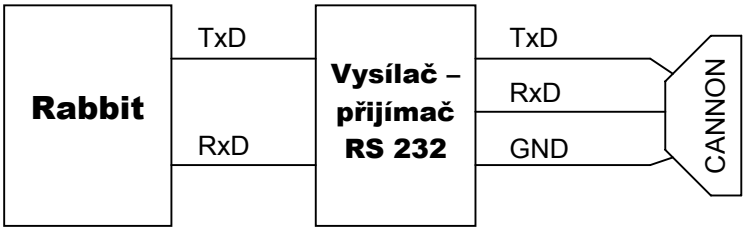


obr. 6: Náhled spojení A2000 – Rabbit - Uživatel

5.2 PŘENOS DAT

Mezi aplikačním procesorem a mikroprocesorem Rabbit 2000 je realizován přenos dat po sériovém asynchronním rozhraní. Spojení je provedeno třívodičově s ukončením pomocí devítipinových konektorů Canon. Formát přenášených dat je v běžném tvaru a to jeden start bit, osm datových bitů, jeden paritní bit a jeden stop bit. Přenosová rychlost je u přístroje A2000 volitelná z ovládacího panelu. V programu pro Rabbit bude nastavená po inicializaci na 19200 bps, ovšem uživatel ji může později změnit pomocí Java Appletu.

Ze strany mikroprocesoru Rabbit je třeba připojit tyto vývody na piny sériového portu. Jelikož sériový port A je použit pro nahrávání programu do paměti z PC, je nutné zvolit některý ze zbývajících portů. Pro realizaci byl zvolen port D. Jeho signály TxD (vysílací signál) a RxD (přijímací signál) jsou na vývodech 59 a 60 a jsou tak sdíleny s vývody paralelního portu C. Tyto signály mají úroveň TTL logiky a je tedy potřeba provést přizpůsobení a oddělení signálu. To se provede pomocí vysílacího/přijímacího obvodu k rozhraní RS 232.



obr. 7: Zapojení sériového rozhraní

5.3 TYPY PŘENÁŠENÝCH ZPRÁV

Při komunikaci rozlišuje přístroj A2000 několik typů přenášených zpráv. Jsou to zkrácená, řídící a plná zpráva. Následuje přehled tvarů těchto zpráv. Podrobné informace o přesném nastavení a významu jednotlivých znaků uvnitř konkrétní zprávy je možné nalézt ve firemní literatuře [5] a základní přehled v tomto projektu použitých zpráv v další podkapitole „Obsah komunikačních zpráv“.

5.3.1 Zkrácená zpráva

Je posílána v dotazovacím směru (z masteru) jako příkazy pro instrumenty (např. reset) a žádost o důležitá data z instrumentů. Také může být poslána ve směru odpovědi (z A2000) jako potvrzení příkazu, který nevyžadoval žádná data v odpovědi. Má následující strukturu:

Č. znaku	Obsah	Význam	Poznámka
1	10h	Start zprávy	jen pro zkrácenou zprávu
2	0..FAh,FFh	Adresa instrumentu (IA)	
3		Funkční pole (FF)	
4		Kontrolní součet (CS)	součet IA a FF
5	16h	Konec zprávy	stejné pro všechny typy zpráv

5.3.2 Řídící zpráva

Je přenášena pouze z masteru do přístroje A2000. Bývá u všech žádostí o data, která nemohou být dotazována ve zkrácené zprávě, protože potřebují hlubší specifikaci. Má následující tvar:

Č. znaku	Obsah	Význam	Poznámka
1	68h	Start zprávy	
2	03h	Délka	počet znaků od IA do CS
3	03h	Délka (opětovně)	
4	68h	Start zprávy (opětovně)	
5	0..FAh,FFh	Adresa instrumentu (IA)	
6		Funkční pole (FF)	
7		Index parametru (PI)	
8		Kontrolní součet (CS)	
9	16h	Konec zprávy	stejné pro všechny typy zpráv

5.3.3 Plná zpráva

Tento typ je použit přístrojem A2000 k přenosu příkazů a parametrů do instrumentu a k nahrání dat z instrumentu. Má následující tvar:

Č. znaku	Obsah	Význam	Poznámka
1	68h	Start zprávy	
2		Délka (DL)	počet znaků od IA do CS
3		Délka (opětovně)	
4	68h	Start zprávy (opětovně)	
5	0..FAh,FFh	Adresa instrumentu (IA)	
6		Funkční pole (FF)	
7		Index parametru (PI)	
…		Blok dat (DB)	
délka + 5		Kontrolní součet (CS)	
délka + 6	16h	Konec zprávy	stejné pro všechny typy zpráv

5.4 OBSAH KOMUNIKAČNÍCH ZPRÁV

5.4.1 Reset Instrument

Slouží k inicializaci. Přístroj provede reset a přepne se do výchozího nastavení. Také smaže některá statistická data jako maxima měřených veličin.

Zpráva z mastera (zkrácená zpráva):

10h	IA	09h	CS	16h

Odpověď z A2000: Bez odpovědi

5.4.2 Instrument OK?

Kontroluje připravenost a správnou funkci přístroje. Dle významu příchozího „FF“ lze zjistit stav přístroje.

Zpráva z mastera (zkrácená zpráva):

10h	IA	29h	CS	16h

Odpověď z A2000 (zkrácená zpráva):

10h	IA	FF	CS	16h

5.4.3 Request Data from A2000

Veškerá data z přístroje A2000 se přenáší tímto typem zpráv. Jedná se o všechny měřené veličiny, parametry, konfiguraci, podmínky, identifikaci instrumentu atd. Potřebný význam zprávy je určen hodnotou bajtu PI (parameter index).

Zpráva z mastera (řídící zpráva):

68h	DL	DL	68h	IA	89h	PI	CS	16h

Odpověď z A2000 (plná zpráva):

68h	DL	DL	68h	IA	FF	PI	Data	…	Data	CS	16h

5.4.4 Transmit Data to the A2000

Všechny parametry, konfigurace a operační podmínky, které mohou být změněny operátorem, se posílají těmito zprávami. Z příchozího „FF“ lze zjistit, zda byla příchozí data správně zpracována.

Zpráva z mastera (plná zpráva):

68h	DL	DL	68h	IA	69h	PI	Data	…	Data	CS	16h

Odpověď z A2000 (zkrácená zpráva):

10h	IA	FF	CS	16h

5.5 PŘEHLED VÝZNAMŮ „PI“

Měřené veličiny – posílají se vždy po skupinách

PI	Skupina veličin
00h	Fázová napětí
01h	Sdružená napětí
02h	Fázové proudy
03h	Průměrné fázové proudy
04h	Činný výkon
05h	Jalový výkon
06h	Zdánlivý výkon
07h	Účiník
08h	Energie
09h	Intervalová energie (P int)
0Ah	Intervalová energie (Q int)
0Bh	Intervalová energie (S int)
0Fh	Síťová frekvence

Konfigurace relé, pulsních a analogových výstupů

PI	Popis konfigurace	Formát	Jednotka	Rozsah hodnot
10h	Relé 1 hystereze	16 bitů	dle „zdroje“	0..100
	Relé 2 hystereze	16 bitů	dle „zdroje“	0..100
	Relé 1 limit	±15 bitů	dle „zdroje“	-1999..1999
	Relé 2 limit	±15 bitů	dle „zdroje“	-1999..1999
11h	Relé 1 zdroj	8 bitů		viz. [5]
	Relé 2 zdroj	8 bitů		viz. [5]
	Relé 1 konfigurace	8 bitů		viz. [5]
	Relé 2 konfigurace	8 bitů		viz. [5]
12h	Pulsní výstup 1	16 bitů	1/kWh (MWh)	0..5000
	Pulsní výstup 2	16 bitů	1/kWh (MWh)	0..5000
13h	Zdroj pro pulsní výstup 1	8 bitů		viz. [5]
	Zdroj pro pulsní výstup 2	8 bitů		viz. [5]
14h	Analogový výstup 1 – dolní limit	±15 bitů	dle „zdroje“	-9999..9999
	Analogový výstup 2 – dolní limit	±15 bitů	dle „zdroje“	-9999..9999
	Analogový výstup 3 – dolní limit	±15 bitů	dle „zdroje“	-9999..9999
	Analogový výstup 4 – dolní limit	±15 bitů	dle „zdroje“	-9999..9999
15h	Analogový výstup 1 – horní limit	±15 bitů	dle „zdroje“	-9999..9999
	Analogový výstup 2 – horní limit	±15 bitů	dle „zdroje“	-9999..9999
	Analogový výstup 3 – horní limit	±15 bitů	dle „zdroje“	-9999..9999
	Analogový výstup 4 – horní limit	±15 bitů	dle „zdroje“	-9999..9999
16h	Analogový výstup 1 - zdroj	8 bitů		viz. [5]
	Analogový výstup 2 - zdroj	8 bitů		viz. [5]

	Analogový výstup 3 - zdroj	8 bitů		viz. [5]
	Analogový výstup 4 - zdroj	8 bitů		viz. [5]
	Analogový výstup 1 - konfigurace	8 bitů		viz. [5]
	Analogový výstup 2 - konfigurace	8 bitů		viz. [5]
	Analogový výstup 3 - konfigurace	8 bitů		viz. [5]
	Analogový výstup 4 - konfigurace	8 bitů		viz. [5]

Kontrola stavu přístroje a řídící příkazy

PI	**Popis**	**Formát**	**Rozsah hodnot**
20h	stav nastavení přístroje	16 bitů	viz. [5]
21h	chybová hlášení přístroje	2 x 16 bitů	viz. [5]
24h	nulování max. hodnot U a I	2 x 8 bitů	viz. [5]
25h	nulování max. hodnot výkonů	3 x 8 bitů	viz. [5]
26h	nulování všech hodnot energie	16 bitů	= 55AAh
27h	obnova defaultních parametrů	16 bitů	= A965h
29h	Data-Logger start/stop	8 bitů	= 55h: Stop = AAh: Start

Specifikace instrumentu

PI	**Popis**	**Formát**	**Rozsah hodnot**
32h	Převodní poměr U na V	±7 bitů	
	Převodní poměr I na A	±7 bitů	
	Převodní poměr P na W	±7 bitů	
	Převodní poměr E na Wh	±7 bitů	
33h	Typ připojení 3L/4L	8 bitů	55h, AAh, 33h
34h	Synchronizační interval pro energii	8 bitů	0, 1..60
3Bh	Rozsah napětí U_{tprim}	100V/16 bitů	1..7500
	Rozsah napětí U_{tsek}	1V/16 bitů	100..500
3Ch	Rozsah proudu I_{tprim}	1A, 5A/16 bitů	0, 1..30000
	Rozsah proudu I_{tsek}	16 bitů	0, 1
3Fh	Jas displeje	8 bitů	0..7

Hodiny reálného času, Data-Logger

PI	**Popis**	**Formát**	**Rozsah hodnot**
90h	Sekundy	8 bitů	0..59
	Minuty	8 bitů	0..59
	Hodiny	8 bitů	0..23
91h	Den	8 bitů	1..31
	Měsíc	8 bitů	1..12
	Rok	8 bitů	0..99

	Milénium	8 bitů	19..20
92h	Data-Logger – vzorkovací interval	8 bitů	0..13
	Data-Logger – čas záznamu	8 bitů	8..21
	Data-Logger – specifikace spouštění	8 bitů	00h..3Fh
	Data-Logger – kanál 1	8 bitů	00h..B3h
	Data-Logger – kanál 2	8 bitů	00h..B3h
	Data-Logger – kanál 3	8 bitů	00h..B3h
	Data-Logger – kanál 4	8 bitů	00h..B3h
	Data-Logger – kanál 5	8 bitů	00h..B3h
	Data-Logger – kanál 6	8 bitů	00h..B3h
	Data-Logger – kanál 7	8 bitů	00h..B3h
	Data-Logger – kanál 8	8 bitů	00h..B3h
	Data-Logger – kanál 9	8 bitů	00h..B3h
	Data-Logger – kanál 10	8 bitů	00h..B3h
	Data-Logger – kanál 11	8 bitů	00h..B3h
	Data-Logger – kanál 12	8 bitů	00h..B3h

6. NÁVRH SCHÉMATU A DESKY PLOŠNÝCH SPOJŮ

6.1 VÝCHOZÍ PŘEDPOKLADY

Pro návrh zapojení bylo jako výchozí materiál použito schéma zapojení mikroprocesoru Rabbit 2000 v prostředí TCP/IP Development Kit. Z tohoto schématu bylo převzato uspořádání součástek na jednotlivých listech, vzhled schematických značek součástek, které nejsou dostupné ve standardních knihovnách kreslících programů, způsob připojení pamětí a zapojení ethernetového rozhraní přes ethernetový řadič RTL8019AS.

