České Vysoké Učení Technické v Praze
Fakulta Elektrotechnická – Katedra řídicí techniky

Implementace síťového rozhraní Ethernet pro řídicí aplikace

Diplomová práce

Vedoucí práce:
Ing. Pavel Píša

Michal Sojka

Praha 2003

Na tomto místě bych rád poděkoval Ing. Pavlu Píšovi, za všechny jeho rady a náměty, které mi při realizaci práce dával, a také za to, že mi vůbec umožnil na takto zajímavém projektu pracovat. Můj dík směruje rovněž mojí rodině, která mě po celou dobu studia trpělivě podporovala.

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, software 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 22. května 2003 .....

Abstract

This work solves the problem of connection of control system based on Motorola MC68376 processor to the Ethernet network. Hardware part of Ethernet interface is based on CS8900A circuit and makes it possible to connect the system to 10BASE-T physical medium. Software has been written to support operating systems RTEMS and uClinux. The work also contains brief introduction to Ethernet (IEEE 802.3) networks.

Abstrakt

Tato práce se zabývá tvorbou hardwaru a softwaru pro připojení řídicí jednotky vybavené procesorem Motorola MC68376 k síti Ethernet. Hardwarová část založená na obvodu CS8900A umožňuje připojení fyzické vrstvy 10BASE-T. Softwarová podpora je vytvořena pro operační systémy RTEMS a uClinux. Práce rovněž obsahuje stručný popis technologie Ethernet (IEEE 802.3).

Obsah

Seznam symbolů a zkratek

Symboly

1001_b	binární zápis čísla,
$3F8_h$, 0x3F8	hexadecimální zápis čísla,
$\overline{\text{AEN}}$	signál aktivní v log. 0.
¬WR	negace logické hodnoty.

Zkratky

ARP Address Resolution Protocol.

AUI Attachment Unit Interface.

BDM Background Debug Mode.

BSP Board Support Package.

BST Boundary Scan Test.

CAN Controller Area Network.

CRC Cyclic Redundancy Check.

CSMA/CD Carrier Sense Multiple Access with Collision Detection.

DDD Data Display Debugger.

DHCP Dynamic Host Configuration Protocol

DMA Direct Memory Access.

DNS Domain Name Service

DPS Deska plošných spojů.

GCC GNU Compiler Collection.

GNU GNU's Not Unix.

GPL General Public License.

HTTP Hyper Text Transfer Protocol.

ICMP Internet Control Message Protocol

IP Internet Protocol

ISA Industry Standard Architecture

ISO International Organisation for Standartization.

ISQ Interrupt Status Queue.

JTAG Joint Test Action Group.

LAN Local Area Network – lokální počítačová síť.

LLC Logical Link Control.

MAC Medium Access Control.

MAU Medium Attachment Unit – jednotka připojení k médiu.

MII Media Independent Interface.

MMU Memory Management Unit.

NFS Network File System.

OSI Open System Interconnection.

OS Operační systém.

PLS Physical Layer Signaling.

PMA Physical Medium Attachment.

RPC Remote Procedure Call.

RTEMS Real-Time Executive for Multiprocessor Systems.

SIM System Integration Module.

SMTP Simple Mail Transfer Protocol.

TCP Transission Control Protocol.

TP Twisted Pair – typ kabelu (kroucená dvoulinka).

UDP User Datagram Protocol.

Úvod

Možnosti komunikace řídicích systémů s jejich okolím jsou v dnešní době často důležitější, než samotný výkon řídicího systému. Jednou z možností, jak realizovat komunikaci s okolím je vytvoření rozhraní k síti Ethernet. Ethernet je velmi rozšířená technologie, která se používá pro stavbu lokálních počítačových sítí.

Tato práce se zabývá návrhem a implementací připojení řídicího systému MO_CPU1 obsahujícího procesor Motorola MC68376 právě k síti Ethernet. Díky tomuto propojení bude možné ovládat činnost jednotky z jiného počítače, který se vůbec nemusí nacházet v blízkosti jednotky.

Práce je rozdělena do čtyřech kapitol. V kapitole 1 je zmíněna historie vývoje technologie Ethernet a jsou nastíněny základní principy této sítě. Popsány jsou funkce fyzické a spojovací vrstvy ISO/OSI modelu.

Kapitola 2 obsahuje popis dostupných obvodů pro realizaci ethernetového řadiče a dále stručný popis některých projektů, které řeší podobný problém jako tato práce. Na závěr je zdůvodněn výběr řešení, které je v této práci prezentováno.

Popis hardwarové konstrukce ethernetového rozhraní je v kapitole 3. Je zde popsán obvod CS8900A použitý v zapojení a stručně je popsána i jednotka MO_CPU1, ke které bude rozhraní připojeno. Nakonec uvádím vlastní návrh zapojení rozhraní.

Poslední, 4. kapitola se zabývá vývojem programového vybavení pro ethernetové rozhraní. Nejprve jsou popsány nástroje potřebné pro vývoj programů a pak následuje popis testovací aplikace *mo_ether* a popis tvorby podpory pro operační systémy RTEMS a uClinux.

Součástí práce je i přiložené CD se zdrojovými kódy.

Kapitola 1

Síť Ethernet

Ethernet je technologie pro budování lokálních počítačových sítí (LAN[1]), která zahrnuje dvě nejnižší vrstvy (fyzickou a spojovací) ISO[2] OSI[3] modelu. Mezi její hlavní rysy patří používání sdíleného média s přístupovou metodou CSMA/CD[4].

V běžné mluvě se často zaměňují pojmy Ethernet a IEEE 802.3. Standard IEEE 802.3 je novější (viz sekce 1.1), a zahrnuje v sobě zahrnuje i původní standard pro Ethernet. Navíc obsahuje mnoho rozšíření (např. tzv. Fast Ethernet). Ethernet i IEEE 802.3 mají téměř stejný formát rámce (viz 1.2.1), takže spolu mohou koexistovat na jednom společném médiu.

V této práci budu slovem Ethernet nazývat sítě IEEE 802.3. V místech, kde bude zmiňován původní Ethernet to výslovně uvedu.

1.1 Historie

Historie Ethernetu sahá až do roku 1972, kdy Bob Metcalfe a jeho spolupracovníci z Palo Alto Research Center ze společnosti Xerox Corporation navrhli síťovou technologii pro propojení počítačů *Alto*. Tato technologie se nazývala *Alto Aloha Network* a přenosová rychlost byla 2,94 Mbit/s.

V roce 1973 byl název změněn na *Ethernet*, aby lépe vystihoval fakt, že se jedná o univerzální technologii určenou ne jen pro počítače Alto. V roce 1976 byla technologie rozšířena o *příposlech nosné* (carrier sense) a byla vybudována síť propojující 100 počítačů kabelem o délce 1 km.

Xerox Ethernet byl velmi úspěšný a tak v roce 1980 společnosti Digital Equipment Corporation, Intel Corporation a Xerox vydali tzv. *Blue Book Standard* pro 10 Mbit Ethernet (verze 1), který se též nazýval (podle prvních písmen vydávajících společností) *DIX Ethernet*. V roce 1985 byl standard mírně rozšířen a vydán jako *Ethernet II*. Tento standard se stal základem pro IEEE 802.3.

[1] Local Area Network
[2] International Organisation for Standartization
[3] Open System Interconnection
[4] Carrier Sense Multiple Access with Collision Detection

V roce 1985 byl schválen standard IEEE 802.3 10Base5, v roce 1990 10Base2, v roce 1991 10BaseT, v letech 1994–1995 10BaseF. Standard IEEE 802.3u pro 100 Mbit/s Ethernet (Fast Ethernet) byl vydán v roce 1995.

O historii Ethernetu se píše v [8] a [16].

1.2 Spojovací vrstva

Standard [9] rozděluje spojovací vrstvu ISO/OSI modelu na tři podvrstvy:

- LLC[5]
- MAC control (volitelná)
- MAC[6]

V této sekci se stručně zmíním o MAC podvrstvě, která je pro pochopení funkce Ethernetu podstatná. Tato vrstva je zodpovědná za *zapouzdření dat*, a definuje formát Ethernetového rámce (sekce 1.2.1), způsob adresování (1.2.2) a ochranu proti chybám při přenosu. Rovněž se stará o *správu přístupu k médiu*, což zahrnuje přidělování média a řešení kolizí (více v sekci 1.2.3).

1.2.1 Formát rámce

Formát ethernetového rámce je znázorněn na obrázku 1.1. Jednotlivá pole mají následující význam:

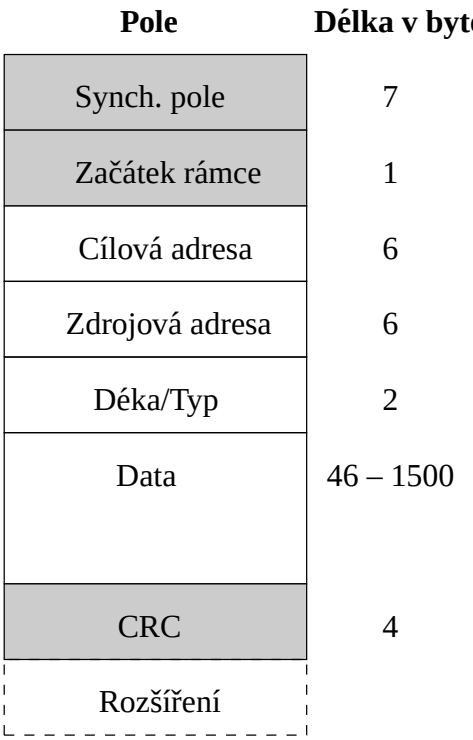
- *Synchronizační pole* slouží k synchronizaci obvodů odesílatele a příjemce a tvoří jej střídající se bity 1 a 0.
- Značka *začátku rámce* je byte s hodnotou 10101011_b.
- *Cílová adresa* určuje stanici (stanice) které je rámec určen.
- *Zdrojová adresa* identifikuje stanici odesílající rámec. Formát adres je popsán v 1.2.2.
- Pole *Délka/Typ* má dva významy v závislosti na hodnotě.

  Hodnoty větší nebo rovny 1536 udávají typ protokolu vyšší vrstvy, který je rámcem přenášen. Například pro IP datagramy je zde uloženo 800_h. Tento význam má pole ve standardech Ethernet verze 1 a 2 i IEEE 802.3.

  Je-li hodnota menší než 1536 (600_h), pak má význam délky dat obsažených v rámci. Tento význam zavedl standard IEEE 802.3.

[5] Logical Link Control
[6] Medium Access Control

- Pole *Data* obsahuje libovolná data přenášená rámcem. Pokud je jejich délka menší než 46 bytů, musí být data na tuto délku doplněna.
- *CRC*[7] je kontrolní součet sloužící ke kontrole správnosti přijatého rámce. Jeho hodnota je počítána z těchto polí: zdrojová a cílová adresa, délka/typ a data.
- *Rozšíření* je definováno v IEEE 802.3 a používá se jen pro přenosovou rychlost 100 Mbit/s. Pole slouží k prodloužení vysílací doby krátkých rámců na minimální hodnotu stanovenou standardem.



Obrázek 1.1: Formát rámce IEEE 802.3 a Ethernetu II. Šedá pole většinou generuje hardware, hodnoty bílých polí obvykle poskytuje software.

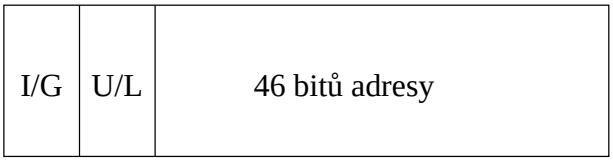
1.2.2 Adresování

Pro adresování stanic v síti se používají 48 bitové adresy. Adresy se dělí na individuální a skupinové. Individuální adresy jsou adresy konkrétních stanic, skupinové adresy označují několik stanic najednou. Speciální skupinovou adresou je všesměrová adresa, jejíž všechny bity mají hodnotu 1.

Formát ethernetové adresy je na obrázku 1.2. První, nejméně významný bit (I/G), určuje zda se jedná o individuální adresu (hodnota 0) nebo skupinovou adresu (hodnota

[7] Cyclic Redundancy Check

1). Druhý bit (U/L) udává, je-li adresa globální (univerzální) nebo lokální. Zbytek adresy tvoří 46 bitový identifikátor. U globálních individuálních adres je prvních 24 bitů celé adresy identifikátor organizace, který je přidělován Registrační autoritou IEEE.



Obrázek 1.2: Formát ethernetové adresy. Bit I/G rozlišuje mezi individuální a skupinovou adresou, bit U/L rozlišuje mezi globální a lokální adresou.

1.2.3 Přístup k médiu

Ethernet je síť založená na přístupové metodě CSMA/CD[8]. Všechny stanice sdílejí jedno přenosové médium (MA) a naslouchají probíhá-li přenos mezi jinými stanicemi (CS). Pokud začne vysílat více stanic najednou, je detekována *kolize* (CD).

Chce-li stanice vyslat rámec, musí postupovat podle algoritmů uvedených v [9] v sekci 4.2.3. Tyto algoritmy lze pro half duplex režim zjednodušeně shrnout do těchto kroků:

1. Stanice neustále monitoruje stav kanálu (obsazen/volný). Ihned po změně stavu z obsazen na volný začne odměřovat čas povinné mezirámcové mezery.
2. Pokud není kanál volný nebo neuplynula mezirámcová mezera, čekej.
3. Začni vysílat. Není-li během přenosu detekována kolize (způsob detekce závisí na fyzické vrstvě), je přenos úspěšně ukončen. Při signalizované kolizi pokračuj krokem 4.
4. Pokračuj ještě daný čas ve vysílání, aby všechny stanice detekovaly kolizi (jam signál).
5. Vygeneruj náhodné celé číslo r, pro které platí $0 \leq r < 2^k$, kde $k = \min(n, 10)$ a n je pořadové číslo pokusu o vyslání daného rámce. Pak čekej r časových jednotek.
6. Je-li kanál volný, pokračuj krokem 3.
7. Není-li vyčerpán maximální počet pokusů o odeslání rámce, pokračuj krokem 2. V opačném případě signalizuj chybu.

[8] Carrier Sense Multiple Access with Collision Detection

1.3 Fyzická vrstva

Způsob přenosu signálu mezi stanicemi je určen specifikací fyzické vrstvy. Ta udává, jaké je použito přenosové médium, jaké konektory a jak je signál kódován (1.3.1). V [9] je fyzická vrstva ISO/OSI modelu rozdělena na několik podvrstev. V případě přenosové rychlosti 10 Mbit to jsou podvrstvy:

- PLS[9]
- PMA[10]

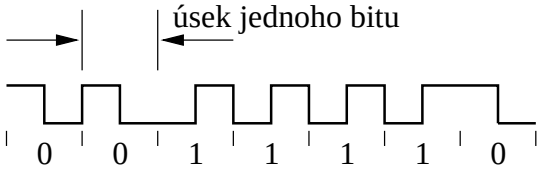
PLS se stará o kódování signálu a PMA převádí výstup z PLS na konkrétní přenosové médium. Podvrstva PMA je realizována obvody souhrnně označovanými jako MAU[11].

Standard [9] specifikuje mnoho typů fyzických vrstev. Nejvíce se zde budu věnovat typu 10BASE-T, který používám ve své práci. Stručně se zmíním také o některých dalších typech.

1.3.1 Kódování bitů

Bity přenášené po fyzickém médiu jsou podvrstvou PLS kódovány kódem *Manchester II*. V tomto kódu je doba pro vysílání jednoho bitu rozdělena na dvě poloviny. V první polovině se vysílá negovaná hodnota vysílaného bitu a druhé polovině přímá hodnota. V každém přenášeném bitu se tedy vyskytuje hrana, podle které se můžou synchronizovat přijímací obvody přijímací stanice.

Příklad binárního signálu zakódovaného do kódu Manchester II je na obrázku 1.3.



