STŘEDNÍ PRŮMYSLOVÁ ŠKOLA SDĚLOVACÍ TECHNIKY

110 00 Praha 1, Panská 856/3 URL: www.panska.cz

☎ 221 002 111, 🖹 221 002 666 e-Mail: sekretariat@panska.cz

MATURITNÍ ZKOUŠKA

PRAKTICKÁ ZKOUŠKA Z ODBORNÝCH PŘEDMĚTŮ

Hardwarový Omega Server

Studijní obor:	**26-45-M/004** **Digitální telekomunikační technika**	
Třída:	**4.A**	**Jan Kapic, Václav Košař**
Školní rok:	**2006/2007**	jméno a příjmení autora

1 ANOTACE:

Cílem naší práce bylo nahradit program Xapi server, určený pro vzdálené programovaní ústředny Ateus Omega přes síť Internet, modulem, který bude zabudován přímo v ústředně, čímž odstraníme problém závislosti na zapnutém počítači. V podstatě se jedná o jednoduchou pasivní síťovou kartu, přičemž tato karta dále komunikuje s ústřednou (Sériový port).

2 ANNOTATION:

The aim of our work has ordered us to substitute the Xapi server application by a HW module, directly built in the phone exchange, which removes the problem of the dependence on switched PC. The Xapi server is designated for distant programming of the phone exchange Ateus Omega via Internet network. It goes in general about a simple passive network card while this card communicates further with the phone exchange (Serial port).

„Prohlašujeme, že jsme tuto práci vypracovali samostatně a použili jsme literárních pramenů a informací, které citujeme a uvádíme v seznamu použité literatury a zdrojů informací.“

V Praze, dne ..... ……..... …….....

Podpis

3 Úvod

Cílem naší dlouhodobé maturitní práce bylo navrhnout a sestavit „hardwarový Xapi server“. Xapi server je softwarová utilita, která umožňuje komunikaci ústředny s ostatními počítači v síti Internet. Nevýhoda tohoto způsobu se skrývá v podstatě celého řešení problému- Xapi server je program a tudíž je závislý na běžícím počítači. My jsme se pokusili tuto nevýhodu odstranit nahrazením Xapi serveru za náš modul, který bude zabudován přímo do ústředny. Základními otázkami celé práce byly:

1. Jak propojit DsPIC s Ethernetem a jak pracovat s jeho protokoly?
2. Jak propojit DsPIC s ústřednou sériovým portem (zapojení konektoru, napěťové úrovně) ?
3. K jakým změnám v datech dochází při přechodu Ethernet – Sériový port?

První otázku jsme vyřešili použitím integrovaného obvodu ENC28J60, který pracuje jako Ethernetový řadič operující na 1. a 2. vrstvě OSI modelu, nastudováním příslušné problematiky a sestavením balíku programů pro DsPIC schopných zpracovávat data vyšších protokolů. Což bylo programově a časově náročné, avšak s kompletní dokumentací do budoucna možné.