Návrh schématu byl proveden v programu Eagle 4.0. Pasivní součástky byly použity ze stávajících knihoven programu, přidané číslicové součástky byly překresleny dle rozměrů a zapojení skutečných součástek, které budou při realizaci použity do nově vytvořené knihovny Rabbit.lbr.

6.2 POPIS ČÁSTÍ SCHÉMATU

6.2.1 Zapojení mikroprocesoru

Mikroprocesor Rabbit 2000 je SMD součástka umístěná v pouzdru PQFP100. Adresová a datová sběrnice jsou připojeny k pamětem a k ethernetovému řadiči. Signály CS0\, CS1\, OE0\, OE1\, WE0\ a WE1\ jsou zapojeny k pamětem pro řízení výběru obvodu. Z vstupně/výstupních bran jsou využity PB, PC a PE. Brána PA a PD jsou zcela nevyužity. Na vývody XTALA1, XTALA2, XTALB1 a XTALB2 jsou zapojeny obvody oscilátorů. Mezi XTALA1 a XTALA2 je zapojen krystal s rezonančním kmitočtem 32,768 kHz. Mezi vývody XTALB1 a XTALB2 je možné připojit buď krystal 18,432 MHz nebo aktivní oscilační součástku o kmitočtu 30 MHz. Nevyužitá část tohoto obvodu se neosadí součástkami. Ještě jsou vyvedeny napájecí vstupy a vstupy /RESET, STATUS, SMODE0 a SMODE1 pro programovací port.

6.2.2 Paměť programu a dat

K mikroprocesoru Rabbit 2000 jsou připojeny tři číslicové obvody sloužící jako paměť programu a paměť dat.

Jako paměť programu je navržen obvod AT29F040 firmy ATMEL, což je FLASH 512K x 8 s přístupovou dobou 90 ns uložená v pouzdru PLCC32. Obvod bude usazen v patici PLCC32Z. Struktura zapojení vývodů umožňuje poměrně jednoduchou výměnu na obdobné obvod jiných firem se stejnou velikostí paměti nebo na obvody s menší velikostí paměti jako 29C020-12JC (256Kx8), 29C010-12JC (128Kx8) a další. Nutnou podmínkou pro správnou činnost mikroprocesoru je připojení paměti se sektory o maximální velikosti 4096 bajtů. To z důvodů způsobu programování a odlaďování pomocí prostředí Dynamic C a následných možných úprav obsahu paměti FLASH za běhu programu. Výpis kompatibilních typů pamětí je možné nalézt ve firemní literatuře, viz. [3].

Na patici paměti FLASH jsou přivedeny adresové signály A0 – A18. Při použití paměti s menším adresovým prostorem jsou přebytečné vývody nezapojeny. Pro volbu obvodu jsou z mikroprocesoru přivedeny signály CS0\, OE0\ a WE0\ na odpovídající vývody paměti.

Jako operační paměť (paměť dat) je navržen obvod HY628100, což je paměť typu SRAM o velikosti 128 kB od firmy Hyundai v pouzdru SOP 32L. Paměť je z mikroprocesoru aktivována signály CS1\, OE1\ a WE1\. Opět je možné použít kompatibilní typ jiného výrobce.

6.2.3 Sériová datová FLASH (paměť souborů)

Dalším paměťovým obvodem připojeným k mikroprocesoru je sériová datová FLASH firmy Atmel AT45DBxxx sloužící zejména k uložení Java Appletu. Toto řešení bylo zvoleno z důvodů podstatného rozšíření Appletu během návrhu a také velmi špatné dostupnosti navržené paměti programu.

Konkrétně navržen je obvod AT45DB041, což je paměť velikosti 512 KB, ale je možné použít i jiné obvody vývodově a funkčně kompatibilní. Navržený obvod je použit v provedení pouzdra 8-SOIC. Sériová paměť FLASH komunikuje

s mikroprocesorem pomocí synchronního sériového portu B. Je připojena pomocí signálů vstupu SI (na Rabbitu připojen k PC4), výstupu SO (PC5), hodinových pulsů (PB0), volby obvodu /CS (PB7) a napájecích signálů. Obvody této řady od firmy Atmel jsou napájeny napětím 3,3 V (rozsah 2,7 V až 3,6 V), proto je potřeba zajistit snížení napětí. K tomuto účelu je před vlastní paměť zařazen stabilizátor napětí LE33CD (obvod v pouzdru SO-8), který zajišťuje na výstupu stabilizované napětí 3,3 V. Pro správnou funkci jsou na vstup i výstup stabilizátoru připojeny kondenzátory. K opětovnému převodu výstupního signálu na úroveň 5 V je použit unipolární tranzistor jehož pracovní bod je nastaven dvěma rezistory. Výstup obvodu je třeba vzhledem k zapojení v mikroprocesoru softwarově invertovat. Vstupní vývody jsou k paměti zapojeny bez napěťové konverze, jelikož obvod toleruje vstupní napětí o velikosti 5 V. Celé zapojení sériové paměti s příslušejícími obvody je realizováno na samostatné desce plošných spojů s vývody na dvouřadý konektor PINHEAD s roztečí 2,54 mm. Obdobně jsou připojeny i signály na základní desce s mikroprocesorem.

6.2.4 Testovací vývody

Testovací vývody byly zařazeny z důvodů snadnějšího oživování. Jedná se o jednořadé konektory PINHEAD s roztečí 2,54 mm, k nimž se může připojit sonda logického analyzátoru nebo osciloskopu. Na tyto konektory jsou vyvedeny důležité řídící signály z Rabbita, datové signály a některé adresové signály.

6.2.5 Rozhraní k A2000

S přístrojem A2000 komunikuje Rabbit prostřednictvím sériové linky RS232. K oddělení a přizpůsobení úrovní signálů mezi logikou mikroprocesoru a přenosovými impulsy pro RS232 slouží obvod MAX232. To je integrovaný obvod v pouzdru DIL16. K němu jsou pro správnou funkci zapojeny ještě kondenzátory. Pro snadné spojení, například pomocí klasického sériového kabelu (LapLink), je jako konektor použit CANON – vidlice do plošného spoje s devíti vývody.

6.2.6 Rozhraní Ethernet

K rozhraní Ethernet je Rabbit připojen přes ethernetový řadič Realtek RTL8019AS. To je SMD součástka se 100 vývody v pouzdru QFP100. Ze strany mikroprocesoru jsou k ní přivedeny adresové (A0 – A4) a datové signály a dále signály PE0, PE1, PE6 a PE7. Toto zapojení podporují knihovny s funkcemi pro TCP-IP protokol a HTTP server. Obvod je taktován oscilátorem Q1 s kmitočtem 20 MHz a tento je s rezistorem R5 a kondenzátory C6 a C7 přiveden na vývody X1 a X2. Z řadiče jsou vyvedeny dvě indikační diody pro signalizaci přenosu dat. Dále je připojena sériová paměť typu EEPROM (obvod 93C46), která slouží k uchování IP adresy. Signály TPIN+,TPIN-,TPOUT+ a TPOUT- jsou přivedeny na ethernetový transformátor FA163079 a z něj na konektor K1, což je zásuvka WEBP8-8 pro osmipinový konektor RJ-45 na kroucený kabel. Jedná se tedy o standardní připojovací rozhraní. U této části obvodu je možné provést náhradu za konektor, který má transformátor integrován v sobě.

6.2.7 Programovací port

Jedná se o zapojení některých vývodů z mikroprocesoru na konektor pro spojení se sériovým portem PC. Tímto se zprostředkovává komunikace mezi prostředím mikroprocesoru a platformou „Dynamic C“ v níž je vytvářen program na PC. Signály jsou vyvedeny na standardní 10-ti vývodový konektor s roztečí 2,54 mm s označením PINHEAD 2x5 a dále na obdobný konektor s roztečí 2 mm. Pomocí tohoto připojení může PC komunikovat s cílovým zařízením (např. nahrát program), provést reset a reboot.

Standardně programovací port využívá sériového portu A. Pokud ovšem potřebujeme využít sériový port A v uživatelské aplikaci, máme možnost využít alternativních vývodů. Vyvedené signály jsou tedy PC7, PB1, /RESET, SMODE0, SMODE1, STATUS a PC6.

6.2.8 Napájení

Na napájecí vstup je třeba přivést stejnosměrné napětí 8,5 V až 30 V. To je stabilizováno na vnitřní napájecí napětí Vcc = 5V pomocí stabilizátoru 7805, diody 1N4007, fóliového kondenzátoru 330 nF, dvou elektrolytických kondenzátorů 100 nF a jednoho keramického kondenzátoru 100 nF. Obvod mikroprocesoru a ethernetového řadiče mají ve schématu vyvedeny napájecí vývody jako samostatné brány umístěné v této části schématu. Pamět SRAM a obvod MAX232 mají také napájecí vývody samostatné, ale jsou umístěny ve schématu v těsné blízkosti těchto součástek. K ostatním součástkám je napájení přivedeno přímo. Ke každému napájecímu vstupu integrovaných obvodů ve schématu se mezi napájecí napětí Vcc a uzemnění GND zařadí kondenzátor 100 nF. Pro indikaci přivedeného napájení Vcc je použita LED dioda.

6.2.9 Resetovací obvod

Tato část schématu má několik funkcí. Základní součástkou je zde integrovaný obvod TL7705A skrze něj jsou zprostředkovány všechny požadavky na reset mikroprocesoru. Dále se zde nachází tlačítko pro uživatelský reset a pasivní součástky pro zajištění funkčnosti integrovaného obvodu. Jako vstupní signál je přiveden požadavek na reset z programovacího konektoru a výstupem je signál /RESET pro Rabbit. Obvod TL7705A má několik funkcí:

- „Power-On Reset Generator“ nebo-li generátor signálu /RESET při náběhu systému. Tato činnost je následovná. Po připojení napájecího napětí se při určité jeho úrovni vystaví žádost o reset. Až dosáhne napájecí napětí další úrovně (v našem případě cca 4,5V) je spuštěno časování a po jeho uplynutí je signál /RESET zrušen. Tím je zaručen počáteční reset obvodu.
- Automatický reset při poklesu napětí. Pokud poklesne napětí pod určitou minimální přípustnou mez, je proveden reset. Pokud se napětí vrátí do přípustných mezí je opět spuštěno časování a až následně zrušen resetovací signál. Tímto se zajistí, aby při nebezpečném napájecím napětí nedocházelo k náhodným jevům v systému.

- Pulsní úprava externích žádostí o reset. Tím se zajistí úprava tvaru signálu, který je přiveden na vstup /RESIN a má se následně promítnout na signál /RESET. Takto je tedy zapojeno resetovací tlačítko a externí reset z programovacího portu.
- Obvod má dále možnost výstupu jak negativní tak positivní úrovně signálu reset. V našem případě je využita jen negativní úroveň, protože ji vyžaduje Rabbit. Také dokáže pracovat v širokém rozsahu napájecího napětí (3,5 až 18 V), při kterém zajistí svoji funkci.

6.3 VÝCHOZÍ PŘEDPOKLADY PRO NÁVRH DPS

Návrh desky plošných spojů je proveden v programu Eagle 4.0 a vychází z již popsaného schématu. V této návrhové fázi nejsou kladeny požadavky na rozměry desky. Tato má sloužit jako prototyp k odladění a ověření funkčnosti jak hardwaru tak i softwaru a poté se předpokládá přepracování do profesionální podoby pro sériovou výrobu (pravděpodobně čtyřvrstvá deska s SMD součástkami).

Pro danou realizaci je tedy použit návrh na oboustrannou desku plošných spojů osazenou jak klasickými součástkami (s vývody skrz desku) tak součástkami SMD což jsou součástky k povrchové montáži na desku plošného spoje.

Pro snadnou vyrobitelnost desky je návrh dále proveden dle následujících požadavků:

- minimální šířka spojů a mezer 0,28 mm
- minimální otvor prokovu (Drill) 0,7 mm
- minimální ploška prokovu (Diameter) 1,4 mm

Dále je třeba brát v úvahu některá další praktická hlediska pro návrh. Například při návrhu byla využita snaha na jednu stranu desky umístit především vertikální spoje a na druhou stranu spoje horizontální. Dále bylo třeba zajistit, aby blokovací kondenzátory byly co nejblíže příslušné napájecí dvojici VCC a GND. Také obvody oscilátorů je třeba pro správnou funkci umístit blízko příslušných součástek (resp. příslušných vývodů) a totéž platí pro jednotlivé součástky těchto obvodů. K dalším hlediskům patří umístění součástek a spojů do mřížky a zalomení

spojovacích cest nejlépe v úhlu 45°. Desce bylo přiděleno jméno „Rabbit TCP/IP board“, které je na ní také vypsáno.