Obrázek 1.3: Ukázka signálu v kódu Manchester II.

1.3.2 10BASE5

Tento typ fyzické vrstvy byl používán původním Ethernetem verze 1 a 2. Přenosové médium je tzv. *tlustý koaxiální kabel* s impedancí 50 Ω na frekvenci 10 MHz. Kabel musí být na koncích opatřen 50 Ω zakončovacími rezistory a jeho maximální délka je 500 m. Vysílací stanice budí kabel proudem a přijímací stanice vyhodnocují napětí na kabelu. Přenosová rychlost je 10 Mbit/s.

[9] Physical Layer Signaling
[10] Physical Medium Attachment
[11] Medium Attachment Unit

1.3.3 10BASE2

Typ 10BASE2 je téměř shodný s 10BASE5, až na to, že používá *tenký koaxiální kabel* RG 58. Maximální délka jednoho segmentu je 185 m.

1.3.4 10BASE-T

Typ 10BASE-T umožňuje přenos dat rychlostí 10 Mbit/s. Použité médium je tzv. TP[12] kabel (kroucená dvoulinka) s impedancí 85–111 Ω. Maximální délka kabelu bez opakovače je 100 m.

Nedílnou součástí sítě 10BASE-T bývá i opakovač (specifikovaný rovněž v [9]), který umožňuje propojení více jak dvou stanic. Jednoduchý opakovač se často nazývá *hub*. Dnes se stále častěji používají tzv. přepínací opakovače (switching hub, switch), které zabraňují vzniku kolizí (viz 1.2.3).

Pro přenos dat se používají dva páry vodičů, které jsou buzeny diferenciálním signálem. Špičkové diferenciální napětí na výstupu budiče má být při použití zakončovacího odporu 100 Ω mezi 2,2 a 2,8 V. Obvody stanice by měly být elektricky odděleny od všech vodičů sítě, tak, aby po dobu 60 s vydržely na síti stejnosměrné napětí 2250 V nebo střídavé efektivní napětí 1500 V. K oddělení se používá miniaturní transformátor.

Obdélníkový signál (kód Manchester II) který vstupuje do MAU musí být filtrován, aby neobsahoval vyšší harmonické. K filtraci se používají filtry, které jsou buď součástí hybridního transformátoru, nebo jsou přímo součástí čipu (jako v případě CS8900).

V době, kdy není obvody MAU vysílán žádný rámec, je generován signál TP_IDL, který je často označován jako *linkové pulzy*. Tento signál slouží k detekci propojení stanic a k detekci polarity připojeného vedení. Signál TP_IDL je tvořen krátkým (250 ns) pulzem, nasledovaným 16 ms $\pm$ 8 ms ticha.

Podrobnější popis 10BASE-T uvádí např. [2].

1.3.5 10BASE-F

Umožňuje přenos dat optickým vláknem rychlostí 10 Mbit/s až na vzdálenost 2 km.

1.3.6 100BASE-T

100BASE-T je standard fyzické vrstvy pro přenosovou rychlost 100 Mbit/s. Tento standard definuje jiné rozdělení fyzické vrstvy na podvrstvy a používá jinou terminologii. Jeho součástí jsou například podvrstvy 100BASE-TX pro přenos dat po TP kabelu, 100BASE-FX pro přenos po optických vláknech..

[12] Twisted Pair

Kapitola 2

Popis řešeného problému

Úkolem této práce je návrh a implementace ethernetového rozhraní k řídicí jednotce MO_CPU1. V této kapitole jsou stručně popsány některé integrované ethernetové řadiče (2.1) a některé projekty, které řeší podobný problém jako tato práce (2.2). Na závěr je zdůvodněna volba řešení použitého v této práci.

2.1 Integrované ethernetové řadiče

V této sekci budou porovnány některé dostupné integrované obvody plnící funkci ethernetového řadiče. V úvahu budou brány zejména vlastnosti související s aplikací do vloženého (embedded) zařízení.

2.1.1 Cirrus Logic CS8900A

Tento obvod (viz [4]) má rozhraní pro připojení k ISA[1] sběrnici (8 a 16 bitový režim). Z ethernetových fyzických vrstev lze přímo připojit 10BASE-T. MAU pro jiné fyzické vrstvy lze připojit k AUI[2] rozhraní.

Registry obvodu lze namapovat do adresového prostoru ISA sběrnice a přistupovat k nim přímo, nebo lze pro přístup k registrům použít osm 16 bitových I/O portů. Na čipu jsou integrovány 4 kB RAM pro přijímané a odesílané rámce. Obvod obsahuje všechny analogové obvody pro připojení fyzické vrstvy 10BASE-T.

Jsou dostupné zdrojové kódy ovladačů pro OS[3] Linux a RTEMS[4].

2.1.2 Realtek RTL8019AS

Obvod RTL8019AS je velmi podobný CS8900A (viz [15]). Má rovněž rozhraní pro sběrnici ISA a navíc má interface pro fyzickou vrstvu 10BASE2. Na čipu je integrováno

[1] Industry Standard Architecture
[2] Attachment Unit Interface
[3] Operační systém
[4] Real-Time Executive for Multiprocessor Systems

16 kB RAM. Obvod neobsahuje analogové filtry pro 10BASE-T a musí se použít hybridní transformátor s filtry.

Softwarově je obvod kompatibilní se standardem NE2000, což zaručuje dostupnost ovladačů.

2.1.3 Standard Microsystems Corporation LAN91c111

LAN91c111 je ethernetový řadič určený speciálně pro vložené aplikace. Podporuje přenosové rychlosti 10 a 100 Mbit/s a fyzické vrstvy 10BASE-T a 100BASE-TX. Na čipu je integrováno 8 kB RAM pro uchovávání odesílaných a přijímaných rámců. Sběrnice obvodu je navržena tak, aby jej bylo možno jednoduše připojit ke 8, 16 i 32 bitovým sběrnicím různých procesorů. Mezi podporované procesory patří i Motorola 683xx a další jsou například ARM, SH, Power PC, Coldfire…

Pro Linux jsou dostupné zdrojové kódy ovladačů. Více informací lze nalézt v [17].

2.1.4 Asix AX88796

Tento obvod, popsaný v [1], umožňuje přenos dat rychlostmi 10 a 100 Mbit/s. Je registrově kompatibilní se standardem NE2000, což zaručuje dostupnost ovladačů. Obvod lze pomocí 8 nebo 16 bitvé sběrnice jednoduše připojit k procesorům MCS-51, 80186, M68K a ke sběrnici ISA. Obvod standardně umožňuje připojení fyzických vrstev 10BASE-T a 100BASE-TX. Pomocí rozhraní MII[5] jdou připojovat další. Obvod také obsahuje rozhraní Standard Printer Port pro použití v tiskových serverech.

2.2 Podobné projekty

V této sekci se zmíním o některých podobných projektech a porovnám je s řešením, které je prezentováno v následujících kapitolách této práce.

2.2.1 Web51

Český projekt Web51 [2] řeší připojení procesoru z rodiny Intel x51 k Ethernetu. Připojení je relizováno síťovým řadičem Realtek RTL8019AS zapojeným v 8 bitovém režimu. Cílem projektu není vytvoření výkoného systému, který by byl schopen obsluhovat objemné datové toky. Projekt je zaměřen na jednoduché aplikace, kde je hlavním kritériem nízká cena a jednoduchost stavby zařízení.

Součástí projektu Web51 jsou softwarové moduly, které implementují:

- jednoduchý HTTP[6] server.
- jednoduchý souborový systém pro potřeby HTTP serveru.

[5] Media Independent Interface
[6] Hyper Text Transfer Protocol

- interpret P-kódu, díky němuž je umožněno ovládat činnost procesoru z jiné stanice.
- TCP[7] stack implementucjící částečně protokol TCP.
- SMTP[8] server.

Všechny tyto funkce lze implementovat vyrazně jednodušeji na systému MO_CPU1 při použití univerzálního OS jako např. uClinux (viz 4.4).

Porovnání celkové koncepce Web51 a MO_CPU1

Jádrem projektu Web51 je 8 bitový procesor ATMEL 89C8252, který obsahuje 8 kB FLASH, 256 B RAM, 2 kB EEPROM. Naproti tomu MO_CPU1 je založen na 32 bitovém procesoru Motorola MC68376 a systém může obsahovat až 2 MB RAM a až 2 MB FLASH paměti.

Z parametrů procesoru 89c8252 je vidět, že programové vybavení projektu Web51 musí být velmi jednoduché a šité na míru danému systému. Procesor MC68376 a velké množství paměti naopak umožňuje využít univerzální operační systémy či exekutivy a snadno implementovat další funkce.

2.2.2 Embedded Ethernet.com

Projekt [6] se zabývá návrhem ethernetového rozhraní pro téměř libovolný procesor. Zdokumentováno je připojení k mikrokontrolerům Atmel ATmega103 nebo AT90S8515 a k PIC16C74. V zapojení po použit obvod CS8900A v 8 bitovém režimu. Hlavní předností prezentovaného návrhu jsou malé rozměry celého rozhraní. Pro tento projekt rovněž není prioritou výkonost celého systému.

2.2.3 Ethernut

Ethernut je otevřený hardwarový a softwarový projekt zabývající se návrhem a stavbou ethernetového rozhraní pro vložené aplikace. Základem hardwarové části projektu je deska s 8 bitovým RISCovým procesorem Atmel ATmega128 a ethernetovým řadičem RTL8019AS.

Softwarová část projektu zahrnuje jednoduchý operační systém Nut/OS a TCP/IP stack Nut/Net. Nut/OS je malý operační systém reálného času, který mimo jiné podporuje kooperatvní multithreading, periodické a jednorázové časovače a dynamickou správu paměti. Nut/Net je TCP/IP stack vystavěný nad Nut/OS podporující protokoly ARP[9], IP[10],

[7] Transission Control Protocol
[8] Simple Mail Transfer Protocol
[9] Address Resolution Protocol
[10] Internet Protocol

ICMP[11], UDP[12] a TCP. Na vyšších vrstvách jsou podporovány protokoly DHCP[13], DNS[14] a HTTP.

Procesor ATmega128 má na čipu integrováno 128 kB FLASH paměti a 4 kB EEPROM. Deska Ethernut obsahuje navíc 32 kB SRAM. To stačí na jednodušší jednoúčelové aplikace. Výhodnou (oproti např. Web51) je možnost použít při vývoji kompilátor jazyka C založený na překladači GCC[15].

2.2.4 uCsimm

Projekt uCsimm [7] byl navržen speciálně pro použití s operačním systémem uClinux (viz 4.4). Po hardwarové stránce je založen na procesoru Motorola DragonBall 68EZ328 a ethernetovém řadiči CS8900A. Systém je vybaven 2 MB paměti FLASH a 8 MB DRAM. Dále systém umožňuje připojení grafického monochromatického displeje.

Dostupná velikost paměti dovoluje použití už zmíněného OS uClinux pro téměř libovolné aplikace nebo OS RTLinux pro aplikace reálného času.

2.3 Navržené řešení

Při výběru ethernetového řadiče byl zvolen obvod CS8900A, který je velmi často používán v podobných projektech jako tento. K dispozici jsou zdrojové kódy ovladačů pro OS RTEMS a uClinux a obvod má velmi podrobnou dokumentaci. V manuálu k obvodu [3] je dokonce přímo popsáno propojení s procesorem firmy Motorola.

Použití řadiče pro přenosovou rychlost 100 Mbit/s by nemělo smysl, neboť výkon procesoru MC68376, který je použit v jednotce MO_CPU1, by nedostačoval ke zpracování tak velkých datových toků.

[11] Internet Control Message Protocol
[12] User Datagram Protocol
[13] Dynamic Host Configuration Protocol
[14] Domain Name Service
[15] GNU Compiler Collection

Kapitola 3

Hardware

3.1 Obvod CS8900A

CS8900A je levný ethernetový řadič optimalizovaný pro připojení na sběrnici ISA. Jeho vysoký stupeň integrace eliminuje nutnost použití většího množství dalších externích součástek. Obvod obsahuje na čipu integrovanou paměť RAM, obvody MAU pro fyzickou vrstvu 10BASE-T včetně analogových filtrů a budiče ISA sběrnice. Díky vysokému stupni integrace je mimořádně vhodný pro vložené aplikace, neboť minimální plocha potřebná pro realizaci kompletního ethernetového rozhraní je zhruba 10 cm^2.

V této sekci bude stručně popsána struktura obvodu (3.1.1), jeho vývody (3.1.2) a funkce obvodu (3.1.3). Podrobnější popis obvodu je v [4].

3.1.1 Základní struktura obvodu

Blokové schema obvodu je na obrázku 3.1. V následujících odstavcích jsou popsány části důležité pro zapojení použité v této práci.

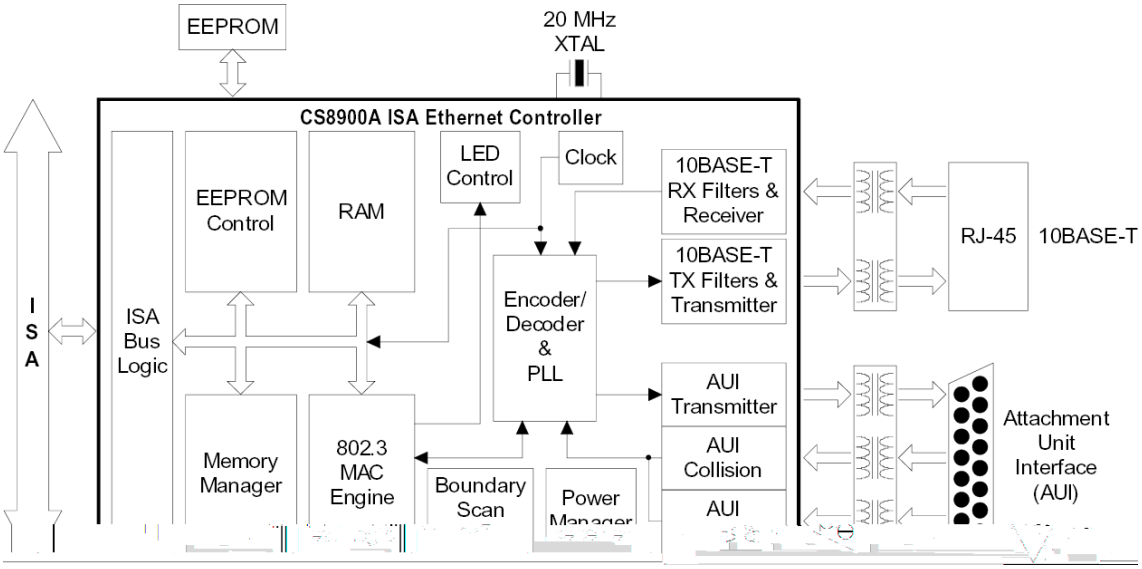
Rozhraní ISA sběrnice Pomocí tohoto rozhraní lze obvod jednoduše připojit k ISA sběrnici. Vzhledem k jednoduchosti této sběrnice není problém tento obvod připojit k jiných sběrnicím, jako například ke sběrnici jednotky MO_CPU1.

Součástí rozhraní jsou 24 mA budiče sběrnice a konfigurace obvodu umožňuje výběr ze čtyřech signálů přerušení a třech DMA[1] kanálů.

Integrovaná paměť Obvod má přímo na čipu integrovány 4 kB paměti pro přijímané a odesílané rámce. Díky této paměti nemusí být součástí návrhu ethernetového rozhraní další paměťové obvody zabírající místo na DPS[2]. K interní paměti lze přistupovat jednou ze tří metod (viz 3.1.4):

- přes I/O rozhraní,

[1] Direct Memory Access
[2] Deska plošných spojů



Obrázek 3.1: Blokové schéma obvodu CS8900A.

- přes paměťové rozhraní,
- pomocí DMA kanálu.