Dle těchto předpokladů byl vytvořen návrh v programu Eagle 4.0. Ze schématu vytvořeného také v programu Eagle byla vygenerována deska s náhledy pouzder a rozmístěním vývodů součástek. Samotné propojení spojovacích cest a konkrétní rozmístění součástek muselo být provedeno ručně.

7. OŽIVOVÁNÍ A TESTOVÁNÍ

7.1 POSTUP PŘI OŽIVOVÁNÍ MIKROPROCESORU

V příručce k mikroprocesoru Rabbit „Designer's Handbook" je popsán doporučený postup při oživování systémů založených na bázi Rabbita, který byl použit i v této práci. Jedná se několik testů a drobných měření, kterými se má návrhář přesvědčit o funkčnosti mikroprocesoru, programovacího portu, paměti RAM atd.

7.1.1 Počáteční kontrola

Následující kontrola mikroprocesoru se provádí při aktivním signálu /RESET, tedy na vstupu resetu do Rabbita musí být cca 0 V.

- S voltmetrem se proměří přítomnost napětí +5V nebo dalších přípustných hodnot operačního napětí na vývodech 3, 28, 553, 78, 92 a 42. Obdobně nulové napětí kontrolujeme na vývodech 2, 27, 39, 52, 77 a 89.
- S osciloskopem ověříme správnou funkci oscilátoru 32,768 kHz na vývodu 41 (XTALA2). Musí být naměřen správný oscilační kmitočet.
- S osciloskopem zkontrolujeme hlavní systémový oscilátor proměřením signálu CLK (vývod 1).

7.1.2 Diagnostický test č. 2

Tento test předpokládá připojení programovacího kabelu na programovací port Rabbita. Potom je postup následující:

1. Aktivovat signál /RESET na nejméně ¼ sekundy a poté jej uvolnit
2. Při studeném startu poslat po sériovém portu A následující trojice znaků:
   80 0E 20 // nastaví vývod STATUS na log. 0
   80 0E 30 // nastaví vývod STATUS na log. 1
   80 0E 20 // nastaví vývod STATUS opět na log. 0
3. Čekat přibližně ¼ sekundy a potom opakovat postup z bodu 1

Během spuštěného testu pozorujeme osciloskopem výsledek. Je možné takto ověřit funkci mikroprocesoru (správné nastavování vývodu STATUS) a především přenos dat po programovacím kabelu. Osciloskop můžeme spouštět náběžnou hranou signálu reset. Potom je možné sledovat přenos dat na vstup RXA procesoru. Každý přenášený bajt obsahuje 8 datových bitů před nimiž je start bit (nízká úroveň signálu) a za nimiž následuje stop bit (vysoká úroveň signálu). Datové bity jsou ve vysoké úrovni pro log. 1 a v nízké úrovni pro logickou 0.

7.1.3 Diagnostický test č. 3

Tento test ověří funkčnost paměti RAM připojené k signálům /CS1, /OE1 a /WE1. Po vyvolání resetu se pošle následující sekvence trojic bajtů. Posláním poslední trojice se aktivuje start programu od adresy 0 v paměti RAM.

```
80 14 05    // vybere paměť RAM jako aktivní
80 09 51    // připraví watchdog k nečinnosti
80 09 54    // přestane používat časovač watchdogu
            // následující sekvence zapisuje program do paměti
00 01 21    // první dva bajty udávají adresu paměti
00 02 01    // a třetí bajt udává data, která se mají do paměti zapsat
00 02 00
00 04 06
00 05 10
00 06 7E
00 07 29
00 08 10
00 09 FC
00 0A C3
00 0B 00
80 24 80    // přeruší bootstrap a spustí program na adrese 0
```

Vlastní program v assembleru má následující tvar:

```
        ld hl,1         ;počáteční nastavení pro adresu
        ld b,16         ;počet cyklů
loop:   ld a, (hl)      ;do akumulátoru načte obsah paměti s adresou v hl
        add hl,hl       ;posune výše adresový signál
        djnz loop       ;opakuje smyčku do vynulování reg. b
        jp 0            ;celý program opakuje od začátku
```

Tímto způsobem můžeme prověřit prvních 16 adresových signálů, ovšem předpokládá se funkčnost všech datových signálů, aby program pracoval správně.

7.2 OŽIVENÍ ETHERNETOVÉHO ŘADIČE

Ke správné činnosti obvodu Realtek RTL8019AS je třeba naplnit připojenou sériovou EEPROM potřebnými údaji nutnými pro základní činnost. V této sériové paměti jsou uloženy některé konfigurační informace použité jako výchozí po resetu řadiče, avšak povinným údajem je MAC adresa řadiče. Tato adresa (MAC = medium access control) je základním předpokladem správné komunikace síťových zařízení. Pro každé zařízení je jedinečná a výskyt dvou stejných adres by způsobil kolizi daných zařízení připojených k síti.

Příslušná MAC adresa byla pro testovací účely okopírována z modulu TCP/IP Development Kit za předpokladu, že pro provoz s připojením do sítě Internet musí být získána konkrétní jedinečná adresa. K vlastnímu uložení do paměti EEPROM byl použit program „eeprom_init.c“ volně šiřitelný na Internetu.

7.3 PROPOJENÍ S PROSTŘEDÍM DYNAMIC C

Po úspěšném provedení základního oživovacího postupu a odstranění nalezených nedostatků v zapojení je dalším krokem ověření propojení desky

s prostředím Dynamic C na PC. Pomocí něj je možné již překládat, nahrávat a odlaďovat rozsáhlejší programy psané v jazyku C (resp. ve variantě tohoto jazyka).

7.3.1 Studený start Rabbita s Dynamic C

Dynamic C se ihned po spuštění pokusí detekovat přítomnost mikroprocesoru Rabbit. Pokud je funkční deska s Rabbitem připojená programovacím kabelem k PC, tak dojde k nahrávání BIOSu do paměti Rabbita. To má několik fází. Nejprve je rychlostí 2400 bps poslán zaváděcí program v bootstrap módu pomocí trojic bajtů. Tento program je po resetu spuštěn a umožní nahrání pilotního BIOSu rychlostí 57 600 bps (tato rychlost je podporována až od verze Dynamic C 7.05). Pilotní BIOS je opět spuštěn a umožní nahrání BIOSu, který je zkompilován prostředím Dynamic C dle nastavení uživatele. Po této fázi obdržíme na PC hlášení o úspěšném zavedení systému do desky s Rabbitem. Následné uživatelské programy jsou nahrávány rychlostí 115 200 bps.

7.3.2 Ověření funkce

Činnost navržené desky byla ověřena pomocí několika různých programů ze sady ukázkových příkladů k prostředí Dynamic C.

Program „ucospong.c“ využívá operační systém reálného času uC/OSII, pomocí něhož definuje několik úloh. Jedna z nich pravidelně vypisuje stav hodin reálného času a další provádějí ukázkové zobrazení pomocí standardního výstupu.

Program „echochr.c“ vrací nazpět znaky přijaté po sériové lince RS 232C. Tím bylo provedeno testování rozhraní pro následný styk s přístrojem A2000.

Program „static.c“ slouží jako jednoduchý http server zobrazující statickou HTML stránku. Takto se prověřila správná funkce ethernetového rozhraní.

Další ověření probíhalo již při samotném vývoji a testování aplikačního softwaru.

.

8. NÁVRH SOFTWARU

Ke správné činnosti navrhovaného rozhraní A2000 – Ethernet je potřeba vybavit mikroprocesor Rabbit odpovídajícím softwarem. Ten má plnit několik funkcí:

- Komunikace s přístojem A2000 po asynchronní sériové lince RS 232C
- Provoz HTTP serveru s patřičným uživatelským obsahem
- Komunikace s uživatelským Java Appletem
- Provoz FTP serveru pro možnost záměny obsahu poskytovaného HTTP serverem

8.1 POPIS VYUŽITÝCH PROSTŘEDKŮ DYNAMIC C

8.1.1 Operační systém reálného času

K zajištění všech požadovaných funkcí v reálném čase a možnosti multitaskingu bylo zvoleno použití operačního systému reálného času uC/OSII dodávaného v rámci vývojového prostředí Dynamic C. Tento operační systém umožňuje nadefinování úloh s potřebnými prioritami, jejich spuštění a následné řízení.

Jednotlivé úlohy mají určenu prioritu číslem a dvě úlohy nemohou mít stejnou úroveň priority. Tato priorita se určuje při definování jednotlivých úloh, ale lze ji měnit i za běhu operačního systému. Nižší číslo odpovídá vyšší prioritě. Úlohu vytvoříme následovně:

*INT8U OSTaskCreate (void (*task)(), void *pdata, INT16U stk_size, INT8U prior);*

kde: *task* je ukazatel na startovací adresu úlohy
pdata je ukazatel na inicializační parametry úlohy
stk_size je velikost zásobníku pro úlohu
prior je jedinečné číslo udávající prioritu
Návratovou hodnotou je chybový kód.

Předávání strojového času musíme zajistit tak, že v rámci úlohy s vyšší prioritou definujeme časovou prodlevu (jakési vnitřní přerušení úlohy) během níž jsou dle svých priorit vykonávány úlohy zbývající. Jakmile časová prodleva vyprší jsou procesy s nižšími prioritami přerušeny a řízení je předáno původní úloze dokud opět nebude povolena prodleva. Prodlevu můžeme definovat buď pomocí klasického časového úseku v hodinách, minutách, sekundách a milisekundách nebo pomocí násobku tzv. TICKů. V rámci operačního systému reálného času máme možnost definovat rychlost s jakou bude prováděno přerušení úloh hlavní rutinou systému. To se definuje jako počet TICKů za 1 sekundu a standardní nastavení je 64. Oba způsoby prodlevy vypadají následovně:

void OSTimeDly (INT16U ticks);
INT8U OSTimeDlyHMSM (INT8U hours, INT8U minutes, INT8U seconds, INT16U milli);

kde příslušné významy jsou patrné z názvů. Ve druhém případě funkce vrací chybový kód a skutečné trvání prodlevy je zaokrouhleno na nejbližší hodnotu TICKů za sekundu.

Potřebujeme-li opustit úlohu z nižší prioritou za účelem návratu k úloze s prioritou vyšší, můžeme využít funkci, která předá řízení nazpět:

INT8U OSTimeDlyResume (INT8U prio);

kde *prio* udává prioritu úlohy které chceme předat řízení a návratovou hodnotou je chybový kód.

Samotné spuštění operačního systému je provedeno funkcí:

void OSStart ();

8.1.2 Sériová komunikace

Pro snadnější komunikaci přes sériové porty, obsahuje Dynamic C knihovnu RS232.LIB s potřebnými funkcemi pro přenos jednotlivých znaků či celých bloků dat. U sériových portů v asynchronním režimu lze nastavit použití žádné, sudé či liché parity, sedmi nebo osmibitový formát a jeden nebo dva stop-bity. V případě komunikace s přístrojem A2000 je třeba použít osmibitový formát se sudou paritou a jedním stop-bitem. Potřebné funkce pro nastavení při použití sériového portu D vypadají následovně:

void serDparity(int state);

kde *state* nabývá hodnot *PARAM_OPARITY*
PARAM_EPARITY
PARAM_2STOP
PARAM_NOPARITY

void serDdatabits(int state);

kde *state* nabývá hodnot *PARAM_7BIT*
PARAM_8BIT

int serDopen(long baud);

je funkce pro otevření sériového portu s komunikační rychlostí *baud*

void serDclose();

naopak slouží k uzavření portu.

Definicemi hodnot DINBUFSIZE a DOUTBUFSIZE nastavíme velikost vyrovnávacích pamětí pro příjem či odesílání. Hodnoty musí mít velikost $2^n - 1$.

Pro vlastní blokový přenos dat (vysílání a příjem) slouží následující funkce:

*int serDwrite(void *data, int length);*
*int serDread(void *data, int length, unsigned long tmout);*

kde **data* je ukazatel na blok dat, do kterého se zapisuje při příjmu nebo se z něj vyčítá při odesílání.
length je počet bajtů, které chceme přijmout nebo odeslat.
tmout je doba v milisekundách mezi dvěma příchozími bajty, za kterou se ukončí příjem, i když není načten potřebný počet bajtů.

Návratové hodnoty jsou potom skutečný počet přijatých nebo odeslaných bajtů.

Potřebujeme-li vyprázdnit příchozí nebo odchozí vyrovnávací paměť daného sériového portu máme možnost použití následujících dvou funkcí.

void serDwrFlush();
void serDrdFlush();

První funkce je pro odchozí vyrovnávací paměť, druhá pro příchozí.

8.1.3 HTTP server

Další z knihoven v prostředí Dynamic C slouží k zajištění funkcí HTTP serveru. Existují dvě možnosti, jak provést patřičné asociace souborů uložených v pamětech Rabbita s HTTP serverem. První je použití struktury HttpSpec zaimplementované přímo v knihovně serveru. Druhou možností, v programu využitou, je použití struktury ServerSpec z knihovny ZSERVER.LIB. Tímto umožníme přístup k souborům jak HTTP severu tak FTP serveru. Funkce pro práci s touto strukturou budou popsány dále.

Pro inicializaci paketového ovladače použijeme funkci:

int sock_init();

Inicializaci HTTP serveru pak provedeme následující funkcí:

int http_init(void);

která při úspěšném provedení vrací hodnotu 0.

Pro HTTP server je dále třeba provést rezervaci využívaného portu č. 80 a to následovně:

word tcp_reserveport(word port);

Po těchto krocích již stačí pravidelně volat funkci, která zajišťuje pasivní řízení serveru:

int http_handler(void);

Pokud chceme komunikovat aktivně např. s CGI funkcí pomocí metody POST musíme zajistit patřičnou obsluhu událostí plynoucích ze získaných parametrů, které HTTP server předá funkci. Tento postup bude popsán v kapitole 8.4.2.

8.1.4 FTP server

Vlastní výkonné funkce FTP serveru jsou obsaženy v knihovně FTP_SERVER.LIB. Z této se pak mnohé operace jako čtení či zápis souborů a získání potřebných informací o uložených souborech odkazují na knihovnu ZSERVER.LIB. Soubory používané pro FTP server mají pouze jedinou možnost asociace a to právě přes strukturu ServerSpec. Inicializace FTP serveru je pak obdobná jako u HTTP serveru. Nejprve se musí aktivovat paketový ovladač, potom případně rezervovat port č. 21 a provést vlastní inicializaci FTP serveru:

*void ftp_init(FTPhandlers *handlers);*

kde **handlers* je ukazatel na funkce obsluhující jednotlivé příkazy z terminálu. Pokud je hodnota NULL, použijí se standardně definované funkce v rámci knihovny FTP_SERVER.LIB.

Dále je třeba provádět periodické volání obsluhy funkcí:

void ftp_tick(void);

8.1.5 Knihovna ZSERVER.LIB

Jak již bylo popsáno, slouží tato knihovna k propojení FTP a HTTP serverů tak, že jim umožňuje sdílet soubory definované ve struktuře ServerSpec.

Přidání souboru uloženého v sériové paměti do struktury se provede následovně:

int sspec_addsf1000file(char name, byte filenum, word servermask);*

kde **name* je ukazatel na řetězec znaků se jménem souboru
filenum je označení pozice souboru v sériové paměti Flash
servermask udává servery, se kterými budou soubory asociovány.

Položka *servermask* může nabývat předdefinovaných hodnot SERVER_FTP, SERVER_HTTP a SERVER_USER. Dále byla dodefinována hodnota SERVER_BOTH jako logický součet hodnot serverů HTTP a FTP.

Výše popsaná funkce byla do knihovny doplněna, protože ta dříve neobsahovala potřebné rozhraní pro práci se sériovou pamětí FLASH. Naopak obsahuje obdobné asociační funkce pro soubory uložené v základní (root) či rozšířené (xmem) paměti a nebo v souborovém systému FS (file system). Dále je možné asociovat funkce a proměnné, což ovšem není v této práci využito.

Pro odstranění asociace souboru ze struktury ServerSpec se použije následující funkce:

int sspec_remove(int sspec);

kde *sspec* je číslo asociace ve struktuře. To je třeba nejprve určit další vyhledávací funkcí, proto je v našem případě výhodnější použít funkci, která odstraní všechny záznamy daného typu:

int sspec_removebytype(int type);

kde *type* udává typ souboru např. SF1000FILE.

Dále obsahuje knihovna funkce pro načtení části souboru nebo zjištění délky souboru, atd. Tyto funkce jsou využity například knihovnami HTTP serveru a FTP serveru. Protože však nejsou využity přímo v navrhovaném softwaru, nebudou zde podrobněji popsány.

8.2 KOMUNIKAČNÍ PROTOKOL HTTP SERVERU

Pro komunikační metodu POST, která zajišťuje předávání dat mezi HTTP serverem a klientem je třeba vytvořit protokol, který bude dbát na správný význam jednotlivých přenášených položek.

Data jsou při použití metody post přenášena ve tvaru:

název_parametru1=hodnota1&název_parametru2=hodnota2&...,

kde znak & je oddělovač jednotlivých položek, které představují parametry pro CGI program (skript) vykonávaný na HTTP serveru. Stačí nám tedy definovat posloupnost jednotlivých parametrů a jejich význam. Tento význam musí být shodně interpretován jak na straně HTTP serveru, tak i na straně klienta, proto přesná struktura významů parametrů vznikla společně s navazující prací „Web server pro přístroje A2000“ studenta Jiřího Procházky, v níž je navržen jak HTTP server, tak i Java Applet, který zajišťuje klientskou stranu komunikace.

8.2.1 Řídící parametry

První parametr *query* (dotaz) je tzv. řídící parametr, který rozděluje typ přenášeného dotazu do tří skupin:

- Inicializační (*init*)
- Konfigurační (*get_config, set_config*)
- Měřené veličiny (*measured_values*)

Dle těchto skupin je dále určen význam následujících parametrů. Parametr *query* může nabývat hodnot uvedených v závorce za položkami seznamu. Zápis pro získání konfigurace pak vypadá následovně:

query=get_config&...

8.2.2 Parametry pro inicializaci

Tyto parametry jsou použity k provedení základních inicializačních a testovacích procedur přístroje A2000, popř. k dalším nastavením přístroje A2000 nebo komunikačních parametrů programu. Jsou určeny parametrem *query*, který následuje po parametru *proc* a mohou být následující:

- *Is_OK* - testuje základní připravenost přístroje
- *Reset_Instrument* - provádí základní inicializaci přístroje
- *Reset_All* - provede zákl. inicializaci a smazání trvalých hodnot
- *Reset_Memories* - provede smazání trvalých měř. hodnot přístroje
- *Get_Comparam* - zjistí aktuální parametry komunikace s přístrojem
- *Set_Comparam* - nastaví tyto parametry (rychlost v baudech a adresa)
- *Start_Logger* - spustí záznam dat do paměti přístroje
- *Stop_Logger* - ukončí záznam dat do paměti přístroje

V případě nastavení komunikačních parametrů následují další předávané parametry s hodnotami a to *address* a *baudrate*. Ostatní volby jsou bez dalších parametrů. Například základní inicializace se provede následovně:

query=init&proc=Reset_Instrument

8.2.3 Parametry pro konfiguraci přístroje

Tyto parametry slouží pro získání aktuálního stavu konfigurace nebo pro změnu nastavení. Pro získání konfigurace slouží hodnota *get_config* a pro nastavení nové hodnota *set_config* parametru *query*. Po něm následuje parametr *hardware*, který může nabývat následujících hodnot:

- *relay_1, relay_2* – pro konfiguraci relé
- *input* – pro konfiguraci vstupů
- *pulse_out_1, pulse_out_2* – pro konfiguraci pulsních výstupů

- *display* – pro nastavení displaye
- *analog_out_1, ... , analog_out_4* – pro konfiguraci analogových výstupů
- *logger* – pro konfiguraci zapisovače
- *rtc* – pro nastavení hodin reálného času

Při vyčítání aktuální konfigurace se již neuvádějí žádné další parametry. Při nastavování je třeba uvést ještě parametry s hodnotami jednotlivých konfiguračních údajů:

- pro relé – *source, limit, level, hystereze, store*
- pro vstupy – *line_type, primary_U, secondary_U, primry_I, secondary_I, synch_input*
- pro pulsní výstupy – *source, energy_unit, type, pulses*
- pro display – *brightness*
- pro analogové výstupy – *source, output_type, range_lo, range_hi*
- pro zapisovač – *interval, time, start_mode, ext_run, pre_run, write_mode, channel_1 ... channel_10*
- pro hodiny – *seconds, minutes, hours, day, month, year, millenium*

Parametry je třeba uvádět v naznačeném pořadí a v tomtéž pořadí jsou vráceny zpět při dotazu na aktuální hodnoty konfigurace. Následuje příklad pro zápis konfigurace hodin reálného času.

query=set_config&hardware=rtc&seconds=2&minutes=11&hours=14&day=12&month=5&year=2&millenium=20

8.2.4 Parametry pro měřené veličiny

Pro získání měřených veličin je třeba k parametry *query* uvést hodnotu *measured_values*. Poté následuje seznam veličin, které požadujeme. Příslušná hodnota odpovídající měřené veličině je vždy připojena k parametru *magnitude*.

Pokud chceme získat všechny měřené veličiny, bude parametr *magnitude* obsahovat hodnotu *all*.

Následuje výčet hodnot odpovídající jednotlivým měřeným veličinám:

U1, U2, U3, U1_max, U2_max, U3_max, U12, U23, U31, U12_max, U23_max, U31max, I1, I2, I3, I1_max, I2_max, I3_max, I1_avg, I2_avg, I3_avg, I1_avg_max, I2_avg_max, I3_avg_max, P1, P2, P3, P_sum, P1_max, P2_max, P3_max, P_sum_max, Q1, Q2, Q3, Q_sum, Q1_max, Q2_max, Q3_max, Q_sum_max, S1, S2, S3, S_sum, S1_max, S2_max, S3_max, S_sum_max, PF1, PF2, PF3, PF_sum, PF1_max, PF2_max, PF3_max, PF_sum_max, P_int_sum, Q_int_sum, S_int_sum, Freq.

Příklad zápisu pro získání všech měřených veličin:

query=measured_values&magnitude=all

8.3 POUŽITÍ PAMĚTI DATAFLASH

Pro práci se sériovou pamětí FLASH, označovanou jako DataFLASH, slouží knihovna SF1000.LIB. Tato dále spolupracuje s knihovou SPI.LIB pro použití standardního sériového komunikačního přístupu definovaného jako SPI. Knihovna SF1000 obsahuje pro uživatele jen funkce pro inicializaci paměti, čtení a zápis dat z příslušného místa nebo mazání bloků dat. Proto musela být vytvořena koncepce ukládání souborů a jejich odkazů v návaznosti na další funkce a knihovny, které budou s těmito soubory dále pracovat.

V hlavním programu je možné definovat velikost tabulky souborů. Tato se určuje v násobcích paměťových stránek o velikosti 256 bajtů. Každá stránka dokáže uchovat záznam až o osmi souborech, přičemž jejich jména můžou být dlouhá 20 bajtů.

Po spuštění programu je provedena inicializace paměti DataFLASH, která kontroluje tabulku záznamů a najde-li uložené soubory, asociuje je se strukturou ServerSpec pro servery HTTP a FTP. Soubory jsou ukládány kontinuálně za sebe. Jednotlivý soubor nelze přepsat. Vymazat lze jen všechny soubory. Potom je možné libovolně nahrávat soubory nové, dokud se nevyčerpá kapacita paměti nebo tabulky souborů. Tato omezení byla zavedena vzhledem k předpokládanému jednoduchému a jednoúčelovému využití paměti DataFLASH.

Pro možnost použití HTTP serveru s adresou bez přesné specifikace stránky je provedena asociace jména „\“ k prvnímu soubory uloženému v paměti DataFLASH. Proto prvním souborem by měla být HTML stránka (např. index.html, ale na jménu nezáleží) s dalšími potřebnými odkazy, resp. zajišťující spouštění Java Appletu.

8.4 VLASTNÍ PROGRAM

Program je realizován v prostředí Dynamic C, za pomoci výše popsaných prostředků zahrnujících funkce standardně obsažené v dodávaných knihovnách a nově vytvořených komponent. Tyto jazykové komponenty (funkce, proměnné, konstanty, struktury a definice) jsou obsaženy zejména v hlavním programovém modulu, ale také byly doplněny do původních knihoven. Veškeré změny provedené v původních knihovnách jsou popsány v kapitole 8.5.

8.4.1 KONCEPCE PROGRAMU

Jádrem programu je operační systém reálného času se dvěma úlohami, čímž je zajištěn paralelní, resp. pseudo-paralelní běh jednotlivých částí. Na začátku každé úlohy jsou provedeny potřebné inicializační úkony a poté je zbytek kódu uzavřen do nekonečné smyčky:

```
while(1) {
... kód úlohy...
}
```

V první úloze (s vyšší prioritou) je realizována část komunikace s přístrojem A2000. Z důvodu nutného předávání řízení úloze s nižší prioritou je provádění rozděleno do malých úseků zajišťujících zpracování vždy jen jedné či několika málo komunikačních zpráv. Tento časový multiplex je zajištěn pomocí následující konstrukce:

```
switch(sercomstate) {
 case 0:
 ...
 case n:
 default:
}
```

kde globálně definovaná proměnná *sercomstate* určuje, která část se bude vykonávat. Tato proměnná je nastavována buď v jednotlivých výkonných úsecích nebo z vnějšku této úlohy např. dle požadavků Java Appletu.