Ethernet MAC řadič Tato část obvodu zajišťuje všechny potřebné funkce MAC podvrstvy vyžadované standardem IEEE 802.3. Obstarává kompletní příjem a odesílání ethernetových rámců včetně generování a detekce synchronizačního pole, detekce kolizí, výpočtu a kontroly CRC. Mezi volitené funkce MAC řadiče patří automatické opakování odeslání rámce při detekci kolize a doplňování krátkých rámců na minimální délku.

Rozhraní fyzické vrstvy Toto rozhraní obsahuje Manchester II kodér a dekodér, obvody pro synchronizaci časování a budiče LED diod pro indikaci stavu obvodu.

Obvody fyzické vrstvy 10BASE-T zahrnují budiče a přijímače signálu včetně všech analogových filtrů. K CS8900A pak stačí přímo připojit pouze levný izolační transformátor. Podporovány jsou 100, 120 a 150 Ω kabely. Obvody rozvněž detekují a napravují obrácenou polaritu přijímacího a vysílacího páru vodičů.

3.1.2 Popis vývodů

Obvod CS8900A je vyráběn ve 100 vývodovém pouzdru TQFP. Jeho vývody lze rozdělit do několika kategorií, které jsou popsány v následujících tabulkách. U každého vývodu je vyznačen směr chodu signálu: ← vstup, → výstup. Seznam všech vývodů včetně jejich fyzického umístění je patrný z obrázku 3.3.

Vývody rozhraní ISA sběrnice jsou v tabulce 3.1, vývody rozhraní 10BASE-T v tabulce 3.2 a ostatní vývody v tabulce 3.3. Další (v mém zapojení nepoužité) vývody spadají do těchto kategorií: rozhraní pro EEPROM a Boot PROM a rozhraní AUI.

Signál	Směr	Popis
SA0–19	↔	Signály adresové sběrnice.
SD0–15	↔	Signály datové sběrnice.
$\overline{\text{MEMR}}$, **$\overline{\text{MEMW}}$**	←	Spuštění operace čtení resp. zápisu z/do obvodu v paměťovém režimu.
$\overline{\text{IOR}}$, **$\overline{\text{IOW}}$**	←	Spuštění operace čtení resp. zápisu z/do obvodu v I/O režimu.
$\overline{\text{SBHE}}$	←	Slouží k přepnutí obvodu do 16 bitového režimu.
INTRQ0–2	→	Signalizuje žádost přerušení.
AEN	←	Používá se při DMA přenosu. V testovacím režimu (viz 3.1.5) má jiný význam.
RESET	←	Reset celého obvodu.
$\overline{\text{CHIPSEL}}$	←	V paměťovém režimu signalizuje platnou adresu na adresové sběrnici.

Tabulka 3.1: CS8900A – vývody rozhraní ISA sběrnice.

Signál	Směr	Popis
TXD+/TXD-	→	Výstup diferenciálního budiče 10BASE-T.
RXD+/RXD-	←	Diferenciální vstup pro příjem z 10BASE-T.

Tabulka 3.2: CS8900A – vývody rozhraní 10BASE-T.

3.1.3 Funkce obvodu

V normálním režimu vykonává obvod dvě funkce: příjem ethernetových rámců a vysílání ethernetových rámců. Před začátkem plnění těchto funkcí musí být obvod nakonfigurován.

Konfigurace Při konfiguraci obvodu je nutno specifikovat bázovou adresu pro přístup v paměťovém režimu, MAC adresu stanice, typ použité fyzické vrstvy a typy rámců, které mají být přijímány.

Konfiguraci lze provést při resetu obvodu načtením konfiguračních údajů z paměti EEPROM, nebo ji lze provést z hostitelského procesoru.

Odesílání rámců Odesílání rámců se skládá ze dvou fází. V první fázi je rámec přesunut hostitelským procesorem do vyrovnávací paměti na čipu. Přesun začíná zápisem

Signál	Směr	Popis
XTAL1,2	↔	Vývody pro připojení 20 MHz krystalu nebo generátoru hodinového signálu.
$\overline{\text{LINKLED}}$	→	Výstup pro připojení LED diody, která indikuje platné linkových pulzy na rozhraní 10BASE-T. Tento výstup může být ovládán i programově (viz 4.2.3).
$\overline{\text{LANLED}}$	→	Výstup pro připojení LED diody. Při vysílání nebo příjmu rámce přejde na 6 ms do stavu log. 0.
$\overline{\text{TEST}}$	←	Log. 0 přepne obvod do tzv. Boundary Scan testovacího módu (viz 3.1.5).
RES	←	Vstup pro připojení 4,99 kΩ referenčního rezistoru sloužícího ke kalibraci vnitřních analogových obvodů.
DVDD1–4	←	Napájení digitálních obvodů.
DVSS1–4	←	Zem digitálních obvodů.
AVDD1–3	←	Napájení analogových obvodů.
AVSS0–4	←	Zem analogových obvodů.

Tabulka 3.3: Ostatní vývody CS8900A.

příkazu k odeslání rámce, kde je specifikováno zda se má rámec začít odesílat po přesunu 5, 381, 1021 nebo všech bytů a zda-li se mají automaticky doplňovat krátké rámce a generovat CRC. Následně se zapíše délka rámce a je-li dostatek volné vyrovnávací paměti, pokračuje se přenosem vlastních dat rámce.

Ve druhé fázi je obvodem rámec odeslán do sítě. Obvod vygeneruje synchronizační pole a značku začátku rámce. Pak začne vysílat data předaná hostitelským procesorem (zahrnující adresu příjemce, odesílatele a typ rámce), která případně doplní na minimální délku a je-li to požadováno, připojí i CRC.

Příjem rámců Příjem rámců se stejně jako jejich odesílání dělí na dvě fáze. Ve fázi první je obvodem detekováno synchronizační pole a značka začátku rámce, které slouží jen pro synchronizaci časovacích obvodů. Následně je přečtena cílová adresa, která je zpracována adresovým filtrem. Pokud adresa splňuje kritéria specifikovaná v adresovém filtru, je rámec uložen do vyrovnávací paměti na čipu a je zkontrolováno CRC. V závislosti na konfiguraci může být o přijatém rámci informován hostitelský procesor.

V duhé fázi jsou data přenášena z vyrovnávací paměti na čipu ke zpracování hostitelským procesorem. Přenos lze provést v I/O režimu opakovaným čtením datového portu, v paměťovém režimu blokovým přenosem, nebo s využitím DMA kanálu.

3.1.4 Architektura PacketPage

Architektura PocketPage, na které je obvod CS8900A založen, umožňuje snadný přístup hostitelského procesoru k vnitřním registrům a vyrovnávací paměti obvodu. Jádrem této architektury je paměť o velikosti 4 kB, která je umístěna přímo na čipu a slouží jednak

jako vyrovnávací paměť pro dočasné uložení přijímaných a vysílaných rámců, a jednak obsahuje registry obvodu. Přístup k této paměti je možný buď pomocí *I/O režimu* nebo pomocí *paměťového režimu*, které jsou popsány dále. Přístup k přijímaným a odesílaným rámcům je možný i pomocí DMA kanálu.

I/O režim V tomto režimu se k registům a k vyrovnávací paměti obvodu přistupuje pomocí osmi 16 bitových portů, které jsou uvedeny v tabulce 3.4. Tento režim je vždy aktivní a po resetu obvodu je bázová adresa portů 300_h. Při konfiguraci obvodu lze tuto adresu změnit.

Ofset	Přístup	Popis
0000_h	RW	Příjem/odesílání dat (port 0)
0002_h	RW	Příjem/odesílání dat (port 1)
0004_h	W	TxCMD (příkaz pro odeslání rámce)
0006_h	W	TxLength (délka dat)
0008_h	R	ISQ (fronta žádostí o přerušení)
$000A_h$	RW	Ukazatel do PacketPage paměti.
$000C_h$	RW	Data PacketPage paměti (port 0).
$000E_h$	RW	Data PacketPage paměti (port 1).

Tabulka 3.4: Popis I/O portů pro přístup v I/O režimu. Sloupec přístup udává, je-li port pro čtení (R) nebo zápis (W).

Pro přístup k portům musí být na vývodech SA0–15 platná adresa, **AEN** v log. 0 a aktivní $\overline{\text{IOW}}$ nebo $\overline{\text{IOR}}$.

Paměťový režim Po aktivaci paměťového režimu jsou všechny registry obvodu a vyrovnávací pamět namapovány do 4 kB souvislého bloku paměti hostitelského procesoru. Bázová adresa tohoto bloku je konfigurovatelná a musí začínat na adrese dělitelné 1000_h. K obvodu se přistupuje v paměťovém režimu, pokud na vývodech SA0–19 je platná adresa z nakonfigurovaného rozsahu, $\overline{\text{CHIPSEL}}$ je v log. 0 a $\overline{\text{MEMW}}$ nebo $\overline{\text{MEMR}}$ je aktivní.

3.1.5 Testovací režimy

Obvod CS8900A je vybaven několika diagnostickými režimy, ve kterých lze testovat jednotlivé části obvodu, či správnost propjení obvodu s externími součástkami. U moderních obvodů jsou tyto režimy ovládány většinou pomocí rozhraní JTAG[3] (IEEE 1149.1). Obvod CS8900A standard JTAG nepodporuje a nabízí vlastní rozhraní.

Testy lokálních smyček a kolizí V závislosti na nastavení třech bitů v konfiguračních registrech přejde obvod do jednoho z testovacích režimů. Lze například propojit výstup Manchester kodéru se vstupem dekodéru a testovat, zda odeslané rámce budou

[3] Joint Test Action Group

správně přijaty (při full-duplexním spojení), nebo zda bude vygenerována kolize (half-duplex). Podobně lze testovat i analogové obvody, spojíme-li externí propojkou vývody TXD+ s RXD+ a TXD- s RXD-.

Boundary Scan test BST[4] umožňuje efektivně testovat propojení obvodu CS8900A s deskou plošných spojů či s dalšími obvody. Rovněž ho lze využít k nalezení zkratů. BST se aktivuje nastavením vývodu $\overline{\text{TEST}}$ do stavu log. 0. Průběh BST je řízen signálem AEN a skládá se ze dvou fází. První, tzv. *výstupní* fáze, slouží k testování digitálních výstupů a obousměrných vývodů. V druhé, *vstupní* fázi, se testují digitální vstupy a obousměrné vývody.

Při přechodu obvodu do režimu BST jsou všechny výstupy nastaveny do stavu vysoké impedance. Vývod AEN slouží během testu jako hodinový vstup pro časování testu. Ve výstupním cyklu jsou s každou sestupnou hranou AEN postupně uváděny jednotlivé signály do stavu log. 0. Ve vstupním cyklu jsou logické hodnoty čtené z jednotlivých vstupů kopírovány na vývod EEDataOut.

3.2 Řídicí jednotka MO_CPU1

Jádrem jednotky MO_CPU1, ke které je v této práci navrhováno ethernetové rozhraní, je procesor Motorola MC68376, který je stručně popsán v 3.2.1. Jednotka obsahuje tyto součásti:

- až 2 MB SRAM a 2 MB FLASH,
- sériové rozhraní,
- rozhraní BDM[5] (sekce 3.2.1),
- rozhraní pro komunikaci CAN[6],
- obvod reálného času,
- rozhraní sběrnice I^2C.

Blokové schéma jednotky je uvedeno v příloze A. Jednotku je možno rozšiřovat o další zařízení, která se připojují pomocí systémové sběrnice popsané v sekci 3.2.2.

[4] Boundary Scan Test
[5] Background Debug Mode
[6] Controller Area Network

3.2.1 Procesor MC68376

Procesor Motorola MC68376 je 32 bitový procesor s jádrem CPU32, který je určen speciálně pro řídicí aplikace. Kromě procesorového jádra je v obvodu integrováno velké množství rozšiřujících modulů jako například modul časovacího koprocesoru TPU, modul sériového rozhraní QSM či modul A/D převodníků QADC. Podrobný popis celého procesoru je v [11]. Popis jádra CPU32 včetně instrukčního souboru uvádí [10]. Dokument [14] obsahuje historii vývoje procesorů firmy Motorola a popis procesorů 683xx v českém jazyce.

BDM

BDM je speciální režim procesorů založených na jádře CPU32. Po přechodu do tohoto režimu není prováděn kód uložený v paměti a procesor je možné plně ovládat pomocí vnějšího sériového rozhraní. Lze modifikovat registry procesoru, přistupovat ke všem periferiím apod. Při použití jednoduchého rozhraní, připojeného na standardní paralelní port, lze činnost procesoru ovládat z počítače PC. Při použití debuggeru GDB (popsáno v 4.1.2) lze pohodlně ladit veškeré programy.

3.2.2 Sběrnice

Sběrnice jednotky MO_CPU1 slouží pro připojení periferních zařízení k procesoru MC68376. V podstatě se jedná o rozvedenou sběrnici procesoru, která je rozšířena o několik signálů.

Rozšíření sběrnice má na starosti programovatelný logický obvod ISP22V10. Ten generuje signály RD a WR na základě signálů $\overline{\text{DS}}$ a R/$\overline{\text{W}}$ a rozšiřuje počet signálů chip-select. Signály na jednotlivých konektorech sběrnice jsou popsány v příloze A.

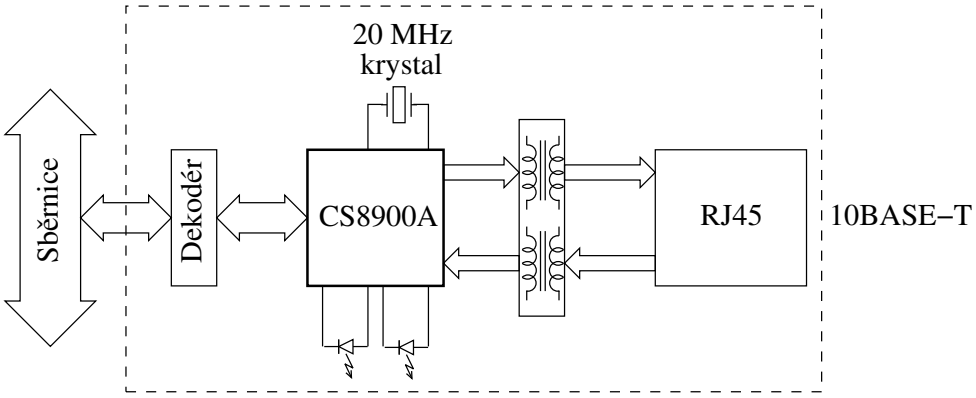
3.3 Popis zapojení ethernetového rozhraní MO_ETH

Jádrem ethernetového rozhraní (MO_ETH) pro řídicí jednotku MO_CPU1 je obvod CS8900A. Jeho zapojení vychází zejména z doporuční uvedených v [3]. Blokové schéma zapojení je na obrázku 3.2. Kompletní elektrické schéma je na obrázku 3.3.

3.3.1 Připojení ke sběrnici

Obvod CS8900A je připojen ke sběrnici jednotky MO_CPU1 signály A0–12, D0–15, RD, WR, $\overline{\text{IRQ2}}$, $\overline{\text{RES}}$ a jedním z $\overline{\text{CS7}}$–$\overline{\text{CS10}}$. Některé z těchto signálů vstupují do dekodéru, který je popsán v sekci 3.3.2.

Signály A0–A12 adresové sběrnice jsou spojeny přímo s vývody obvodu **SA0**–**SA12**. Signál A0 je navíc přes odpor spojen s vývodem **SBHE**, takže při čtení z resp. zápisu na libovolnou lichou adresu dojde k přepnutí obvodu do 16 bitového režimu. Zkratováním



Obrázek 3.2: Blokové schéma jednotky MO_ETH.

propojky J2 lze tomuto přepnutí zabránit. Obvod pak bude pracovat pouze v 8 bitovém režimu.