Před nekonečnou smyčkou je inicializována sériová komunikace otevřením portu na počáteční komunikační rychlosti 19200 baudů a nastavením sudé parity. Poté jednotlivé části komunikace využívají volání funkcí, které jsou umístěny mimo její vlastní kód. Tyto slouží např. pro odeslání nebo příjem zprávy, popř. zpracování přijatých dat. Většina úprav s přijatými daty, jako je převod na znaménková čísla nebo násobení konstantami, se však provádí přímo v dané části komunikační úlohy. Ukázka typické části úlohy:

```
case 1: {
    if (n=ReceiveAnswer(FullRec, Comparam.address, 0x32, &DataSerCom))
     {
       KU = DataSerCom[1];
       if (KU > 127) KU = (256 - KU)*-1;
       KI = DataSerCom[2];
```

```
        if (KI > 127) KI = (256 - KI)*-1;
        KP = DataSerCom[3];
        if (KP > 127) KP = (256 - KP)*-1;
        KE = DataSerCom[4];
        if (KE > 127) KE = (256 - KE)*-1;
        com_error = 0;
        SendContRec(Comparam.address, 0x89, 0x00);
        sercomstate++;
        break;
      }
    else
      {
        com_error = 1;
        sercomstate=0;
        break;
      }
}
```

Tato část se bude provádět v případě, že proměnná *sercomstate* zajišťující časový multiplex nabývá hodnoty 1. Potom je přijata zpráva z přístroje A2000 (žádost o vyslání zprávy s příslušným obsahem byla vystavena na konci předcházející části, tedy pro *sercomstate* = 0) využitím funkce *ReceiveAnswer*. Pokud byla přijata odpovídající data, je provedeno jejich zpracování, nastavení příznaku chyby *com_error* na hodnotu nula, odeslání žádosti o další data funkcí *SendContRec*, zvýšení proměnné *sercomstate* na další komunikační část a opuštění multiplexu příkazem *break*. Pokud byla obdržena špatná data, je příznak chyby nastaven na 1, provádění komunikačních částí na začátek a opět odskok.

Jakmile opustí tok programu části komunikačního multiplexu, dostává se na konec smyčky *while(1) {...}*, kde je provedeno předání řízení úloze s nižší prioritou na dobu 40 ms.

Skladba komunikačních částí je následující (hodnoty proměnné *sercomstate*):

- 0 – 15: Vyčítání měřených dat. Prováděno neustále cyklicky, aby v proměnných uchovávajících tato data byly vždy nejnovější údaje.
- 20 – 46: Vyčítání nebo nastavování konfiguračních hodnot přístroje A2000. Vždy je provedena jen daná část nastavující konkrétní parametr a nastaven návrat na začátek komunikačního multiplexu.
- 50: Tato část je výjimečná tím, že provádí změnu komunikační rychlosti a adresy přístroje A2000. Proto je v ní uzavřen sériový port a následně otevřen s novým parametrem.
- *default*: Pouze nastaví provádění na počátek.

Druhá úloha (s prioritou 2) obsluhuje HTTP server a FTP server. Je velice krátká, jelikož veškeré úkony zajišťují buď původní knihovní funkce nebo nově vytvořené funkce nacházející se mimo tělo této úlohy. Vypadá následovně:

```
nodebug void Servers(void* pdata){
 int i;

  sock_init();
  http_init();
  ftp_init(NULL);
  tcp_reserveport(80);

 for (i=0; i<74; i++)
 measured_values[i] = 0;

  while (1) {
    http_handler();
    ftp_tick();
  }
}
```

Na začátku je provedena inicializace síťových funkcí protokolu TCP/IP. Následně jsou inicializovány proměnné uchovávající měřené veličiny a pak již probíhá jen pravidelné volání funkcí pro obsluhu HTTP serveru a FTP serveru.

8.4.2 Obsluha požadavků komunikace s HTTP serverem

Pro obsluhu požadavků od klienta, které mají účel změny nastavení měřícího přístroje, získání měřených hodnot atd. slouží komunikační metoda POST. K její interpretaci je nutné doplnit stávající funkce HTTP serveru o vlastní funkci, resp. CGI skript, na nějž jsou požadavky odkazovány. Tento musí zajistit jejich správnou interpretaci, v případě chybné komunikační metody (např. GET místo POST) nebo jiné chyby při síťové komunikaci vystavit odpovídající zprávu klientovi a konečně správně obsloužit klientské požadavky. V našem případě se správným obsloužením rozumí např. zaslání hodnot měřených veličin, stavu konfigurace nebo provedení změn ve stávající konfiguraci měřícího přístroje A2000.

Samotné vykonání CGI funkce, konkrétně funkce *a2000.cgi* je z důvodů rovnoměrného rozložení času a včasného předávání řízení zpět obsluze HTTP serveru rozděleno na více částí konstrukcí switch – case. Zde je ovšem jako indikace stavu využita struktura *http_state*, resp. její podúrovně *state* a *substate*. Nejprve je provedena kontrola správnosti komunikační metody a při špatné metodě je odeslán chybový kód 501. Po ní následuje čtení dat a pokud proběhlo správně, je zjištěn počet parametrů získaných od klienta a tyto jsou převedeny do tabulky. Při chybném četní socketu je odeslán návratový kód 500. V tabulce je vždy na daném řádku název parametru a hodnota parametru, počet řádků odpovídá počtu přijatých parametrů.

Následně je volána funkce *proces_table*, která zpracuje požadavek a vytvoří odpověď pro klienta. Pokud proběhla v pořádku, je odeslán kód 200, jinak je odeslán kód 400. Tím je zpracování požadavku ukončeno.

8.4.3 Ostatní funkce v programu

Program obsahuje mnoho funkcí, které jsou využívány buď některou úlohou operačního systému reálného času nebo i vnější knihovnou. Následuje jejich stručný popis:

void send_message(HttpState state, int state_code, char* reply);*

Posílá zprávu klientovi. Parametr *state_code* udává typ odpovědi serveru, **reply* je ukazatel na řetězec znaků, které mají být odeslány a **state* je ukazatel na strukturu *HttpState*, do jejíhož bufferu budou nakopírována data k odeslání.

char format(char* buffer, int index);*

Formátuje měřené veličiny na správný tvar k odeslání. Parametry jsou ukazatel na buffer k uchování formátovaných veličin a index požadované veličiny v tabulce měřených veličin. Návratovou hodnotou je ukazatel na upravenou veličinu v bufferu.

int process_table(int tokens, char reply, HttpState* state);*

Funkce zpracovává požadavek klienta a připraví odpověď k odeslání. Parametr tokens udává počet parametrů v tabulce, *reply je ukazatel na buffer pro odpověď, *state je ukazatel na strukturu HttpState. Pokud vše proběhlo v pořádku vrací hodnotu 1, jinak 0.

int get_tokens(char buffer);*

Naformátuje požadavek klienta uložený v bufferu do tabulky dle jednotlivých parametrů. Vrací počet parametrů v tabulce.

int read_socket(HttpState state, char* buffer);*

Načte socket s požadavkem klienta do bufferu. Vrací počet načtených znaků nebo 0 pokud je vstup příliš dlouhý.

int InitDataFS(void);

Provede kontrolu sériové paměti FLASH na přítomnost souborů. Nalezené soubory asociuje se strukturou ServerSpec pro HTTP a FTP server. Vrací počet nalezených souborů.

*int NewFileInTable(char *f_name);*

Vytvoří v tabulce souborů záznam pro nový soubor a otevře jej k zápisu. Parametrem je ukazatel na jméno. Pokud je místo na záznam souboru v tabulce, je vrácena 0, jinak –1.

*int WriteFileData(long fbegin, char*fdata, int flength);*

Zapisuje data to aktuálně otevřeného souboru. Parametr *fbegin* udává offset od počátku souboru, **fdata* udává umístění zapisovaných dat a *flength* počet zapisovaných bajtů. Vrací hodnotu 0 (zápis v pořádku) nebo –1.

*int ReadFileData(int file_number, long fbegin, char*fdata, int flength);*

Načte data ze souboru definovaného *file_number*. Ostatní parametry jsou stejné jako u předchozí funkce. Vrací počet načtených bajtů.

int EndOfWritingFile(long fsize);

Uzavře soubor, do kterého byl prováděn zápis a do tabulky doplní konečnou délku souboru. Parametrem je délka zapsaného souboru.

*int DeleteFiles(char*f_name);*

Smaže soubory uložené v sériové paměti FLASH, resp. záznamy v tabulce souborů a zruší asociace se strukturou ServerSpec. Pokud je jako parametr předáno slovo *all*, provede se smazání. Jinak se vrací funkce s hodnotou –1. Pokud již není co mazat také funkce vrací hodnotu –1. Při správném provedení vrací 0.

long GetLength(int file_number);

Vrací délku souboru udávaného parametrem *file_number*.

int ReceiveAnswer (int MessType, int InstrumentAddress, int ParameterIndex, void DataBlock);*

Přijme zprávu z přístroje A2000. *MessType* udává o jaký typ zprávy se jedná, *InstrumentAddress* nastavenou adresu měřícího přístroje, *ParameterIndex* specifikaci zprávy a **DataBlock* místo kam se budou ukládat načtená data. Funkce vrací počet načtených bajtů nebo záporné hodnoty jako chybové kódy.

int SendAbbRec (int InstrumentAddress, int FunctionField);

Odešle zprávu Abbreviated Record do přístroje A2000. Vrací 0 při správném odeslání nebo 1 při chybě.

SendContRec (int InstrumentAddress, int FunctionField, int ParameterIndex);

Odešle zprávu Control Record do přístroje A2000. Vrací 0 při správném odeslání nebo 1 při chybě.

int SendFullRec (int InstrumentAddress, int FunctionField, int ParameterIndex, void DataBlock, int DBLength);*

Odešle zprávu Full Record do přístroje A2000. Vrací 0 při správném odeslání nebo 1 při chybě. **DataBlock* ukazuje na data k odeslání a *DBLength* udává počet bajtů, které se mají odeslat.

float tsqr(int pow);

Funkce vrací mocninu deseti. Mocnitel je parametr *pow*. Používá se při posunu řádových čárek.

unsigned char CodeSource(unsigned char Sour);

Zakóduje označení měřené veličiny udávané parametrem *Sour* do tvaru používaného u některých konfiguračních nastavení.

unsigned char DetectSource(unsigned int Sour);

Dekóduje označení měřené veličiny *Sour* na formát používaný web serverem.

8.5 ÚPRAVY V PŮVODNÍCH KNIHOVNÁCH DYNAMIC C

V původních knihovnách prostředí Dynamic C byly provedeny některé změny, potřebné pro funkčnost programu. Část z nich by se dala přenést do hlavního programového kódu nebo do samostatné nově vytvořené knihovny, ale takto je dosaženo větší návaznosti, neboť provedené úpravy jsou jen rozšířením, které nijak neomezuje původní funkčnost knihoven. V některých případech tyto úpravy ani nelze umístit mimo původní knihovny. Zde jsou popsány změny, které byly v původním kódu provedeny a taktéž veškeré upravené knihovny jsou součástí příloh této práce.

8.5.1 Úpravy v FTP_SERVER.LIB

Do knihovny byla doplněna funkce na mazání souborů. Tato ovšem spolupracuje jen se soubory uloženými v paměti DataFLASH. Dále bylo nutné doplnit z příslušných míst kódu odkazy na funkce, které vytváří nový soubor, zapisují jej a uzavírají. Tyto funkce se nacházejí v hlavním programovém kódu. Také muselo být odstraněno blokování ukládání nových souborů.

8.5.2 Úpravy v ZSERVER.LIB

Do této knihovny musela být doplněna definice souborového typu SF1000FILE a definice použití systému souborů v paměti DataFLASH. Byla vytvořena nová funkce *sspec_addsf1000file*, která provádí asociaci souborů se strukturou ServerSpec pro použití se servery HTTP a FTP. Dále do funkcí pro práci se soubory (čtení, zjišťování velikosti, ...) musela být doplněna část, která se stará o návaznost na odpovídající funkce souborů v paměti DataFLASH.

8.5.3 Úpravy v SF1000.LIB

Zde musela být doplněna definice pamětí s potřebnými velikostmi. Původně knihovna podporovala jen paměti o velikosti 32 a 64 Mb. Nyní podporuje i 1 Mb a 4 Mb paměti. Další zásahy nejsou nutné, jelikož veškeré funkce jsou psané univerzálně a paměti jsou navzájem kompatibilní.

8.5.4 Úpravy v HTTP.LIB

Do této knihovny musel být doplněn jediný řádek, a to v části čtení souborů pro následné zpracování a odeslání klientovi. Zde nedokázal server najít soubor označený typem SF1000FILE.

8.6 LADĚNÍ PROGRAMU

Vzhledem k tomu, že navržený modul se nepodařilo včas osadit pamětí FLASH o potřebné velikosti, byl na něm odlaďován program jen do okamžiku, kdy stačila kapacita stávající paměti (128 KB). Poté byla další část ladění přenesena na modul TCP/IP Development Kit (paměť FLASH 256 KB), kde byl návrh doveden do konečné podoby. Na obou modulech však byla odzkoušena komunikace s přístrojem A2000 a funkce rozhraní Ethernet. Spolupráce s pamětí DataFLASH byla odladěna jen při její velikosti 128 KB, opět z důvodu nedostupnosti navrženého obvodu. Vzhledem ke špatné dostupnosti volných a přístupných vývodů prostředí TCP/IP Development Kit byla paměť DataFLASH připojena přes alternativní vývody paralelních portů.

9. ZÁVĚR

Při využití výchozích materiálů popisujících činnost jednotlivých komponentů bylo navrženo a v rámci dostupnosti realizováno zapojení s mikroprocesorem Rabbit 2000 pro komunikaci s přístrojem A2000 a sítí Ethernet. Zapojení aplikačního rozhraní mezi mikroprocesorem Rabbit 2000 a přístrojem A2000 využívá již implementovaného sériového rozhraní RS232. Tímto způsobem je umožněno obejít se bez zásahů do hardwaru i softwaru měřícího přístroje a je tím také zaručena přenositelnost na další měřící přístroje při změně komunikačního softwaru.

Během realizace se bylo nutné potýkat s některými problémy, jako nedostupnost navrhovaných součástek, čímž bylo nutné částečně pozměnit původní koncepci. K pamětem mikroprocesoru Rabbit je tak přidána i sériová datová FLASH sloužící pro systém souborů používaný HTTP serverem a FTP serverem. Tento systém i přes svou relativní jednoduchost podstatně zpříjemňuje aktualizaci dat poskytovaných HTTP serverem.

Návrh aplikačního softwaru byl proveden a odladěn v prostředí Dynamic C. Pro zajištění potřebné funkčnosti byly provedeny malé úpravy i v původních knihovnách. Hlavní program využívá k činnosti operační systém reálného času uC/OSII navržený pro mikroprocesor Rabbit. V jedné úloze je realizována veškerá komunikace s přístrojem A2000, ve druhé úloze pak běží servery HTTP a FTP.

Samotné kompletní odlaďování a testování pak z důvodů problematické plnohodnotné realizace navrženého hardwaru bylo částečně provedeno i na původním modulu TCP/IP Development Kit, který disponoval větší pamětí FLASH. I tak se však dala ověřit dostatečná funkčnost hardwaru i softwaru. Pro praktické použití se pak předpokládá, že firma GMC z navrženého schématu vytvoří čtyřvrstvou profesionální desku plošných spojů osazenou SMD součástkami.

Vzhledem k předpokladu nedostatečného využití byla po konzultaci z realizace vypuštěna služba e-mail klienta.

10. LITERATURA, ODKAZY

[1] Rabbit 2000 Microprocessor User's Manual, Rabbit Semiconductor, 1999

[2] Rabbit TCP/IP Development Kit, Rabbit Semiconductor, 1999

[3] Rabbit 2000 Microprocessor Designer's Handbook, Rabbit Semiconductor, 1999

[4] Dynamic C User's Manual, Z-World, 1999

[5] A2000 Multifunctional Power Meter – Communications Protocol per DIN Draft 19244, GMC, 2001

[6] Kozák, T.: Impl. progr. pro elektroměr s rozhr. Lon Talk - diplomová práce, FEI VUT v Brně, 1999

[7] Pužmanová, R.: Moderní komunikační sítě, Computer Press, Praha 1998

[8] Procházka, J.: Software pro ovládání přístroje A2000 přes Internet - semestrální práce II, FEKT VUT v Brně, Brno 2002

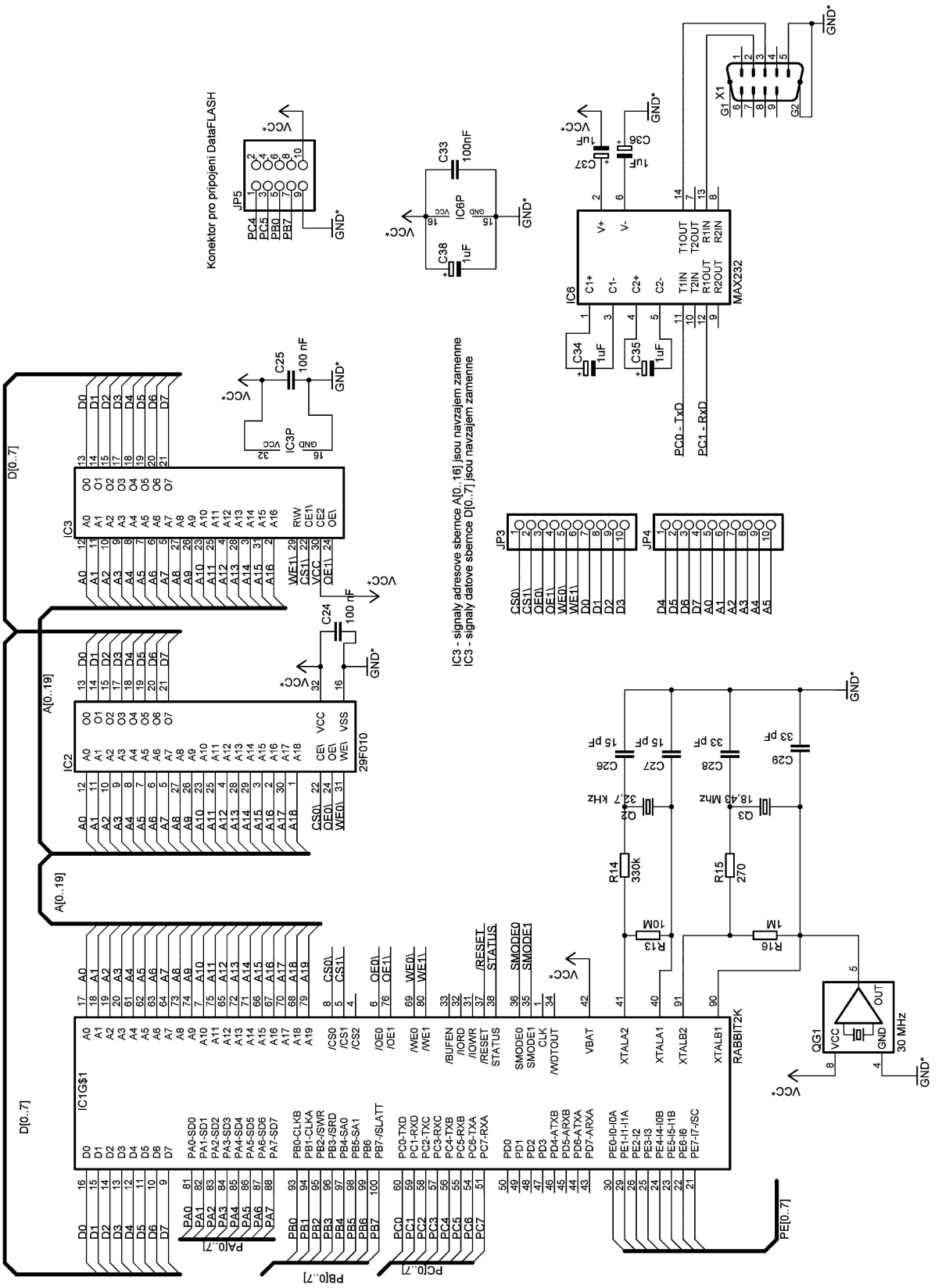
[9] Procházka, J.: Web server pro přístroje typu A2000 - diplomová práce, FEKT VUT v Brně, Brno 2002

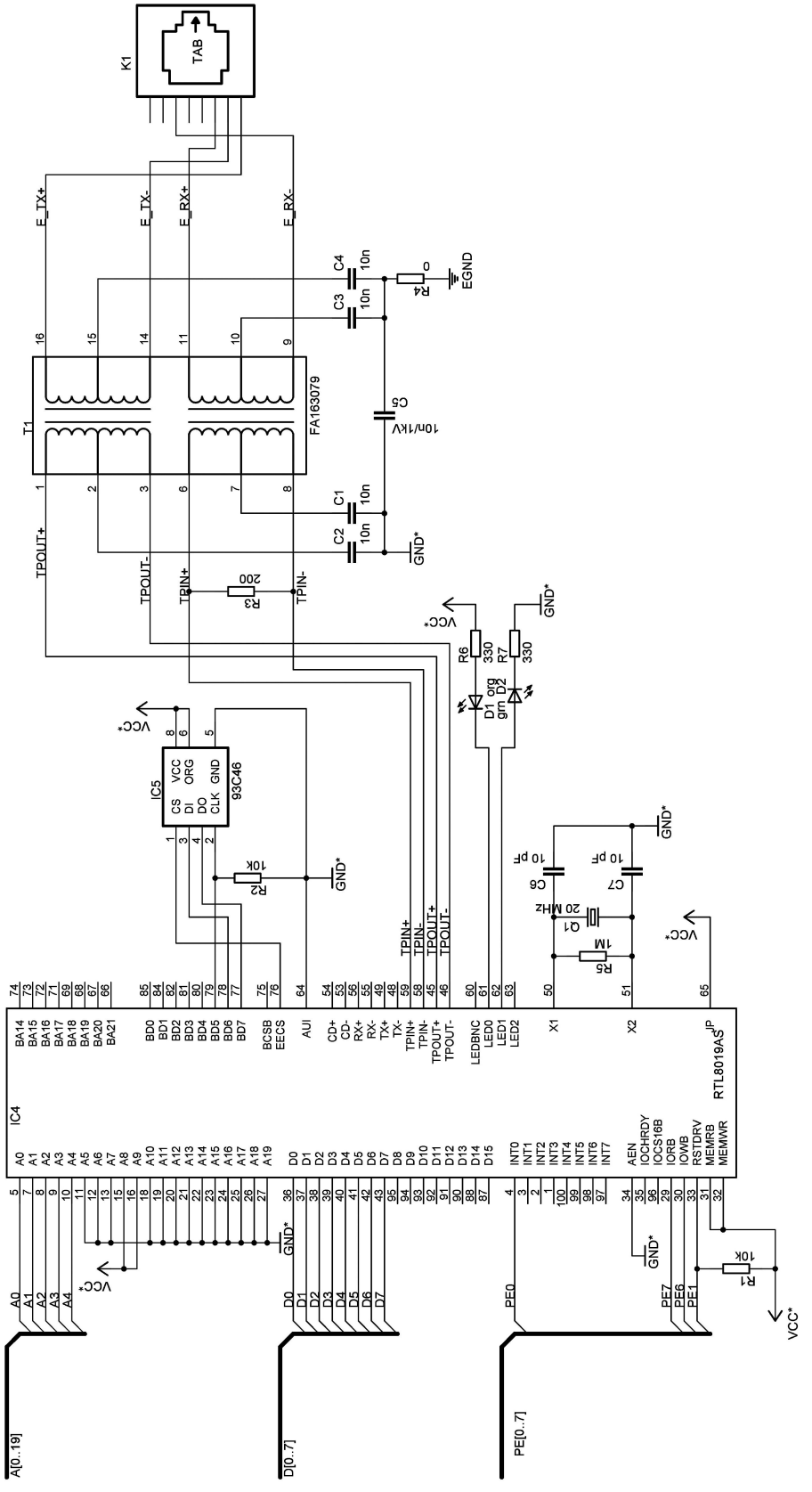
[10] Mikl, M.: Rozhraní přístroje A2000 ke sběrnici Ethernet – semestrální práce II, FEKT VUT v Brně, Brno 2002