Datová sběrnice je zapojena tak, že signály D0–D7 jsou připojeny k **SD8**–**SD15** a signály D8–D15 k **SD0**–**SD7**. Toto prohození vyššího a nižšího bytu při 16 bitovém přenosu je zde proto, že procesory firmy Motorola používají jiné pořadí bytů ve slově, než používá sběrnice ISA. Tento fakt je potřeba mít na paměti při psaní ovladačů.

Signály RD a WR jsou dekódovány, jak je popsáno v následující sekci. Zkratovací propojky J3 umožňují výběr jednoho z signálů $\overline{\text{CS7}}$–$\overline{\text{CS10}}$, který bude použit pro výběr obvodu. Volba tohoto signálu určuje umístění 8 kB oblasti v adresovém prostoru hostitelského procesoru, přes kterou se bude k obvodu přistupovat. Konkrétní adresa začátku této oblasti závisí na nastavení konfiguračních registrů chipselectu v modulu SIM[7] procesoru 68376. Tyto registry také určují typ (8 nebo 16 bitový přenos) a rychlost přístupu k externím obvodům.

Vývod $\overline{\textbf{CHIPSEL}}$ obvodu CS8900A je trvale připojen k zemi a přístup k obvodu je řízen dekodérem popsaným v sekci 3.3.2.

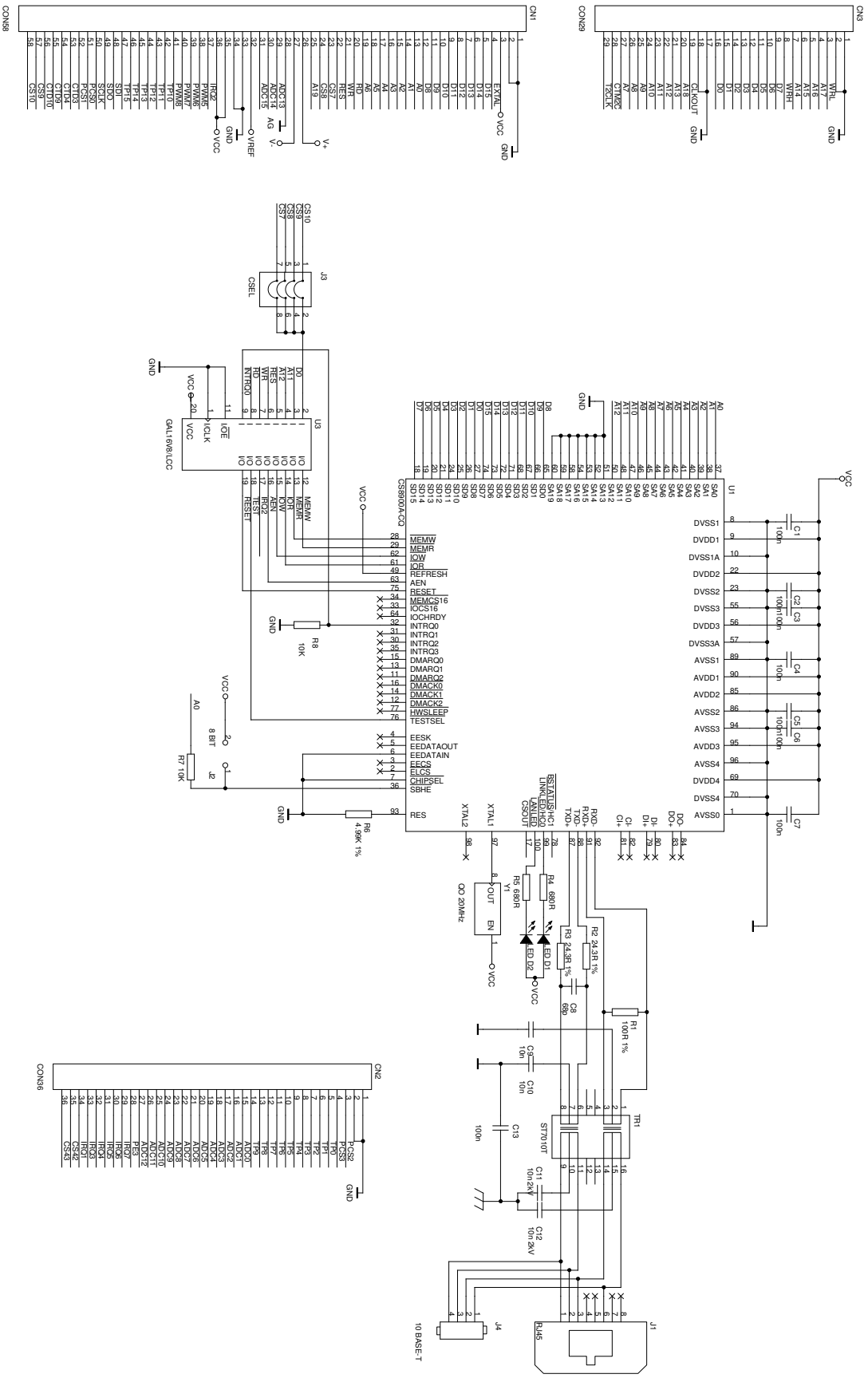
3.3.2 Dekódování signálů

Pro připojení obvodu CS8900A ke sběrnici MO_CPU1 je potřeba dekódovat některé signály z jiných signálů. K dekódování signálů je použit obvod GAL16V8 (U3). Teoreticky by stačilo použít několik negátorů, ale pro první verze rozhraní byl do návrhu radši zahrnut programovatelný dekodér, který umožní jak práci v paměťovém a I/O režimu, tak i provádění Boundary Scan testu (viz 3.1.5).

Do obvodu dekodéru vstupují signály[8] $\overline{\text{CS}x}$, A11, A12, D0, $\overline{\text{RES}}$, WR, $\overline{\text{RD}}$ a **INTRQ0**. Výstup dekodéru tvoří signály $\overline{\textbf{MEMW}}$, $\overline{\textbf{MEMR}}$, $\overline{\textbf{IOW}}$, $\overline{\textbf{IOR}}$, **AEN**, $\overline{\text{IRQ2}}$, $\overline{\textbf{TEST}}$ a **RESET**.

[7] System Integration Module

[8] Tučně jsou znázorněny signály generované nebo čtené obvodem CS8900A, normálním písmem jsou uvedeny signály sběrnice MO_CPU1.



Obrázek 3.3: Schéma zapojení jednotky MO_ETH.

Kombinace hodnot vstupních signálů $\overline{\text{CS}x}$, A11 a A12 určuje pracovní režim, jak je uvedeno v tabulce 3.5. Jednotlivé pracovní režimy jsou popsány v následujících odstavcích.

$\overline{\text{CS}}$	A12	A11	Režim	Oblast paměti
1	X	X	—	—
0	0	0	I/O režim	0x0000 – 0x07FF
0	0	1	test (BST)	0x0800 – 0x0FFF
0	1	X	paměťový režim	0x1000 – 0x1FFF

Tabulka 3.5: Režimy práce obvodu v závislosti na signálech sběrnice.

I/O režim V tomto režimu jsou signály RD a WR připojeny přes negátor na vývody $\overline{\textbf{IOR}}$ a $\overline{\textbf{IOW}}$ a signál **AEN** je trvale v log. 0. Signály $\overline{\textbf{TEST}}$, $\overline{\textbf{MEMW}}$ a $\overline{\textbf{MEMR}}$ jsou trvale v log. 1. Po resetu obvodu je bázová adresa bloku I/O registrů 0x300 a tyto registry jsou tedy přístupné od adresy $\text{CS\_BASE} + 0\text{x}0300$, kde CS_BASE označuje bázovou adresu signálu $\overline{\text{CS}x}$, který je použit k výběru obvodu CS8900A (viz 3.3.1).

Režim Boundary Scan Test V tomto režimu jsou všechny signály pro čtení a zápis neaktivní. Na signál $\overline{\textbf{TEST}}$ je přivedena log. 0 a na vývod **AEN** je kopírována hodnota signálu D0. Ovládání BST se provádí například střídavým zápisem hodnot 0 a 1 do slova na adrese $\text{CS\_BASE} + 0\text{x}0800$.

Paměťový režim V paměťovém režimu jsou signály RD a WR připojeny přes negátor na vývody $\overline{\textbf{MEMR}}$ a $\overline{\textbf{MEMW}}$, signál **AEN** je trvale v log. 0. Signály $\overline{\textbf{TEST}}$, $\overline{\textbf{IOW}}$ a $\overline{\textbf{IOR}}$ jsou trvale v log. 1. Paměťový režim je přístupný od adresy $\text{CS\_BASE} + 0\text{x}1000$.

Paměťový režim	I/O režim	Režim BST
$\overline{\textbf{MEMW}} = \overline{\text{WR}}$	$\overline{\textbf{MEMW}} = 1$	$\overline{\textbf{MEMW}} = 1$
$\overline{\textbf{MEMR}} = \overline{\text{RD}}$	$\overline{\textbf{MEMR}} = 1$	$\overline{\textbf{MEMR}} = 1$
$\overline{\textbf{IOW}} = 1$	$\overline{\textbf{IOW}} = \overline{\text{WR}}$	$\overline{\textbf{IOW}} = 1$
$\overline{\textbf{IOR}} = 1$	$\overline{\textbf{IOR}} = \overline{\text{RD}}$	$\overline{\textbf{IOR}} = 1$
$\textbf{AEN} = 0$	$\overline{\textbf{AEN}} = 0$	$\overline{\textbf{AEN}} = \text{D0}$
$\overline{\textbf{TEST}} = 1$	$\overline{\textbf{TEST}} = 1$	$\overline{\textbf{TEST}} = 0$

Tabulka 3.6: Výstupní hodnoty dekodéru v závislosti na pracovním režimu.

Ostatní signály Signály **INTRQ0** a **RESET** jsou přes dekodér vedeny pouze z důvodu potřeby jejich negace.

Rovnice dekodéru

Z hodnot výstupů dekodéru pro jednotlivé režimy (tabulka 3.6) a hodnot signálů vybírajících daný režim plynou následující rovnice, které popisují činnost dekodéru. Pro signál $\overline{\text{MEMW}}$ dostaneme rovnici

$$\neg\overline{\text{MEMW}} = \neg\overline{\text{CS}} \cdot \text{A12} \cdot \not{\text{WR}}, \tag{3.1}$$

kterou pomocí de Morganových zákonů upravíme do tvaru

$$\overline{\text{MEMW}} = \overline{\text{CS}} + \neg\text{A12} + \text{WR}. \tag{3.2}$$

Obdobně dostaneme rovnice pro další signály

$$\begin{aligned}
\overline{\text{MEMR}} &= \overline{\text{CS}} + \neg\text{A12} + \text{RD}, && (3.3)\\
\overline{\text{IOW}} &= \overline{\text{CS}} + \text{A12} + \text{A11} + \text{WR}, && (3.4)\\
\overline{\text{IOR}} &= \overline{\text{CS}} + \text{A12} + \text{A11} + \text{RD}, && (3.5)\\
\overline{\text{TEST}} &= \overline{\text{CS}} + \text{A12} + \text{A11}, && (3.6)\\
\text{AEN} &= \neg\overline{\text{CS}} \cdot \neg\text{A12} \cdot \text{A11} \cdot \text{D0}, && (3.7)\\
\text{RESET} &= \neg\overline{\text{RES}}, && (3.8)\\
\overline{\text{IRQ2}} &= \neg\text{INTRQ0}. && (3.9)
\end{aligned}$$

3.3.3 Galvanické oddělení

Galvanické oddělení ethernetového rozhraní od síťového vedení je realizováno transformátorem Valor ST7010T (viz [19]). Kondenzátory C11 a C12 by se měly osazovat pouze v případě vysoce zarušené linky. Tyto kondenzátory by měly vydržet nabití na napětí 2 kV. Kondenzátor C13, který spojuje zem všech obvodů s kostrou slouží ke snížení rušení od zdroje a od ostatních obvodů v systému.

3.3.4 Konektory a zkratovací propojky

Celá karta s jednotkou MO_ETH se k jednotce MO_CPU1 připojuje konektory CN1 a CN3. CN1 je základní 8 bitová sběrnice doplněná o další signály, které MO_ETH nevyužívá. CN3 je rozšíření sběrnice na 16 bitů. Konektor CN2 není používán vůbec.

Konektory J1 a J4 jsou výstupy na fyzickou vrstvu 10BASE-T. J1 je standardní konektor RJ45 a J4 slouží jako pomocný konektor, který bude spojen s konektorem RJ45, umístěným například na zadním panelu skříně s celým řídicím systémem.

Zkratovací propojky J2 a J3 jsou popsány v sekci 3.3.1.

3.3.5 Ostatní součástky

Zdrojem hodinového signálu pro CS8900A je 20 MHz krystalový oscilátor Y1. Samotný krystal nebyl použit, protože krystaly pro vyšší frekvence se často rozkmitají na jiném kmitočtu než je jejich vlastní (vyšší harmonické).

Odpor R8 má význam pouze při resetu obvodu. Tehdy je totiž vývod INTRQ0 ve stavu vysoké impedance, a kdyby nebyl odporem připojen k zemi, mohlo by dojít k falešnému generování přerušení. Odpory R1–R3 slouží jako impedanční přizpůsobení linkového vedení. Kondenzátory C1–C7 slouží jako blokovací kondenzátory napájecího napětí obvodu.

Svítivá dioda D1 indikuje přítomnost linkových pulzů (viz 1.3.4) na síti, dioda D2 se rozsvítí při příjmu či odesílání rámce.

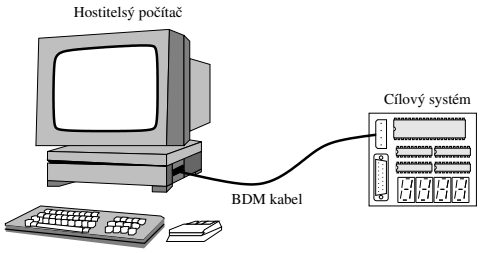
Kapitola 4

Software

Tato kapitola se zabývá vývojem softwaru pro jednotku MO_ETH. V sekci 4.1 jsou popsány některé nástroje použité při vývoji. Dále je popsána jednoduchá aplikace *mo_ether* sloužící převážně k testování hardwarové realizace (sekce 4.2). Popis tvorby podpory ethernetového rozhraní pro systémy RTEMS a uClinux je v sekcích 4.3 a 4.4.

4.1 Vývoj programového vybavení

Pro vývoj programového vybavení jednotky MO_ETH jsou potřeba určité hardwarové a softwarové nástroje. Z hardwaru je potřeba především *hostitelský počítač*, na kterém probíhá vývoj softwaru pro *cílový systém* (v našem případě MO_CPU1). Programy přeložené na hostitelském počítači do binárního tvaru se pak přes BDM rozhraní přenesou na cílový systém, kde jsou spuštěny a nebo můžou být pomocí téhož rozhraní laděny přímo na cílovém systému. Schéma tohoto uspořádání je na obrázku 4.1.



Obrázek 4.1: Uspořádání hardwaru pro vývoj programového vybavení pro MO_CPU1.

Ze softwarových nástrojů je potřeba assembler, linker, překladač jazyka C, debugger, různé pomocné nástroje jako například program `make` a v neposlední řadě i textový editor pro psaní zdrojových kódů. Stručný popis těchto nástrojů je uveden v následujících sekcích. Podrobněji je použití některých nástrojů s jednotkou MO_CPU1 popsáno v [14].

4.1.1 Vývojový řetězec

Nechceme-li psát programové vybavení přímo ve strojovém kódu daného procesoru, potřebujeme nástroje, které do strojového kódu přeloží příkazy vyššího programovacího jazyka. Překlad neprobíhá jednorázově, ale bývá rozdělen do několika fází:

- překlad z vyššího programovacího jazyka do assembleru,
- překlad z assembleru do strojového kódu (objektových souborů) a
- spojování (linkování) objektových souborů do výsledné binární podoby.