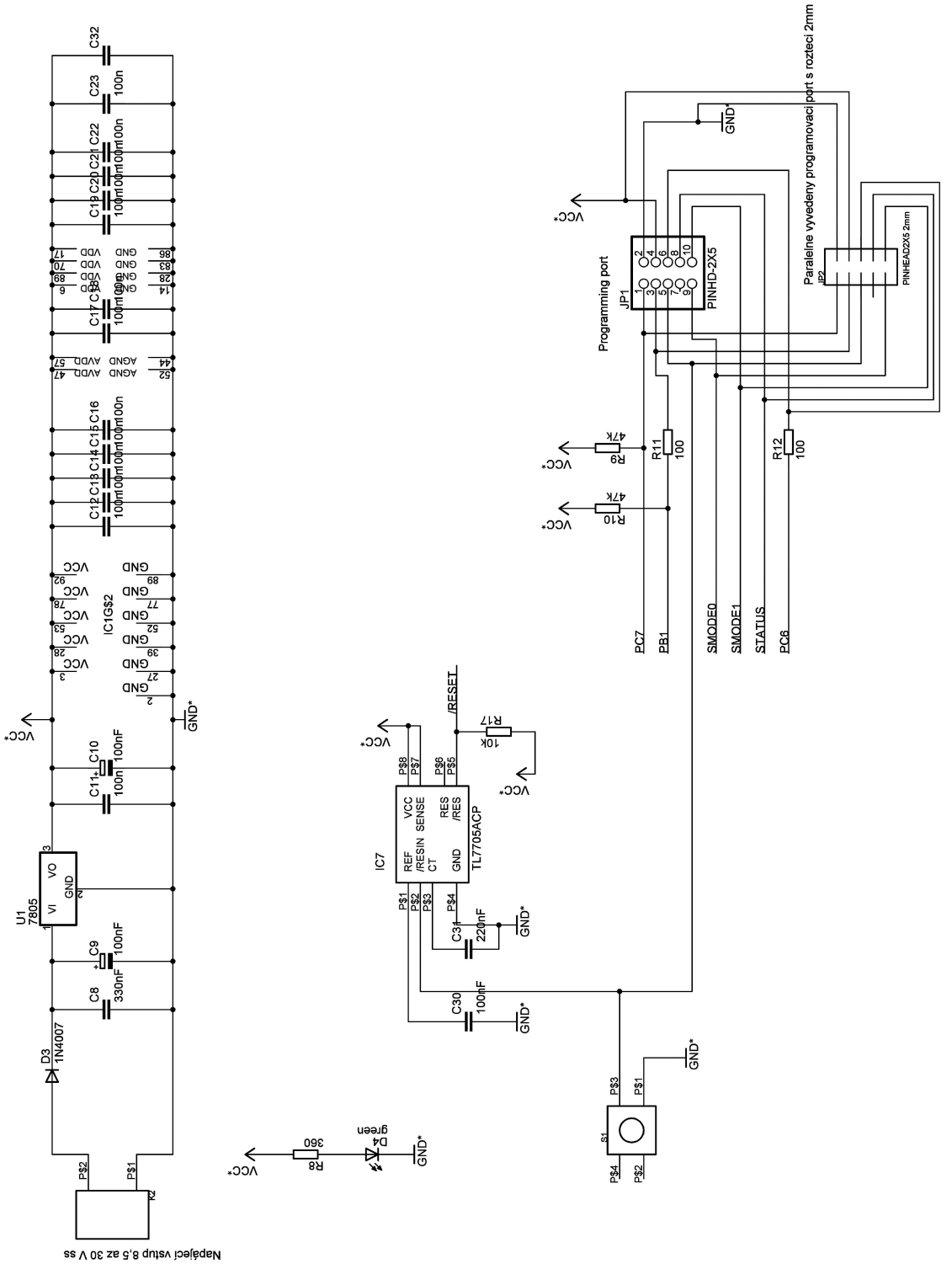
[11] http://www.gmc.cz

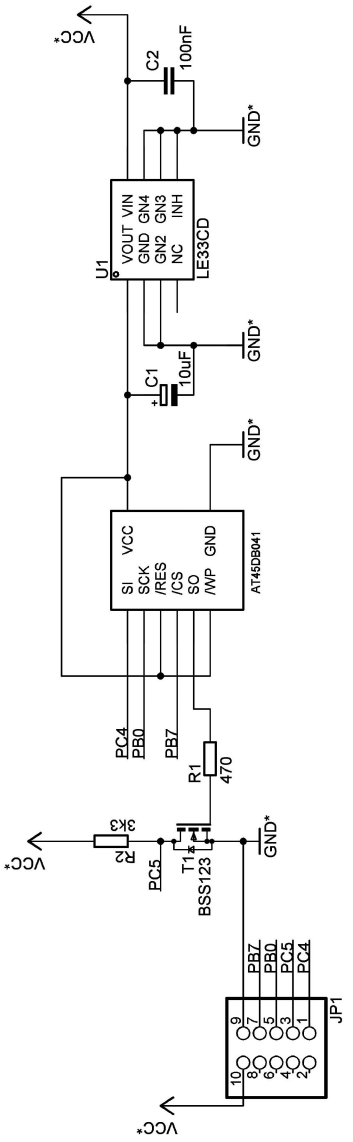
[12] http://www.rabbitsemiconductors.com

[13] http://www.zworld.com

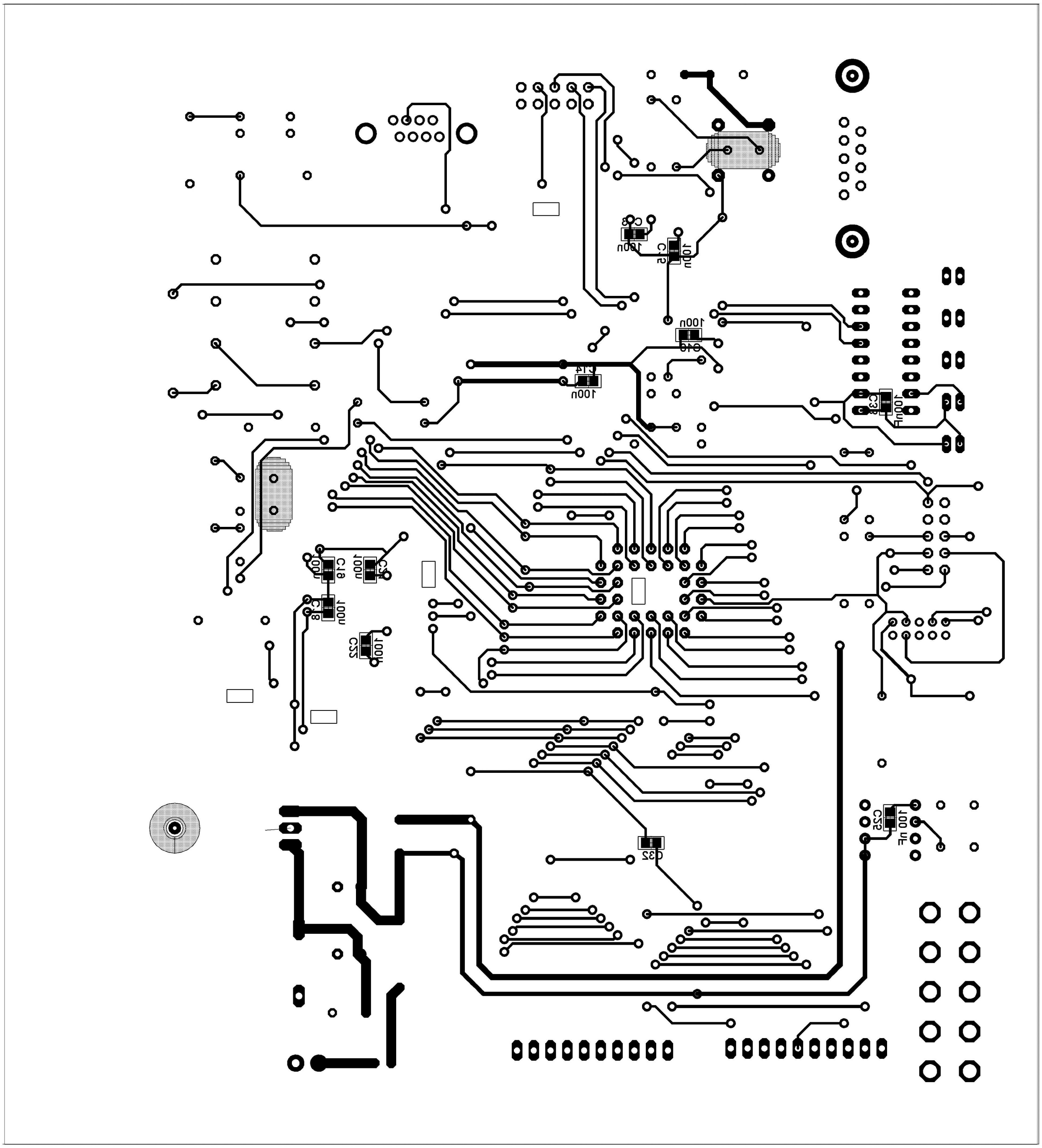

Konektor pro pripojeni DataFLASH
JP5
PC4
PC5
PB0
PB7
VCC*
GND*
C33
100nF
C38
1uF
IC6P
C37
1uF
C36
1uF
X1
G1
G2
IC8
MAX232
C1+
C1-
C2+
C2-
V+
V-
T1IN
T2IN
R1OUT
R2OUT
T1OUT
T2OUT
R1IN
R2IN
C34
1uF
C35
1uF
PC0 - TxD
PC1 - RxD
C25
100 nF
IC3P
IC3
C24
100 nF
IC2
29F010
A[0..19]
D[0..7]
IC3 - signaly adresove sbernice A[0..16] jsou navzajem zamenne
IC3 - signaly datove sbernice D[0..7] jsou navzajem zamenne
JP3
JP4
CS0\
CS1\
OE0\
OE1\
WE0\
WE1\
D0
D1
D2
D3
D4
D5
D6
D7
A0
A1
A2
A3
A4
A5
C26
15 pF
C27
15 pF
C28
33 pF
C29
33 pF
Q2
32,7 kHz
Q3
18,43 Mhz
R14
330k
R15
270
R13
10M
R16
1M
IC1G$1
RABBIT2K
QG1
30 MHz
VCC
GND
OUT
PA[0..7]
PB[0..7]
PC[0..7]
PE[0..7]
PA0-SD0
PA1-SD1
PA2-SD2
PA3-SD3
PA4-SD4
PA5-SD5
PA6-SD6
PA7-SD7
PB0-CLKB
PB1-CLKA
PB2-/SWR
PB3-/SRD
PB4-SA0
PB5-SA1
PB6
PB7-/SLATT
PC0-TXD
PC1-RXD
PC2-TXC
PC3-RXC
PC4-TXB
PC5-RXB
PC6-TXA
PC7-RXA
/CS0
/CS1
/CS2
/OE0
/OE1
/WE0
/WE1
/BUFEN
/IORD
/IOWR
/RESET
STATUS
SMODE0
SMODE1
CLK
/WDTOUT
VBAT
XTALA2
XTALA1
XTALB2
XTALB1

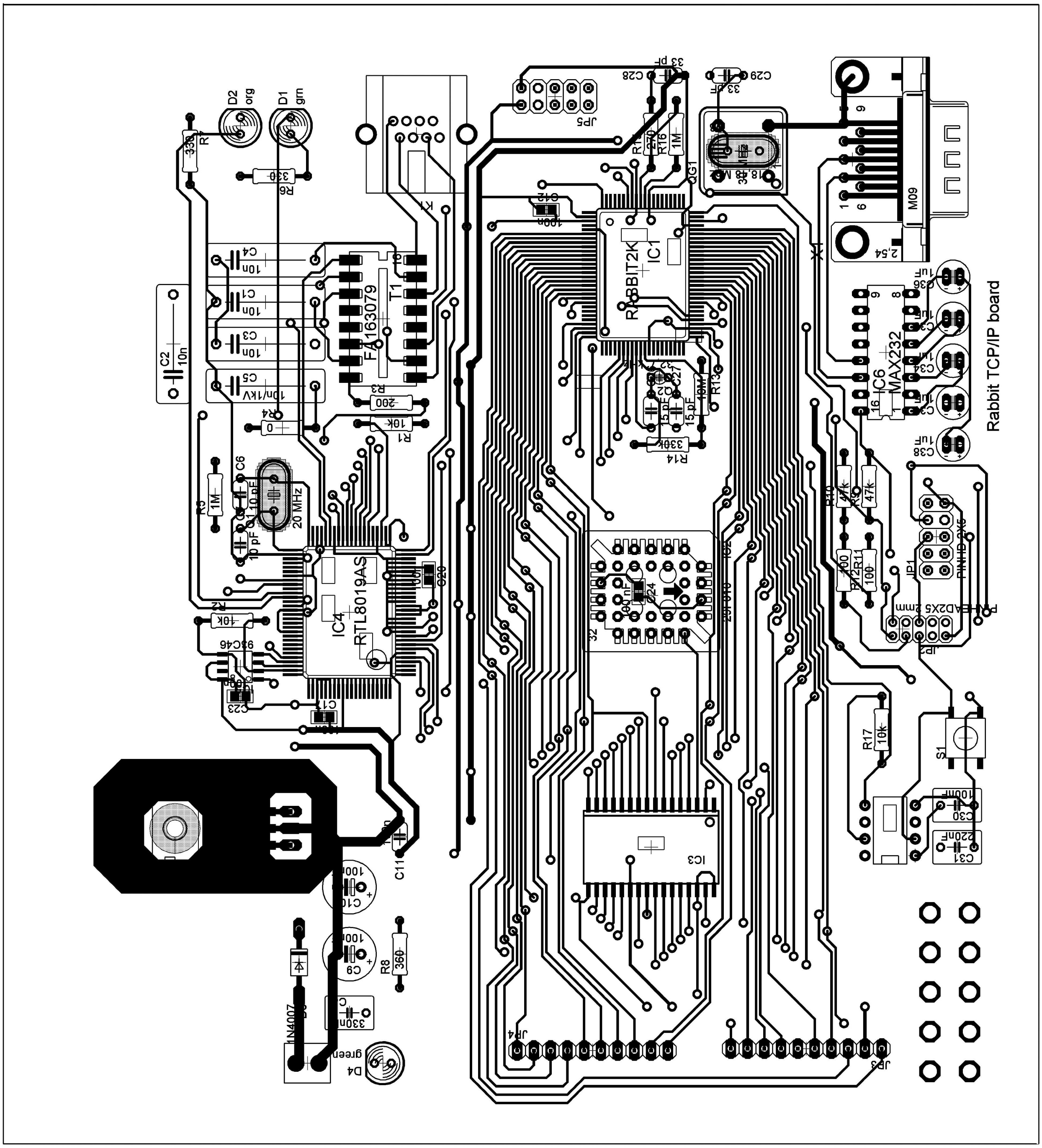

K1
TAB
E_TX+
E_TX-
E_RX+
E_RX-
C4
10n
C3
10n
R4
0
EGND
T1
FA163079
C5
10n/1kV
C1
10n
C2
10n
GND*
TPOUT+
TPOUT-
TPIN+
TPIN-
R3
200
VCC*
R6
330
R7
330
D1 org
grn D2
IC5
93C46
CS
DI
DO
CLK
VCC
ORG
GND
R2
10k
C6
10 pF
C7
10 pF
Q1
20 MHz
R5
1M
IC4
RTL8019AS
BA14
BA15
BA16
BA17
BA18
BA19
BA20
BA21
BD0
BD1
BD2
BD3
BD4
BD5
BD6
BD7
BCSB
EECS
AUI
CD+
CD-
RX+
RX-
TX+
TX-
LEDBNC
LED0
LED1
LED2
X1
X2
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
A16
A17
A18
A19
D0
D1
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
D12
D13
D14
D15
INT0
INT1
INT2
INT3
INT4
INT5
INT6
INT7
AEN
IOCHRDY
IOCS16B
IORB
IOWB
RSTDRV
MEMRB
MEMWR
JP
R1
10k
A[0..19]
D[0..7]
PE[0..7]
PE0
PE7
PE6
PE1

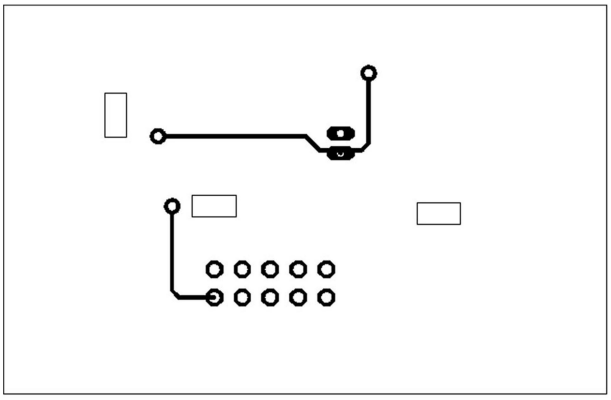

Napájecí vstup 8,5 až 30 V ss
K2
P$2
P$1
D3
1N4007
C8
33OnF
C9
100nF
U1
7805
VI
VO
GND
C11
100n
C10
100nF
VCC*
GND*
IC1G$2
C12 C13 C14 C15 C16
C17 C18 C19 C20 C21 C22 C23 C32
100n
R8
360
D4
green
IC7
TL7705ACP
REF
/RESIN
CT
GND
VCC
SENSE
RES
/RES
C31
220nF
C30
100nF
R17
10K
/RESET
S1
Programming port
JP1
PINHD-2X5
R9
47K
R10
47K
R11
100
R12
100
PC7
PB1
SMODE0
SMODE1
STATUS
PC6
Paralelne vyvedeny programovaci port s rozteci 2mm
JP2
PINHEAD2X5 2mm

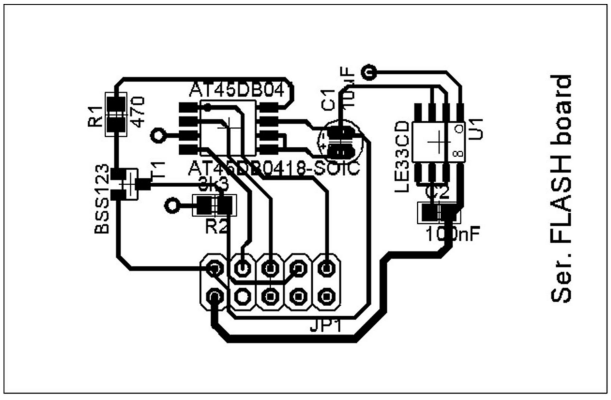

VCC*
C2
100nF
GND*
U1
VOUT VIN
GND GN4
GN2 GN3
NC INH
LE33CD
GND*
C1
10uF
GND*
SI VCC
SCK GND
/RES
/CS
SO
/WP
AT45DB041
PC4
PB0
PB7
R1
470
VCC*
3k3
R2
PC5
T1
BSS123
GND*
PB7
PB0
PC5
PC4
JP1
9
7
5
3
1
10
8
6
4
2
VCC*

Příloha B: Deska plošných spojů **strana 1/3**




Rabbit TCP/IP board
RABBIT2K
IC1
RTL8019AS
IC4
FA163079
T1
MAX232
IC6
IC3
20 MHz
1N4007




AT45DB04
C1
1uF
R1
470
T1
BSS123
3k3
R2
LE33CD
U1
C2
100nF
JP1
Ser. FLASH board

Příloha C: Seznam součástek

List sché-matu	Obvod	Označení součástky	Hodnota	Popis	Pouzdro / rozměry (Eagle 4.0)
1	Rabbit	R13	10M	rezistor	0207/10
		R14	330k	rezistor	0207/10
		R15	270R	rezistor	0207/10
		R16	1M	rezistor	0207/10
		Q2	32,768kHz	krystal	TC26V
		Q3	18,432MHz	krystal	HC49U-V
		C26	15pF	kondenzátor	C050-024x044
		C27	15pF	kondenzátor	C050-024x044
		C28	33pF	kondenzátor	C050-024x044
		C29	33pF	kondenzátor	C050-024x044
		QG1	30MHz	krystalový oscilátor	DIL08S
		IC1	Rabbit 2000	mikroprocesor	PQFP100
		JP5	PINHEAD 2x5	dvouřadý konektor	PINHD-2*5
1	FLASH	C24	100nF	kondenzátor	C0805
		IC2	29F010-90JC	paměť FLASH	PLCC32
1	SRAM	C25	100nF	kondenzátor	C0805
		IC3P	HY628100	napájecí vývody	
		IC3	HY628100	paměť SRAM	SOP 32
1	Test. Vývody	JP3	PINHEAD 1x10	jednořadý konektor	PINHD-1*10
		JP4	PINHEAD 1x10	jednořadý konektor	PINHD-1*10
1	Rozhraní k A2000	C33	100nF	kondenzátor SMD	C0805
		C34	1uF	el. kondenzátor	E2-5
		C35	1uF	el. kondenzátor	E2-5
		C36	1uF	el. kondenzátor	E2-5
		C37	1uF	el. kondenzátor	E2-5
		C38	1uF	el. kondenzátor	E2-5
		X1	CAN9V	konektor CANON 9 V	M09HP
		IC6P	MAX232	napájecí vývody	
		IC6	MAX232	interface RS232/TTL	DIL16
2	Ethernet	IC4	RTL8019AS	ethernetový řadič	QFP100
		IC5	93C446	paměť EEPROM	SO-8
		R1	10k	rezistor	0207/10
		R2	10k	rezistor	0207/10
		R3	200R	rezistor	0207/10
		R4	0R	rezistor	0207/10
		R5	1M	rezistor	0207/10
		R6	330R	rezistor	0207/10
		R7	330R	rezistor	0207/10
		C1	10nF/1kV	kondenzátor	C150-054x183
		C2	10nF/1kV	kondenzátor	C150-054x183
		C3	10nF/1kV	kondenzátor	C150-054x183
		C4	10nF/1kV	kondenzátor	C150-054x183
		C5	10nF/1kV	kondenzátor	C150-054x183
		C6	10pF/500V	kondenzátor	C050-024x044
		C7	10pF/500V	kondenzátor	C050-024x044
		T1	FA163079	transformátor	DIP16
		K1	WEBP8-8	eth. zásuvka pro RJ-45	WEBP8-8
		Q1	20 MHz	krystal	HC49U-V
		D1	LED green	LED dioda	LED5MM
		D2	LED orange	LED dioda	LED5MM

Příloha C: Seznam součástek strana 2/2

List sché-matu	Obvod	Označení součástky	Hodnota	Popis	Pouzdro / rozměry (Eagle 4.0)
3	Programovací port	R9	47k	rezistor	0207/10
		R10	47k	rezistor	0207/10
		R11	100R	rezistor	0207/10
		R12	100R	rezistor	0207/10
		JP1	PINHEAD 2x5	konektor prog. kab. 2,54 mm	PINHD-2*5
		JP2	PINHEAD 2x5	konektro prog. kab. 2mm	PINHD-2*5 2mm
3	Napájení	K2	ARK2	konektor pro 2 vodiče	ARK2
		U1	7805	stabilizátor napětí 5 V	7805
		D3	1N4007	dioda	1N4007
		D4	LED green	LED dioda	LED5MM
		IC1G$2	Rabbit 2000	napájecí vývody	
		IC4G$2	RTL8019AS	napájecí vývody	
		R8	360R	rezistor	0207/10
		C8	330nF / 100V	kondenzátor	C050-050x075
		C9	100uF	elektrolytický kondenzátor	E3,5-8
		C10	100uF	elektrolytický kondenzátor	E3,5-8
		C11	100nF	kondenzátor	C050-050x075
		C12	100nF	kondenzátor SMD	C0805
		C13	100nF	kondenzátor SMD	C0805
		C14	100nF	kondenzátor SMD	C0805
		C15	100nF	kondenzátor SMD	C0805
		C16	100nF	kondenzátor SMD	C0805
		C17	100nF	kondenzátor SMD	C0805
		C18	100nF	kondenzátor SMD	C0805
		C19	100nF	kondenzátor SMD	C0805
		C20	100nF	kondenzátor SMD	C0805
		C21	100nF	kondenzátor SMD	C0805
		C22	100nF	kondenzátor SMD	C0805
		C23	100nF	kondenzátor SMD	C0805
		C32	100nF	kondenzátor SMD	C0805
3	Resetovací obvod	R17	8k2	rezistor	0207/10
		C30	100nF	kondenzátor	C050-050x075
		C31	100nF	kondenzátor	C050-050x075
		S1		tlacitko SMD	TLAC_SMD
		IC7	TL7705ACP	resetovací generátor	DIP8
4	Sériová FLASH (samostatná deska)	R1	470	rezistor SMD	M0805
		R2	3k3	rezistor SMD	M0805
		C1	100nF	kondenzátor SMD	C0805
		C2	10uF	elektrolytický kondenzátor	E1,8-4
		T1	BSS123	tranzistor N-MOSFET	SOP23
		U1	LE33CD	stabilizátor napětí 3,3 V	SO-8
		AT45DB041	AT45DB041	sériová paměť FLASH 512 KB	8-SOIC
		JP1	PINHEAD 2x5	dvouřadý konektor	PINHD-2*5

Abstract

Title: **Communication Interface of Measuring Device A2000 and Ethernet Bus**

Author: **Michal Mikl**

Supervizor: **Doc. Ing. František Zezulka, CSc.**

This graduate thesis is dealing with design and development of communication interface for „A2000“. The „A2000“ is an multifunctional power meter, which can measure most of electric values (like voltage, current, power, etc.), and by the communication interface, it could be handling and servicing by LAN / Internet connection.

The interface is designed as an external module and based on Rabbit 2000 microprocessor, supplied by other hardware. The I/O connection is provided by RS 232C (on the A2000 side) and 10 Base-T (on the Ethernet side).

Furthermore, an operations software for Rabbit 2000 mikroprocessor is described. It was developed in „Dynamic C“ - programming environment, that is similar to „C“ language. There are two tasks running together under the real-time OS. The first guarantees communication with A2000 and the second works as „HTTP“ an „FTP“ server. The HTTP server provides remote access to data through the „Java applet“, and the FTP server enables removal of data content from HTTP.

This thesis has been performed in a joint action with GMC - měřící technika, s.r.o. (a czech subsidiary of GOSEN-METRAWATT GMBH), that produces the A2000.

The developed interface can be used with other measuring appliances, which are able to comunicate through the RS 232C.