Z důvodů jednodušší implementace mívá každou z těchto fází na starosti samostatný program. *Vývojový řetězec* je tedy soubor programů, které provádějí všechny fáze překladu softwaru.

Vývojový řetězec použitý pro vývoj softwaru v této práci je složen z nástrojů GNU[1]. Tyto nástroje jsou šířené pod licencí GPL[2] včetně zdrojových kódů a při splnění celkem volných podmínek je možné programy modifikovat. To je důvod, proč jsou tyto nástroje poměrně rozšířené a podporují valnou většinu moderních procesorů.

Před použitím těchto nástrojů je musíme přeložit ze zdrojových kódů takovým způsobem, aby byly spustitelné na hostitelském počítači a produkovaly kód pro cílový systém. Překladači splňujícímu tuto vlastnost se říká *křížový překladač* (crosscompiler).

Překladače jazyků C, C++ a dalších jsou součástí balíku programů GCC. Pro překlad uClinuxu je potřeba speciálně upravený překladač, který podporuje přepínač `-msep-data`. Při spuštění více kopií jednoho programu přeloženého s touto volbou budou moci všechny kopie sdílet stejný kód, a to i na platformách bez MMU[3]. Programy *assembler* a *linker* jsou součástí balíku programů s názvem *binutils*.

Po úspěšném přeložení těchto nástrojů je možno začít psát vlastní programy pro cílový systém.

4.1.2 Ladící programy

Ladící programy (debuggery), slouží k odhalování chyb v programech. Pro ladění programového vybavení MO_ETH opět využijeme debugger z projektu GNU – program GDB. Tento program lze s úpravami popsanými v [13] přeložit tak, aby bylo možno ladit programy na cílovém systému přes rozhraní BDM (viz 3.2.1). K ladění programů je pak možno použít buď přímo samotný debugger GDB, nebo některou jeho grafickou nadstavbu – např. program DDD[4] nebo dále popsaný editor Emacs.

[1] GNU's Not Unix
[2] General Public License
[3] Memory Management Unit
[4] Data Display Debugger

4.1.3 Editor Emacs

Emacs je na platformě nezávislý textový editor. Jádrem editoru je interpret jazyku Emacs Lisp, což je, jak název napovídá, varianta programovacího jazyku Lisp. Základní funkce editoru a zmíněný interpret jsou napsány v jazyce C, ale zbytek funkčnosti editoru je naprogramován právě v Emacs Lispu. To dává editoru možnost extrémní přizpůsobivosti a konfigurace téměř pro libovolné účely.

Dá se říct, že Emacs velmi usnadní práci při editaci jakéhokoli textového dokumentu, mezi něž samozřejmě patří i zdrojové kódy programů. Protože jsem Emacs používal při práci na programovém vybavení pro jednotku MO_ETH, zmíním se zde o použití Emacsu jako nástroje pro vývoj softwaru.

Základní rysy

Emacs vždy pracuje v jednom z tzv. *hlavních módů*. Každý hlavní mód odpovídá nějakému typu dokumentu, který lze editovat. Existuje tedy mód pro obyčejný text, mód jazyk C, mód pro LaTeX, mód pro XML a mnohé další. V závislosti na aktuálním hlavním módu Emacs *zvýrazňuje syntaxi*, provádí *zarovnávání řádků*, hledá *definice funkcí* atd.

Kromě hlavních módů má Emacs ještě tzv. *vedlejší módy*. Tyto módy rozšiřují funkčnost hlavních módů. Ke každému hlavnímu módu lze zapnout několik vedlejších módů a naopak, jeden vedlejší mód lze aktivovat v různých hlavních módech. Existují vedlejší módy pro automatické doplňování identifikátorů, pro on-line kontrolu pravopisu, pro skrývaní nezajímavých částí kódu atd.

Samozřejmě, že módy nemusíme aktivovat ručně, ale můžeme jejich zapnutí navázat na typ souboru. Například při otevření souboru s příponou `.c` se automaticky aktivuje hlavní mód pro jazyk C a případně další nakonfigurované vedlejší módy.

Emacs lze interaktivně ovládat několika způsoby:

- pomocí klávesových zkratek,
- myší,
- zadáváním příkazů.

Klávesové zkratky Klávesové zkratky se podle zvyklostí zapisují stylem `C-x`, což značí současný stisk kláves `Ctrl` a `x`. Obdobně `M-x` značí současný stisk `Alt` a `x` (u klávesnice PC). Klávesové zkratky editoru mohou začátečníky odradit, protože některé jsou velice dlouhé. Například zkratka `C-x r SPC a` slouží pro zapamatování aktuálního umístění kurzoru v registru A. Po chvíli práce si na ně ale člověk zvykne.

Užitečné příkazy, které definují některé zkratky obvyklé u počítačů PC jsou `pc-selection-mode` a `pc-bindings-mode`.

Ovládání myší Myš funguje v Emacsu jinak, než je dnešní uživatel počítače většinou zvyklý. Je to dáno historickým vývojem Emacsu, který začal již poměrně dávno. Tlačítka

myši se označují podobně jako klávesy. Například `Mouse-1` označuje levé tlačítko. Tímto tlačítkem vždy přemístíme kurzor na dané místo.

V oknech Emacsu je možné mít interaktivní text, který vypadá jako tlačítko nebo se projevuje změnou barvy pozadí při umístění kurzoru myši nad tento text. Interaktivní texty lze aktivovat stiskem prostředního tlačítka myši (`Mouse-2`). Toho lze s výhodou využít při překladu programů, kdy stačí na chybové hlášce kompilátoru stisknout prostřední tlačítko a kurzor se automaticky přemístí na řádek s chybou.

Příkazy I přesto, že klávesových zkratek je velké množství, některé činnosti editoru implicitně žádnou zkratku přiřazenu nemají a musí se vyvolat napsáním příkazu. Příkazy jsou užitečné i v případě, kdy jsme zkratku zapomněli a přesto chceme vyvolat určitou činnost.

Příkazy se v interaktivním režimu zadávají stiskem `M-x` a zapsáním jména příkazu, následovaného stiskem klávesy Enter (`RET`). Aby si uživatel nemusel pamatovat všechny příkazy, po stisku mezery nebo tabulátoru mu Emacs rozepsaný příkaz doplní, nebo pokud je možných doplnění víc, nabídne jaká to jsou. Příklad zadání příkazu je `M-x goto-line RET`.

Tolik velmi stručný úvod do Emacsu. Podrobnější popis editoru je v jeho manuálu, který lze zobrazit například stiskem `C-h i m emacs`.

Použití Emacsu při vývoji softwaru

Emacs nabízí vývojářům mnoho pomocných prostředků. Obsahuje hlavní módy pro většinu programovacích jazyků včetně jazyka C, který byl používán při vývoji softwaru pro jednotku MO_ETH. V těchto módech Emacs zvýrazňuje syntaxi jazyka, zarovnává řádky podle úrovně vnoření bloku kódu, zobrazuje párové závorky atd. Užitečný je vedlejší mód *abbrev*, který umožňuje doplňování rozepsaných identifikátorů po stisku `M-/`.

Další užitečnou pomůckou jsou tzv. *tagy*. Spuštěním programu `etags *.c *.h` se vygeneruje v aktuálním adresáři soubor TAGS, který bude obsahovat informace o umístění funkcí a definic ve všech souborech s příponou .c a .h v aktuálním adresáři. Emacs s tímto souborem umí pracovat a pokud máme kurzor například na nějakém názvu funkce, pak po stisku `M-.` skočí Emacs na začátek definice této funkce. Klávesou `M-*` se můžeme vrátit o neomezený počet úrovní zpět.

Pro překládání programu existuje příkaz `compile`. Po jeho spuštění zadáme příkaz, který se má pro překlad použít. Většinou si vystačíme s nabízeným `make -k`. Po prvním spuštění příkazu `compile` stačí spouštět jen příkaz `recompile`, který už se na nic neptá a jen provede překlad. Je užitečné přiřadit těmto příkazům klávesovou zkratku.

Dalším užitečným příkazem je `speedbar`. Tento příkaz otevře další okno se stromovým seznamem souborů a funkcí definovaných ve zdrojových souborech. Prostředním tlačítkem myši lze seznamem procházet a pohodlně se přemisťovat ve zdrojových kódech.

Ladění programů

Z prostředí editoru Emacs lze jednoduše ladit programy. Po spuštění příkazu `gdb` se nás editor zeptá, jaký debugger chceme použít, a s jakými parametry ho chceme spustit. Poté se otevře nový buffer (okno) s debuggerem a objeví se nové menu. Debugger můžeme ovládat tradičně příkazy z klávesnice, nebo použít menu.

Při trasování programu Emacs sleduje, na jakém řádku se program právě nachází, a podle toho zvýrazňuje daný řádek v bufferu se zdrojovým kódem. V tomto režimu lze jednoduše zadávat pozice breakpointů umístěním kurzoru na požadované místo ve zdrojovém kódu a stiskem `C-x SPC`.

4.2 Testovací aplikace mo_ether

Pro účely testování správnosti hardwarového návrhu a ověření funkce obvodu CS8900A byl vyvinut program *mo_ether*. Tento program je rozšířením již existujícího programu *mo_cpu1* o příkazy pro ovládání ethernetového rozhraní. Program umožňuje přijímat a odesílat holé ethernetové rámce a neimplementuje žádný standardní protokol. Po ověření správné funkčnosti obvodu přestal být program *mo_ether* vyvíjen a proto může obsahovat chyby. Dále je uveden popis programu.

4.2.1 Struktura programu

Hlavní soubor `tst.c` obsahuje funkci `main()`, která je z velké části tvořena nekonečnou smyčkou, zpracovávající příkazy přijaté po sériové lince. Veškerá další činnost je vykonávána jako obsluha příkazů. Popis uživatelských příkazů pro práci s ethernetovým rozhraním obsahuje 4.2.4.

Podpora síťových funkcí je rozdělena na dvě části. V souboru `net.c` je kód nezávislý na použitém hardwaru a v souboru `cs8900.c` je kód pro práci s obvodem CS8900A. Tyto soubory jsou popsány v následujících sekcích.

4.2.2 Soubor net.c

V tomto souboru (a souboru `net.h`) jsou definovány funkce nezávislé na konkrátním síťovém hardwaru. Jedná se zejména o správu front odesílaných a přijatých rámců.

Základem této části je datová struktura `eth_device`, která popisuje ethernetové rozhraní a jejíž zdrojový kód je na obrázku 4.2.

Význam jednotlivých položek struktury je následující:

`address` obsahuje MAC adresu síťového rozhraní,

`rx_head` ukazatel na začátek fronty přijatých rámců (nejdříve přijatý rámec),

`rx_tail` ukazatel na konec fronty přijatých rámců,

```
typedef struct eth_device {
  ETH_ADDRESS address;

  ETH_PACKET *rx_head, *rx_tail;
  ETH_PACKET *tx_head, *tx_tail;
  ETH_PACKET *free_packets;

  void (*do_transmit)();
  WORD (*line_status)();

  int rx_count;
  int tx_count;
  int tx_errors;
  int collision_count;
  int miss_count;

  /* debug */
  int int_count;
} ETH_DEVICE;
```

Obrázek 4.2: Struktura eth_device.

tx_head, tx_tail dtto pro frontu odesílaných rámců,

do_transmit ukazatel na funkci, která provede fyzické odeslání rámce,

line_status ukazatel na funkci vracející stav linky,

***_count** čítače událostí pro statistické účely.

Inicializaci ethernetového zařízení provádí funkce `eth_init()`.

```
void eth_init(ETH_DEVICE *device, ETH_ADDRESS addr, int no_packets)
```

Parametr `device` je ukazatel na nevyplněnou strukturu `ETH_DEVICE`, `addr` je MAC adresa, která bude rozhraní přiřazena, a `no_packets` je počet rámců, pro které se má alokovat paměť (maximální délka fronty přijatých či odesílaných rámců).

Odeslání rámců má na starosti funkce `eth_send()`.

```
void eth_send(ETH_DEVICE *device, ETH_ADDRESS dest, char *data,
              WORD length)
```

Parametr `dest` udává cílovou adresu rámce, parametr `data` je ukazatel na data, která chceme poslat a `length` je délka těchto dat. Funkce zkopíruje data do vyrovnávací paměti alokované v `eth_init()`, doplní cílovou a zdrojovou adresu a zařadí připraveny rámec do fronty. Je-li tento rámec jediný ve frontě, znamená to, že momentálně není odesílán jiný rámec a je zavolána funkce `device->do_transmit()`, která provádí vlastní odesílání.

Funkce `eth_receive()` slouží k získání přijatých rámců.

```
BOOL eth_receive(ETH_DEVICE *device, ETH_PACKET *packet)
```

Funkce vrací logickou hodnotu, udávající úspěšnost operace. Pokud je k dispozici přijatý rámec, je zkopírován do struktury, na kterou ukazuje ukazatel `packet` a funkce vrátí hodnotu `TRUE`. V opačném případě je vráceno `FALSE`.

V souboru jsou dále funkce pro práci s frontami jako například `fifo_get()` a `fifo_add()`.

4.2.3 Soubor cs8900.c

Zde jsou definovány funkce pro přímou komunikaci s obvodem CS8900A, spolupracující s nezávislým rozhraním ze souboru `net.c`. Popis důležitých funkcí následuje dále.

```
void cs8900_init()
```

volá `eth_init()`, inicializuje obvod CS8900A a registruje obslužnou rutinu přerušení od obvodu CS8900A. Toto je jediná funkce, která musí být volána pro inicializaci softwaru i hardwaru síťového rozhraní.

```
static void pp_write (WORD reg, WORD data)
static WORD pp_read(WORD reg)
```

jsou funkce pro zápis/čtení do/z PocketPage paměti. V těchto funkcích je implementována výměna vyššího a nižšího bytu, která je nutná díky překřížené datové sběrnici (viz 3.3.1).

```
static int hardware_write_packet(ETH_PACKET *packet)
```

slouží k odeslání připraveného rámce.

```
static void hardware_read_packet(ETH_PACKET *packet)
```

přenese data z vyrovnávací paměti CS8900A do hlavní paměti na místo, kam ukazuje parametr `packet`.

```
void cs8900_led(BOOL on)
```

Diagnostická funkce, která umožňuje softwarové ovládání LED diody na desce MO_ETH. Parametr `on` určuje, má-li se dioda rozsvítit nebo zhasnout.

```
void cs8900_isr(int intno, void *dev_id, struct pt_regs *regs)
```

Obslužná rutina přerušení. Funkce čte v cyklu hodnotu registru ISQ[5] a dokud je signalizován platný zdroj přerušení, podle přečtené hodnoty volá funkce pro příjem nebo odeslání rámce. Rovněž jsou v této funkci aktualizovány různé statistické údaje.

[5] Interrupt Status Queue

4.2.4 Uživatelské příkazy

Tato sekce popisuje uživatelské příkazy, které je možné použít pro ovládání jednotky MO_ETH. Příkazy se posílají po sériové lince jako ASCII text.

`cssend <data>` odešle rámec obsahující data z prvního parametru na všesměrovou (broadcast) adresu. Úspěšné odeslání je oznámeno výpisem hlášky na konzolu.

`csrec` Pokud je ve frontě přijatý rámec, vypíše jej v přehledné formě na konzolu. V opačném případě je oznámena chyba.

`csstat` vypíše statistiky ethernetového rozhraní.

`cscont <num>` v závislosti na hodnotě parametru zapne nebo vypne nepřetržité odesílání rámců.

`csled <num>` v závislosti na hodnotě parametru je rozsvícena či zhasnuta LED dioda.

4.2.5 Závěr

Program *mo_ether* plní dobře svou funkci při testování funkčnosti hardwaru. Díky jeho jednoduchosti ho lze snadno upravit pro testování jiných vlastností ethernetového rozhraní. Pro praktické nasazení ve skutečných aplikacích by bylo potřeba implementovat nějaký komunikační protokol a odstranit omezení, která program obsahuje.

4.3 RTEMS

RTEMS[6] je exekutiva pro aplikace reálného času původně vyvinutá pro americkou armádu. V současné době je to projekt s otevřeným zdrojovým kódem distribuovaným z velké části pod licencí GPL. RTEMS má implementováno rozhraní podle standardu POSIX 1003.1b, obsahuje TCP/IP stack převzatý z OS FreeBSD a je dostupné pro mnoho různých procesorů.

Veškerá práce s RTEMS popisovaná dále probíhala na verzi (tzv. snapshotu) z 15. 2. 2002. V současných verzích mohou být některé věci jinak. Informace byly čerpány z [12].

4.3.1 Podpora sítí

Součástí síťového kódu RTEMS je TCP/IP stack převzatý z OS FreeBSD. S TCP/IP stackem komunikují na jedné straně uživatelské programy pomocí standardního POSIXového rozhraní a na straně druhé ovladače různých síťových zařízení. Pro komunikaci ovladačů se zbytkem síťového kódu nabízí RTEMS rozhraní, které bude popsáno dále.

Každý ovladač síťového zařízení obsahuje dvě úlohy (tasks). Jedna slouží k odesílání paketů a druhá k jejich příjmu. Tyto úlohy pracují ve smyčce a pokud není přijímán ani

[6] Real-Time Executive for Multiprocessor Systems

odesílán žádný paket, jsou blokovány čekáním na událost systému RTEMS. Událost je zaslána úlohám v okamžiku kdy je síťovým hardwarem oznámeno přijetí paketu nebo když uživatelský program odesílá paket. Úloha pro odesílání vybírá pakety z fronty paketů k odeslání a předává je hardwaru, přijímací úloha plní opačnou funkci.

Kromě těchto dvou úloh obsahují ovladače síťových zařízení další funkce:

driver_attach slouží pro počáteční konfiguraci ovladače a jeho propojení se zbytkem síťového kódu.

driver_start je volána když je potřeba odeslat nějaký paket. Její činnost většinou spočívá v poslání události odesílací úloze.

driver_init inicializuje hardware, zaregistruje obsluhu přerušení a spustí odesílací a přijímací úlohu.

driver_isr obslužná rutina přerušení, která většinou jen posílá RTEMS události přijímací a odesílací úloze.

driver_ioctl obsluhuje IOCTL požadavky.

driver_stat slouží k tisku statistických údajů o síťovém rozhraní.

Názvy jmenovaných funkcí mohou být libovolné, protože síťovému kódu RTEMS jsou funkcí `driver_attach` předávány pouze ukazatele na tyto funkce.

4.3.2 Ovladač CS8900A

Ovladač obvodu CS8900A je v RTEMS rozdělen do dvou částí. Jedna část je nezávislá na použité platformě a obsahuje převážně funkce popsané v 4.3.1. Tato část volá pro platformě závislé operace funkce druhé části ovladače, která je součástí BSP[7].

V použité verzi RTEMS byla obsažena jen část, která je součástí BSP. Druhá část byla převzata z projektu *My Right Boot* [5] z 10. 6. 2001. Oba dva soubory implementující jednotlivé části ovladače byly uloženy do adresáře BSP (`lib/libbsp/m68k/mo376/network` a pojmenovány `cs8900.c` a `cs8900bsp.c`.

Protože ovladač umožňoval přístup k obvodu pouze v paměťovém režimu, který na první verzi MO_ETH nefungoval, musely být provedeny úpravy, které umožnily přístup v I/O režimu. Volba režimu, který je ovladačem používán, závisí na symbolu preprocesoru `CS8900_MEMORY_MODE`. Je-li tento symbol definován, přistupuje se k obvodu v paměťovém režimu, není-li definován, přistupuje se v I/O režimu.

Další úprava ovladače se týkala specifikace MAC adresy. Původní ovladač počítal s připojením EEPROM paměti k CS8900A, ze které byla do obvodu načtena konfigurace včetně MAC adresy. MO_ETH žádnou EEPROM paměť nemá a tak musí být MAC adresa obvodu sdělena při softwarové konfiguraci. MAC adresy pro jednotlivá rozhraní se definují stejně jako bázové adresy či vektory přerušení – jako symboly preprocesoru, které

[7] Board Support Package

jsou použity pro inicializaci statických polí v `cs8900bsp.c`. Definice symbolů jsou umístěny v souboru `lib/libbsp/m68k/mo376/include/bsp.h`. Symbol s definicí adresy nese název `ETHERNET_MAC_ADDRESS`.

4.3.3 Testovací aplikace

Pro odzkoušení funkčnosti ovladače byla vytvořena jednoduchá testovací aplikace s názvem *eth*. Funkce Init, která je vstupním bodem programů RTEMS je na obrázku 4.3.

```
rtems_task Init(rtems_task_argument ignored)
{
  printf( "\n\nEthernet test program\n" );

  rtems_bsdnet_initialize_network();
  printf( "\n\nNetwork initialized\n" );
  rtems_bsdnet_show_if_stats();

  while (1) {
    rtems_task_wake_after(100);
    rtems_bsdnet_show_if_stats();
  }
}
```

Obrázek 4.3: Funkce Init testovací aplikace.

Funkce je velmi jednoduchá a nedělá nic jiného, než že inicializuje síťové knihovny a pak periodicky vypisuje statistiku síťového rozhraní.

Součástí aplikace je ještě její konfigurace v souboru `system.h`, což je povinná část každé RTEMS aplikace. Konfigurace definuje, jaké části RTEMS mají být s aplikací slinkovány, a obsahuje nastavení hodnot různých parametrů jako například maximální počet úloh v aplikaci. V souboru `netconfig.h` je konfigurace síťových knihoven. Podle toho, zda je definován symbol preprocesoru `USE_DHCP` je vybrána konfigurace s použitím protokolu DHCP nebo statická, v níž jsou všechny hodnoty zadány předem v konfiguračním souboru.

Aplikace potvrdila funkčnost ovladače. Po připojení jednotky k síti bylo ethernetové rozhraní nakonfigurováno DHCP serverem a příkazem `ping` byla ověřena funkčnost IP protokolu. Při každém zaslání IP datagramu se patřičně zvětšily hodnoty čítačů zobrazovaných v pravidelně vypisované statistice ethernetového rozhraní.

4.4 uClinux

Projekt uClinux[8] [18] je distribuce OS Linux určená speciálně pro vložená zařízení. Součástí distribuce je upravené jádro Linuxu a poměrně velké množství uživatelských programů. Distribuce je šířena ve zdrojových kódech a pro použití na dané platformě se musí potřebné části přeložit na míru danému systému.

V následujících sekcích bude popsáno, z čeho se distribuce skládá (4.4.1), jak se provádí konfigurace pro určitý cílový systém (4.4.2), jak muselo být upraveno jádro Linuxu (4.4.3). Dále je uvedeno, jaký se používá souborový systém 4.4.4, jak bylo rozvrženo používání paměti 4.4.5 a rovněž jsou popsány startovací skripty 4.4.6). Nakonec jsou popsány možnosti využití ethernetového rozhraní spolu s uClinuxem.

4.4.1 Popis distribuce

Jádro

Součástí uClinuxu jsou jádra z řad 2.0–2.5. Jádro řady 2.5[9] se od tzv. *vanilla* jádra, které vydává Linus Torvalds, původní autor Linuxu, téměř neliší. Starší jádra jsou doplněna o podporu dalších architektur – například *m68knommu*. Tato architektura je určena pro platformy s procesory Motorola 68xxx, jež nemají jednotku správy paměti (MMU). Mezi další úpravy jádra patří i zahrnutí vlastní varianty ovladače obvodu CS8900A.

Knihovny

uClinux obsahuje několik knihoven pro jazyk C (libc). Tyto knihovny jsou využívány všemi programy napsanými v jazyce C. Jednotlivé verze zahrnuté v distribuci se od sebe liší paměťovou náročností a s ní související kvalitou a množstvím nabízených funkcí.

glibc je knihovna z projektu GNU, používaná v běžných distribucích Linuxu, která nabízí bezpochyby největší množství funkcí. Díky tomu má velké paměťové nároky a její rychlost na pomalých procesorech není příliš uspokojivá. Knihovna se používá pro architektury obsahující MMU.

uC-libc je odlehčená verze knihovny jazyka C, standardně používaná uClinuxem, a je použita i v této práci.

uClibc je další odlehčená knihovna jazyka C, jejímž autorem je Erik Andersen. Zdá se, že tato knihovna je nepatrně větší než uC-libc.

[8] uC je zkratka spojení μ-Controller (micro-Controller) a vyslovuje se jako anglické you see.

[9] v současné době vývojová verze jádra Linuxu

Uživatelské programy

V distribuci uClinux je velké množství uživatelských programů, které lze volitelně zahrnout výsledného systému. Pokud je to nutné, jsou tyto programy přizpůsobené pro použití s uClinuxem a s odlehčenými knihovnami jazyka C.

Pro naše účely je velmi užitečný program `busybox`, který v jednom binárním souboru implementuje značné množství standardních unixových příkazů. Volání busyboxu místo samostatných programů je realizovánu prostřednictvím pevných odkazů na soubor `/bin/busybox`.

Dalšími programy, které se hodí pro použití s MO_ETH, jsou různé síťové nástroje, jako například démoni `telnetd` a `dhcpcd`, příkazy pro konfiguraci síťových rozhraní, program `portmap`, umožňující vzdálené volání procedur apod.

4.4.2 Kofigurace pro MO_CPU1

S uClinuxem jsou distribuovány i konfigurační soubory pro různé systémy. Jednotka MO_CPU1 mezi nimi samozřejmě není, a proto bylo nutné tuto konfiguraci vytvořit. Konfigurační soubory se nacházejí v adresáři pojmenovaném podle schématu `vendors/<společnost>/<zařízení>`.

Konfigurace pro MO_CPU1 se tedy nachází v adresáři `vendors/PiKRON/MO_376`[10]. Při tvorbě konfigurace byla jako základ vzata konfigurace desky uCsimm (viz 2.2.4), neboť tato deska obsahuje podobný procesor a stejný ethernetový řadič. Součástí konfigurace pro určitou desku jsou konfigurační soubory pro překlad jádra Linuxu a konfiguraci uživatelských programů. Dále jsou zde startovací skripty systému a soubor `Makefile`, který obsahuje předpis pro tvorbu binárních obrazů pro daný systém.

Konfiguraci lze měnit buď ručně editací konfiguračních souborů a nebo pohodlněji konfiguračními nástroji, spouštěnými z hlavního adresáře distribuce příkazy `make menuconfig` nebo `make xconfig`.

Startovací skripty pro MO_ETH jsou popsány v sekci 4.4.6 a soubor `Makefile` v 4.4.4 a 4.4.5.

4.4.3 Příprava jádra Linuxu

S ohledem na velikost paměti jednotky MO_CPU1 a na kvalitu podpory platformy *m68knommu* bylo zvoleno použití jádra řady 2.0. Aktuální verze, distribuovaná s uClinuxem, nese označení `linux-2.0.39-uc2`.

Podpora pro MO_376 se nachází v adresáři `arch/m68knommu/platform/68376/MO376`. Zde jsou k dispozici dvě konfigurace – jedna pro umístění Linuxu v paměti ROM/FLASH a druhá pro paměť RAM. Pro každou konfiguraci je zde soubor `*.ld`, což je předpis pro sestavovací program (linker), jak sestavit výsledný binární obraz (viz 4.4.5), a soubor `crt0-*.S`, který obsahuje startovací kód spouštěný po startu

[10] MO_376 je zde označení pro desku MO_CPU1 a přídavné periferie jako například MO_ETH.

systému. Jeho úkolem je připravit hardware pro spuštění jádra Linuxu a následně toto jádro spustit.

Kromě přidání tohoto adresáře musely být upraveny některé další soubory, jejichž výčet spolu s popisem úpravy následuje zde:

`arch/m68knommu/platform/68376/Rules.make` Byla změněna hodnota proměnné CROSS_COMPILE tak, aby byl překlad jádra prováděn správným překladačem (prefix m68k-uclinux-elf-).

`arch/m68knommu/Boards.mk` Při konfiguraci jádra pro desku MO_376 se nastaví proměnná BOARD na hodnotu MO376.

`arch/m68knommu/config.in` Zde jsou přidány volby, které se objeví v konfiguračním menu Linuxu (např. `make menuconfig`). Jedná se o volby pro výběr taktovací frekvence procesoru 21 MHz, výběr desky MO_376 a zadání MAC adresy pro MO_ETH.

`drivers/char/68332serial.c` Zde byla rozšířena podmínka podmíněného překladu o symbol MO_376.

`drivers/net/Makefile` Síťový kód jádra nelze přeložit bez funkcí v souboru `autoirq.o` i když tento kód není ovladačem CS8900A využíván. Proto je potřeba zahrnout tento soubor do sestavovacího procesu jádra. Systémovější řešení by bylo provedení podrobnější analýzy kódu a odstranění nepoužívaných odkazů na tyto funkce.

`Makefile` Zde byly upraveny proměnné, které ovlivňují překlad. Nejdůležitější jsou proměnné ARCH, PLATFORM a CFLAGS.

`drivers/block/blkmem.c` Do tohoto souboru byla přidána definice udávající místo v paměti, kde začíná blokové zařízení `/dev/rom0`, a na kterém je začátek kořenového systému souborů.. Toto místo je určeno hodnotou symbolu `__root_fs_start`, který je definován v linkovacím skriptu (viz 4.4.5).

Úprava ovladače CS8900

Protože uClinuxem je velmi dobře podporovaná deska uCsimm, která obsahuje obvod CS8900A, nebylo pro zprovoznění ethernetového rozhraní potřeba do kódu ovladače výrazně zasahovat. Kód ovladače se nachází v souboru `drivers/net/uCcs8900.c`. Jedna potřebná úprava se týká registrace obslužné rutiny přerušení, která je řešena následujícím kódem.

```
#ifdef CONFIG_MO376
        if (request_irq(IRQ_MACHSPEC | 26,
                        cs8900_interrupt,
                        IRQ_FLG_STD,
                        "CrystalLAN_cs8900a", NULL))
```

```
                panic("Unable to attach cs8900 intr\n");

        PFPAR |= 1 << 2;
#endif
```

MO_ETH používá pro obsluhu přerušení tzv. auto-vektor. Periferie používající auto-vektor nemusí při žádosti o přerušení sdělovat procesoru, který přerušovací vektor má pro obsluhu použít, ale vektor je zvolen automaticky v závislosti na prioritě použitého signálu žádosti o přerušení ($\overline{\text{IRQ7}}$–$\overline{\text{IRQ1}}$). Pro tyto přerušení jsou rezervovány vektory 25–31.

Další úprava se týká způsobu, jakým je ovladači předána MAC adresa stanice. Tato adresa se specifikuje při konfiguraci jádra a je do zdrojových kódů je předávána jako symbol preprocesoru CS8900_MAC. Konfigurace MAC adresy je prováděna následujícím kódem.

```
#ifdef CONFIG_MO376
        {
                u64 mac = CS8900_MAC;
                memcpy(dev->dev_addr, (char *)(&mac)+2, 6);
        }
#endif
```

Startovací kód

Startovací kód provádí následující činnosti:

- Inicializuje sériové rozhraní (modul QSM) procesoru MC68376, aby bylo použitelné jako systémová konzola.
- Dále obsahuje inicializaci modulu SIM, kde jsou konfigurovány parametry jednotlivých signálů *chipselect*.
- Při konfiguraci do paměti ROM se musí provést zkopírování sekce inicializovaných dat do paměti RAM.
- Je vynulována oblast paměti pro neinicializované proměnné.
- Nakonec je na konci paměti inicializován systémový zásobník a je předáno řízení Linuxu.

4.4.4 Souborový systém

Po překladu uživatelských programů do binární podoby je potřeba vytvořit souborový systém, který bude obsahovat všechny potřebné soubory a následně bude přenesen na cílové zařízení. S výhodou se zde používá souborový systém *romfs*, který přidává k souborům minimální množství doplňujících informací a obraz souborového systému je tedy

velmi malý. Rovněž kód, který s tímto souborovým systémem pracuje, je velmi jednoduchý a zabírá málo místa v paměti.

Sestavení souborového systému je prováděno podle příkazů uvedených v souboru `vendors/PiKRON/MO_376/Makefile` pod cílem `romfs`. Sestavení provádí program `genromfs` a výsledkem jeho činnosti je jeden soubor obsahující obraz souborového systému.

4.4.5 Paměťový model

V poslední fázi překladu jádra se spojují jednotlivé objektové soubory do jednoho velkého souboru, který obsahuje obraz celého jádra. Binární data jsou v objektových souborech uložena v různých sekcích podle toho, zda se jedná o kód programu, o inicializovaná data nebo třeba o informace pro ladící program. Linker, který tuto činnost provádí, musí mít informace o tom, jak má se kterými sekcemi naložit. Tuto informaci poskytují linkeru konfigurační soubory, kterým se říká *linkovací skripty*.

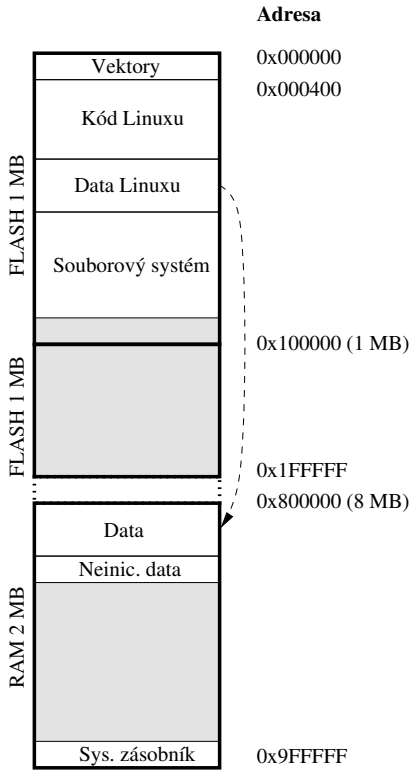
Při tvorbě linkovacího skriptu si musíme rozvrhnout jaké rozdělení paměti bude použito na cílovém systému. Částečně je to dáno umístěním jednotlivých typů paměti (RAM, ROM) v adresovém prostoru a částečně je volba na nás. Paměťový model, který jsem zvolil pro umístění systému do paměti FLASH je na obrázku 4.4. Jednotlivé oblasti jsou popsány dále.

Vektory je 1024 bytů dlouhá oblast paměti, kde je uloženo 256 vektorů obslužných rutin přerušení. Tato tabulka může být v paměti určené pouze pro čtení, protože Linux má své vlastní datové struktury, podle kterých jsou volány funkce pro obsluhu jednotlivých přerušení.

Speciální význam mají první dva vektory. Po resetu procesoru jsou tyto vektory přečteny a hodnota prvního z nich je uložena do registru ukazatele vrcholu systémového zásobníku (A7). Druhá hodnota je načtena do programového čítače a procesor pak začne vykonávat kód od této adresy. V našem případě je vrchol zásobníku umístěn na konec paměti RAM a startovací adresa na začátek kódu v souboru `crt0-rom.S`. Hodnoty ostatních vektorů jsou většinou ukazatele na funkci `trap()` pro ošetření hardwarových výjimek či na funkci `inthandler()` pro obsluhu přerušení. Vektor 32 ukazuje na funkci `system_call` a přes tento vektor provádějí uživatelské programy systémová volání.

Kód Linuxu je oblast kde jsou uloženy sekce `.text` objektových souborů. Tato část obsahuje veškerý spustitelný kód jádra Linuxu.

Data Linuxu je oblast obsahující všechny inicializované proměnné využívané Linuxem. Protože proměnné je potřeba za běhu jádra měnit, musí být před vlastním startem jádra tato sekce zkopírována do paměti RAM. Kopírování opět zajišťuje kód



Obrázek 4.4: Paměťový model pro systém v ROM paměti. Šedě je označena volná paměť.

v souboru `crt0-rom.S`. Linker musí být nakonfigurován tak, aby veškeré odkazy na tato data směřovaly do oblasti RAM, ale ve výsledném obrazu jádra byla data uložena do paměti ROM.

Souborový systém obsahuje obraz kořenového souborového systému. V našem případě se jedná o souborový systém *romfs*, který je určen právě pro účely uchovávání souborů v paměti ROM. O souborovém systému pojednává sekce 4.4.4.

Neinicializovaná data je oblast paměti, kde jsou uloženy všechny statické proměnné Linuxu, které nemají stanovenu počáteční hodnotu. Tato oblast musí být před startem systému vynulována. Nulování provádí kód v `crt0-rom.S`.

Systémový zásobník je 32 kB dlouhá oblast na konci paměti, která slouží jako zásobník jádra. Uživatelské programy mají svůj zásobník jinde v paměti.

Generování binárního obrazu celého systému

Poté, co je přeloženo jádro Linuxu a vygenerován obraz souborového systému, jsou tyto části kvůli snadnější manipulaci spojeny do jednoho souboru. Tuto operaci provádějí příkazy pod cílem `image` v souboru `vendors/PiKRON/MO_376/Makefile`. Nejprve je obraz souborového systému zabalen programem `objcopy` do objektového souboru, který je následně linkerem spojen s jádrem systému a výsledek je uložen do souboru `image/image.coff`.

Výpis sekcí správně sestaveného obrazu uClinuxu uvádí obrázek 4.5. U jednotlivých sekcí je uvedena jejich velikost, adresa umístění v paměti (LMA), adresa v paměti při běhu programu (VMA), příznaky atd.

Pro nahrání výsledného obrazu do paměti FLASH je v souboru `Makefile` cíl `vendor_flash`. K nahrání obrazu do paměti FLASH je použit program `bdmload`. Vygenerování binárního obrazu a nahrání do paměti FLASH se provede z hlavního adresáře distribuce příkazem `make all vendor_flash`.

4.4.6 Startovací skripty

Start každého UNIXového systému je řízen tzv. *startovacími skripty*. Po úspěšném spuštění a inicializaci jádra systému je spuštěn proces `/sbin/init`. Ten vykonává činnost, která je specifikována v souboru `/etc/inittab`. V případě uClinuxu je automaticky proveden i soubor `/etc/rc`, který obsahuje příkazy shellu[11]. Obsah `rc` skriptu je na obrázku 4.6

Nejprve se nastaví síťové jméno počítače, pak je na ramdisk rozbalen obraz prázdného souborového systému, který se rozbalí na ramdisk. Následně je tento souborový systém připojen k adresáři `/var` a jsou v něm vytvořeny potřebné podadresáře. Pak

[11] interpret příkazů

```
Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .romvec       00000400  00000000  00000000  00002000  2**2
                  CONTENTS, ALLOC, LOAD, CODE
  1 .flash        00000000  00000000  00000000  00000000  2**2
                  ALLOC, LOAD, CODE
  2 .text         000524b8  00000400  00000400  00002400  2**2
                  CONTENTS, ALLOC, LOAD, CODE
  3 .eflash       00000000  00100000  00100000  00000000  2**2
                  ALLOC, LOAD, CODE
  4 .data         00012290  00800000  000528b8  00056000  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  5 .rootfs       00089800  00064b48  00064b48  00068b48  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  6 .bss          00017c70  00812290  00812290  00000000  2**2
                  ALLOC
  7 .eram         00000000  00a00000  00a00000  00000000  2**2
                  ALLOC, LOAD, CODE
  8 .stab         00151fc8  00000000  00000000  000f2348  2**2
                  CONTENTS, DEBUGGING
  9 .comment      000057a8  001d7910  001d7910  00244310  2**2
                  CONTENTS, DEBUGGING
 10 .stabstr      00085945  00151fc8  00151fc8  00249ab8  2**0
                  CONTENTS, DEBUGGING
```

Obrázek 4.5: Výpis sekcí objektového souboru s binárním obrazem kompletního systému.

```
hostname mo376
/bin/expand /etc/ramfs.img /dev/ram0
mount -t proc proc /proc
mount -t ext2 /dev/ram0 /var
mkdir /var/tmp
mkdir /var/log
mkdir /var/run
mkdir /var/lock
ifconfig lo 127.0.0.1
route add -net 127.0.0.0 netmask 255.0.0.0 lo
ifconfig eth0 up
/bin/dhcpcd eth0 &
portmap &
sleep 1
ifconfig eth0
cat /etc/motd
```

Obrázek 4.6: Startovací skript /etc/rc.

se přistoupí ke konfiguraci síťových rozhraní. Nejprve se inicializuje lokální smyčka (rozhraní `lo`) a je přidána patřičná položka do směrovací tabulky. Pak se inicializuje zařízení `eth0`, které je v našem případě reprezentováno deskou MO_ETH.

Dále je spuštěn na pozadí (znak &) *klientský démon DHCP*, který se pokusí automaticky nakonfigurovat rozhraní `eth0` na základě informací od DHCP serveru, který musí být umístěn na připojené síti. Pak je spuštěn program *portmap*, který poskytuje ostatním počítačům v síti informace o tom, na kterém portu poslouchají různé programy poskytující služby RPC[12]. Tento program je nutný pro použití protokolu NFS[13] k přístupu na síťové disky. Nakonec je po sekundové pauze (aby DHCP server stačil odpovědět) vypsáno nastavení rozhraní eth0 a zobrazena uvítací zpráva.

Spuštění systému je dokončeno provedením příkazů předepsaných v souboru `/etc/inittab`, jehož obsah je na obrázku 4.7. Nejprve je spuštěn superserver `inetd`, který je v našem případě nakonfigurován[14] tak, aby při příchozím spojení protokolu telnet byl spušten telnet server `/bin/telnetd` a systém se tak dal ovládat z jiného počítače. Nakonec je spuštěn program agetty, který vypíše přihlašovací výzvu a po zadání přihlašovacího jména spustí program login, která se postará o zbytek přihlašovacího procesu. Platní uživatele jsou definováni v souboru `/etc/config/passwd`, který obsahuje pouze uživatele *root* s heslem *uClinux*.

```
inet:unknown:/bin/inetd
ttyS0:vt100:/bin/agetty -l /bin/login ttyS0 9600
```

Obrázek 4.7: Soubor /etc/inittab

4.4.7 Testování a aplikace

Postupem popsaným v předešlých sekcích jsme získali plnohodnotný linuxový systém s podporou sítí. S tímto systémem lze pracovat stejně jako se systémem na osobním počítači. Jsme limitováni pouze velikostí paměti a rychlostí procesoru. Přestože je výkon jednotky MO_CPU1 mnohem nižší, než výkon současných osobních počítačů, lze využívat mnoho nástrojů známých z osobních počítačů či unixových serverů. V následujících odstavcích jsou popsány některé možnosti využití ethernetového rozhraní jednotky MO_376 s uClinuxem.

Vzdálený přístup Standardní vlastností unixových operačních systémů je možnost pracovat na systému vzdáleně. Uživatel se fyzicky nachází u jednoho počítače, ale ve skutečnosti pracuje s jiným počítačem, ke kterému přistupuje přes síť.

K systému MO_376 lze vzdáleně přistupovat protokolem *telnet* pomocí stejnojmenného programu. Pokud se bude řídicí jednotka nacházet na nepřístupném místě

[12] Remote Procedure Call
[13] Network File System
[14] konfigurace je v souboru `/etc/inetd.conf`

(například bude součástí nějakého obráběcího stroje ve velké tovární hale), nemusí obsluha vážit dlouhou cestu k jednotce, ale pohodlně ji nakonfiguruje z operátorského stanoviště.

Nevýhodou programu `telnet` je, že ho lze použít pouze v interaktivním režimu. Pokud by si uživatel připravil na jiném počítači sekvence příkazů, které by chtěl na MO_376 automaticky spouštět, program telnet mu neposlouží. Tento problém řeší program `rsh`, ale ten není součástí distribuce uClinux. Místo něj se v distribuci nachází program `ssh` (klient i server), který veškerou komunikaci šifruje a může nahradit jak telnet, tak rsh. V aktuální verzi uClinuxu, ale chyběly některé soubory Makefile a program nešel přeložit. Mé pokusy o vlastní nakonfigurování programu nebyly úspěšné. Zbývá tedy buď počkat na další verzi uClinuxu nebo problém řešit jinými nástroji.

Síťové disky Kořenový souborový systém uložený v paměti FLASH má omezenou velikost danou velikostí této paměti a v současné době do něj nelze zapisovat. To je celkem podstatné omezení. Jedno z řešení je připojení síťových disků, které jsou umístěny na jiném počítači.

Přístup k síťovým diskům lze realizovat protokolem NFS. Připojení síťového disku se provede standardně příkazem `mount`. Následující příklad připojí adresář /tmp z počítače s IP adresou 192.168.2.1 do adresáře /mnt.

```
# mount 192.168.2.1:/tmp /mnt
```

V praxi lze síťové disky využít například k umisťování datových souborů pro řízení obráběcího stroje připojeného k MO_376. Ze síťových disků lze také spouštět programy, či natahovat dynamicky zaváděné moduly do jádra. Tím se výrazně rozšíří konfigurovatelnost systému.

Starší verze kódu pro obsluhu NFS obsahovala chybu, při které nedocházelo k uvolňování paměti. Proto byly provedeny zátěžové testy, které výskyt chyby v této verzi jádra nepotvrdily.

Další možností jak sdílet data po síti je použití protokolu SMB, který je používán v OS Windows. Použité jádro Linuxu tento protokol podporuje, ale pro připojení je potřeba program `smbmount`. Překlad tohoto programu se v aktuální verzi uClinuxu nezdařil, neboť program používal symboly, které nebyly nikde definovány.

Vzdálené ladění programů Přes síťové rozhraní lze provádět ladění uživatelských programů na dálku. Jednotka MO_376 nemusí být připojena k hostitelskému počítači rozhraním BDM. Pro ladění jádra Linuxu je BDM stále jedinou možností.

Ke vzdálenému ladění slouží na straně cílového systému program `gdbserver`. Použití programu vypadá následovně:

```
# gdbserver <pocitac>:<port> <program> <parametry>
```

Počítač je jméno počítače či adresa, ze které se budeme připojovat debuggerem, *port* je číslo portu, na kterém bude spojení očekáváno a *program* je název laděného programu,

kterému lze předat *parametry*. Laděný program na cílovém systému nemusí obsahovat ladicí informace.

Na hostitelském počítači je potřeba spustit debugger GDB a příkazem `file` sdělit debuggeru jméno laděného programu tentokrát se zahrnutými ladicími informacemi. Příkazem

```
target <cil>:<port>
```

se spojíme s gdbserverem na cílovém systému a můžeme provádět ladění.

Verze gdbserveru v aktuální verzi uClinuxu čas od času způsobí vygenerování neošetřené výjimky a ladění pak není možné. To je závažné omezení při vzdáleném ladění, které bude potřeba odstranit.

Pro spouštění laděných programů je užitečné umisťovat jejich binární obrazy na síťový disk, čímž se značně urychlí vývojový cyklus programu.

4.4.8 Možnosti dalšího vývoje

V sekci 4.4.7 byly uvedeny některé nedostatky, které současná verze systému obsahuje a které by bylo potřeba odstranit. Další možnosti rozšíření systému jsou následující:

Podpora aplikací reálného času Linux, zejména jeho starší verze, není příliš vhodný pro řízení v reálném čase. Jádro systému by šlo upravit tak, aby přerušení s vysokou prioritou nebyla zpracovávána Linuxem ale pouze nějakým modulem pro řízení v reálném čase. Během své činnosti by jádro toto přerušení nikdy nezakazovalo, čímž by se zaručila dostatečně rychlá odezva na vnější podněty.

Spolupráce s jinými řídicími systémy Pro jednotku MO_376 by bylo možné napsat uživatelské programy, které by ji mohly integrovat do nějakého většího řídicího systému. Pomocí takového programového vybavení by například mohla být jednotka ovládána na dálku, čímž by se vyřešil problém s nefunkčností `ssh`.

Možností dalšího vývoje je samozřejmě více. Síťové rozhraní nabízí zcela novou dimenzi pro komunikaci s jinými systémy, a je jen na naší fantazii, jak lze této možnosti využít.

Závěr

Cílem této práce byl návrh a implementace síťového rozhraní pro řídicí jednotku MO_CPU1 s procesorem Motorola MC68376. Hardwarová část rozhraní je založena na integrovaném ethernetovém řadiči CS8900A. Správnost návrhu byla ověřena stavbou rozhraní a následným otestováním při komunikaci s osobním počítačem.

V softwarové části byla vytvořena podpora pro operační systémy RTEMS a uClinux. Pro OS RTEMS byla napsána testovací aplikace, která umožňuje komunikaci pomocí protokolů TCP/IP a nastavení síťového rozhraní je možné provádět buď ručně, nebo automaticky protokolem DHCP.

Systému uClinux lze rovněž využít ke komunikaci protokoly TCP/IP. Jednotku lze pod tímto systémem vzdáleně ovládat pomocí protokolu telnet, je možno připojovat síťové disky jiných počítačů pomocí protokolu NFS a lze nahrávat rozšiřující moduly do jádra systému Linux. Zejména díky přístupu k síťovým diskům tak dostáváme poměrně dobře konfigurovatelný a snadno rozšiřitelný systém.

Ethernetové rozhraní spolu s dobrou softwarovou podporou nám dává k dispozici velmi silný nástroj pro komunikaci a integraci jednotky s jinými systémy. Možnosti komunikace nejsou omezeny vzdáleností mezi komunikujícími stranami, jako tomu je třeba v případě komunikace po sérové lince a rychlost komunikace je také výrazně vyšší. Díky implementaci protokolů rodiny TCP/IP lze komunikovat s jednotkou přes síť Internet z téměř libovolného místa na světě.

Literatura

[1] ASIX ELECTRONICS CORPORATION. *AX88796 Data Sheet*, June 2002.

[2] BENEDIKT, R. *Projekt Web51*. HW Server. `http://web51.hw.cz/`.

[3] BODAS, D., AND AYERS, J. *CS8900A Technical Reference Manual*. Cirrus Logic, Inc., April 1999. Application Note 83.

[4] CIRRUS LOGIC, INC. *CS8900A Product Data Sheet*, 1999.

[5] CYBERTEC PTY LTD. *My Right Boot, a boot ROM for embedded hardware*. `http://www.cybertec.com.au/`.

[6] DESROSIERS, G. *Embedded Ethernet.com*. Systor Vest, 1999. `http://www.embeddedethernet.com/`.

[7] DIONNE, D. J., AND DURRANT, M. *Projekt uCsimm*. Arcturus Networks Inc. `http://www.uclinux.org/ucsimm/`.

[8] Ethernet. `http://www2.rad.com/networks/2001/ethernet/ether.htm`.

[9] Local Area Networks: CSMA/CD, Std 802.3. Tech. rep., ANSI/IEEE, 2000.

[10] MOTOROLA INC. *CPU32 Reference Manual*, 1990.

[11] MOTOROLA INC. *MC68336/376 User's Manual*, October 2000.

[12] OAR CORPORATION. *RTEMS Documentation*, February 2002.

[13] PÍŠA, P. *BDM Interface for Motorola 683xx MCU, Usage with GDB Debugger*, June 2000. `http://cmp.felk.cvut.cz/~pisa/`.

[14] PÍŠA, P. *Moderní mikrokontroléry a využití vývojového prostředí GNU*, Listopad 2001. `http://cmp.felk.cvut.cz/~pisa/`.

[15] REALTEK SEMI-CONDUCTORS CO., LTD. *RTL8019AS Data Sheet*, January 1996.

[16] SPURGEON, C. Charles Spurgeon's Ethernet Web site.
`http://www.ethermanage.com/ethernet/`.

[17] STANDARD MICROSYSTEMS CORPORATION. *LAN91c111 Data Sheet*, September 2002.

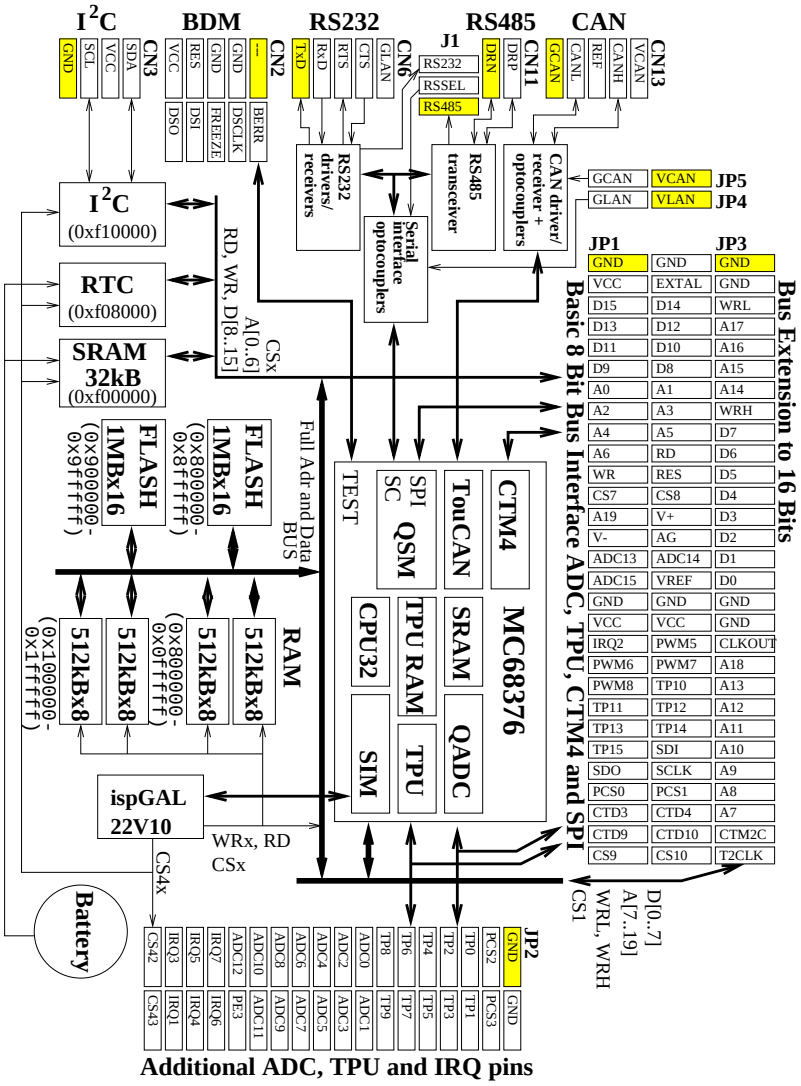
[18] *Projekt uClinux*.
`http://www.uclinux.org/`.

[19] VALOR INC. *ST7010T Data Sheet*.
`http://www.valorinc.com`.

Přílohy

Dodatek A

Blokové schéma MO_CPU1



Dodatek B

Podklady pro výrobu MO_ETH

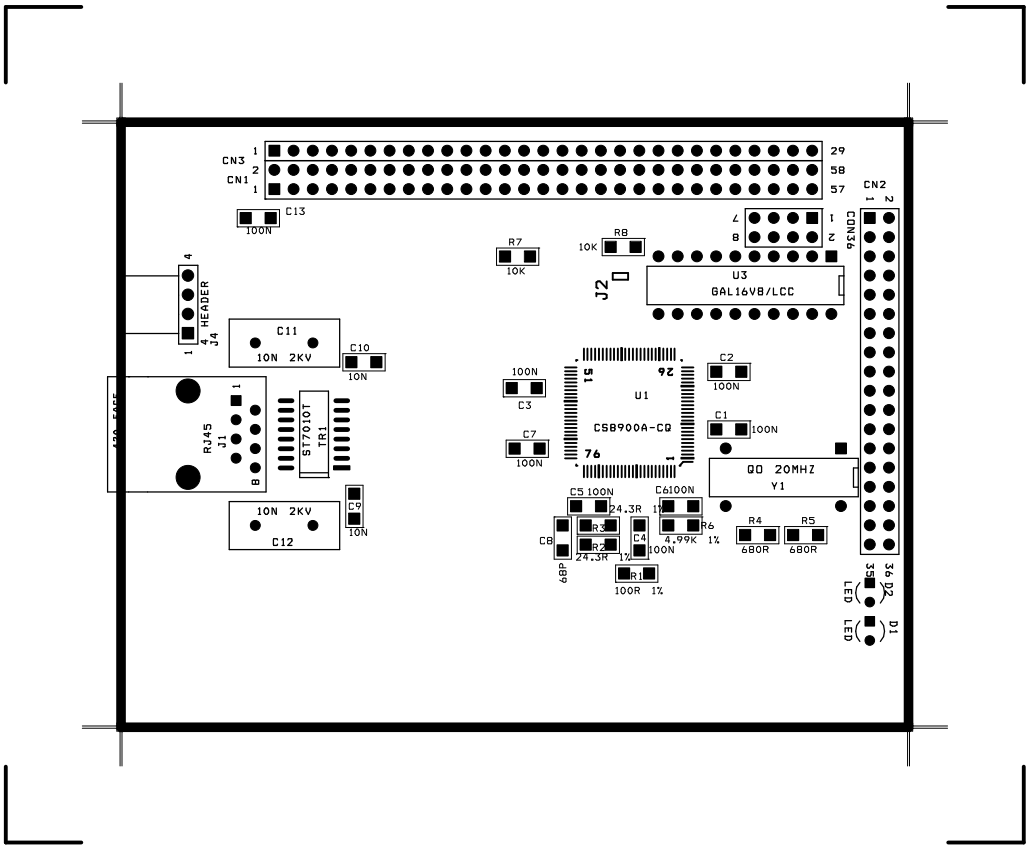
V této příloze jsou podklady potřebné k výrobě jednotky MO_ETH. Seznam potřebných součástek obsahuje tabulka B.1. Motivy tištěného spoje jsou na obrázku B.2 a sestavovací výkres na obrázku B.1. Podklady jsou rovněž k dispozici v elektronické podobě na přiloženém CD.

Množství	Označení	Hodnota	Dostupnost
1	CN1	CON58	
1	CN2	CON36	
1	CN3	CON29	
8	C1, C2, C3, C4, C5, C6, C7, C13	100 nF	
1	C8	68 pF	
2	C10, C9	10 nF	
2	C12, C11	10 nF, 2 kV	TDK (CK45-E3DD103XYNN)
2	D1, D2	LED	
1	J1	RJ45	
1	J2	jumper	
1	J3	4× jumper	
1	J4	CON4, 90	
1	R1	100 Ω, 1%	
2	R2, R3	24,3 Ω, 1%	
2	R4, R5	680 Ω	
1	R6	4.99 kΩ, 1%	
2	R7, R8	10 kΩ	
1	TR1	ST7010T	
1	U1	CS8900A	
1	U3	GAL16V8/LCC	GM Electronic, s. r. o.
1	Y1	QO 20MHz	GM Electronic, s. r. o.

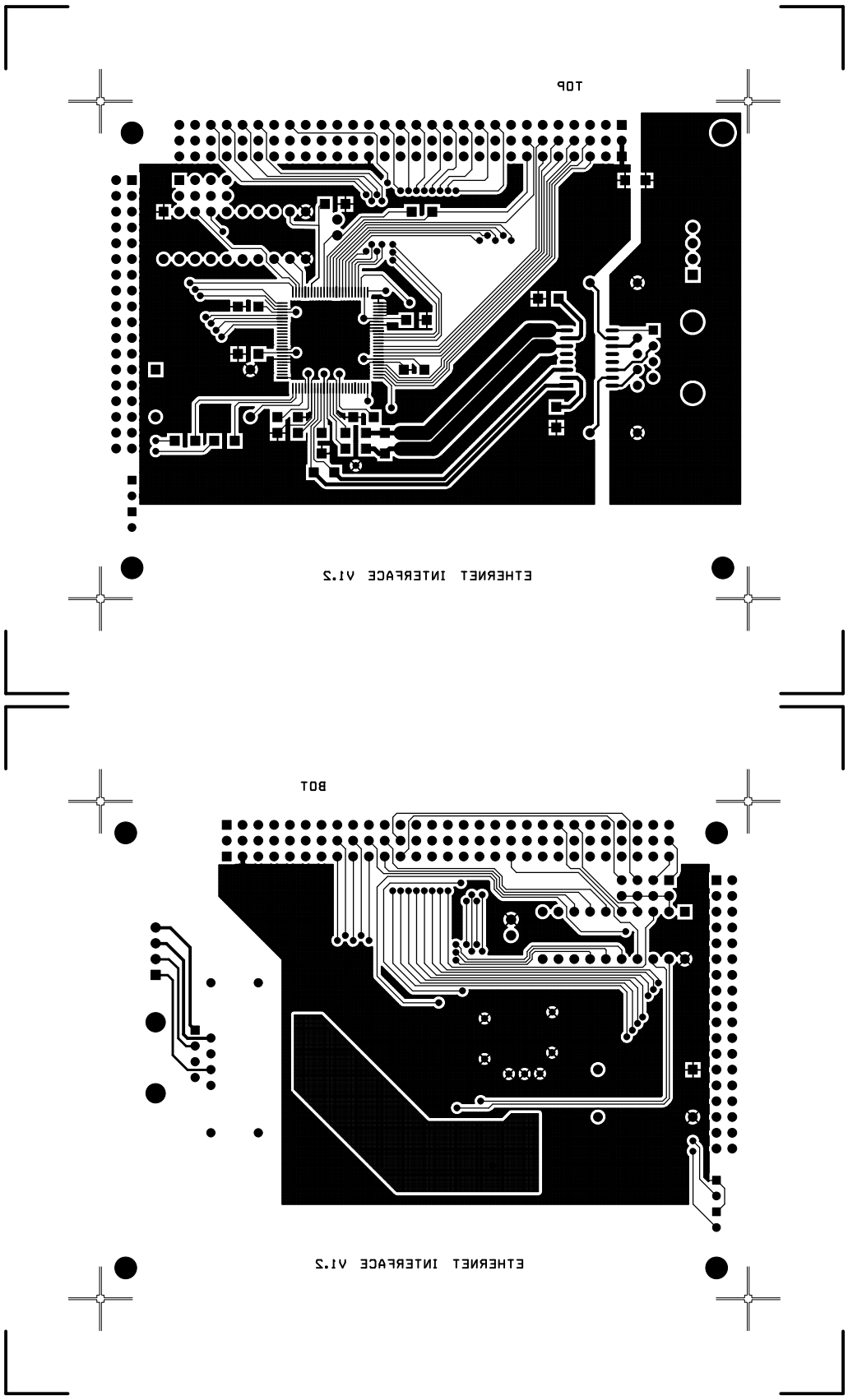
Tabulka B.1: Seznam součástek.

Označení vrtáku	Průměr (mm)	Počet děr
T1	0,5	45
T2	0,8	40
T3	1,0	141
T4	3,2	6

Tabulka B.2: Průměry otvorů po prokovení.



Obrázek B.1: Sestavovací výkres.



Obrázek B.2: Motiv DPS – vrchní a spodní strana.

Dodatek C

Obsah přiloženého CD

`/mo_ether`	Zdrojové kódy testovací aplikace mo_ether.
`/rtems`	Operační systém RTEMS s podporou desky MO_CPU1 a ethernetového rozhraní MO_ETH.
`/uClinux`	Jádro Linuxu a distribuce uClinux s podporou MO_376.