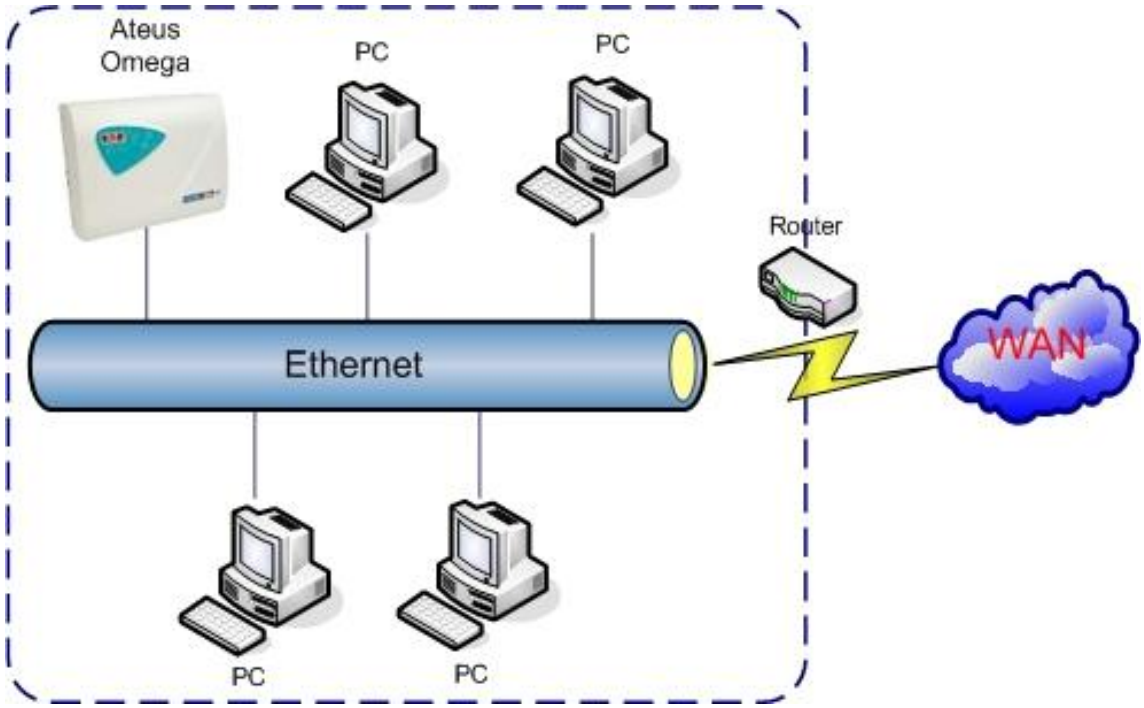
Druhou otázku jsme řešili měřením na konektoru ústředny pro komunikační moduly a překreslením zapojení stávajícího modulu pro komunikaci po RS232.

Třetí problém byl oproti ostatním výrazně komplikovanější. Pokusili jsme se odposlouchávat komunikaci Xapi serveru s oběmi rozhraními, ale výstupních datech jsme nebyli schopni rozpoznat větší návaznosti. Teprve porovnáním dat v TCP paketech při programování pomocí Xapi serveru a dat vysílaných počítačem při „normálním“ programování ústředny přímo použitím Omega programu se jsme začali objevovat souvislosti. Bohužel to nestačilo, proto jsme byli nuceni požádat o originální dokumentaci k sériové komunikaci ústředny od výrobce- firmy 2N. Se získanými katalogovými listy už jsme mohli konečně rozpracovávat první myšlenky celého řešení. Otázkou však stále zůstávalo počítání CRC32.

Následovala již postup v hardwarové a programové konstrukci. Díky cvičné desce jsme uskutečnili první spojení DsPIC – ENC. Vznikli první programy pro sériovou komunikaci. Problém s CRC32 byl vyřešen těsně před vznikem výsledného plošného spoje (MEDEA MODUL) a to především díky velkého nasazení pana profesora Kubalíka (více o CRC32 v příslušné kapitole). Bohužel přestože program pro komunikaci přes TCP/IP byl

téměř kompletní, byli jsme nuceni z důvodu časové tísně zvolit jednodušší variantu, a proto je ústředna programovatelná jen přes UDP.

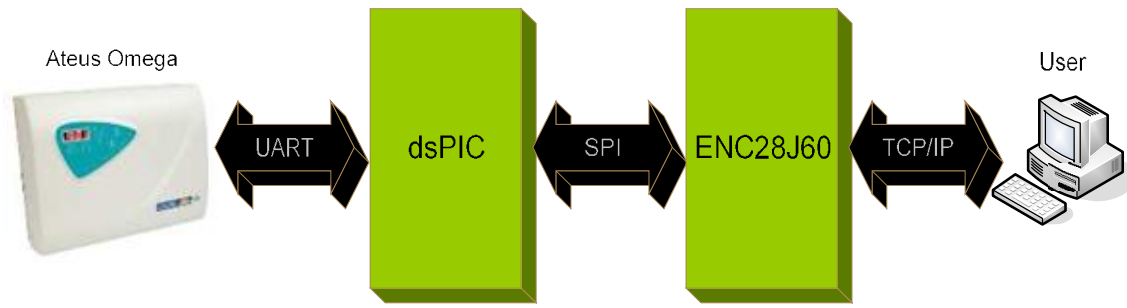
Přesto Médea modul svojí funkci plní – naše práce byla tudíž úspěšná, a to v plném znění zadání.



Použití ústředny v síti Ethernet

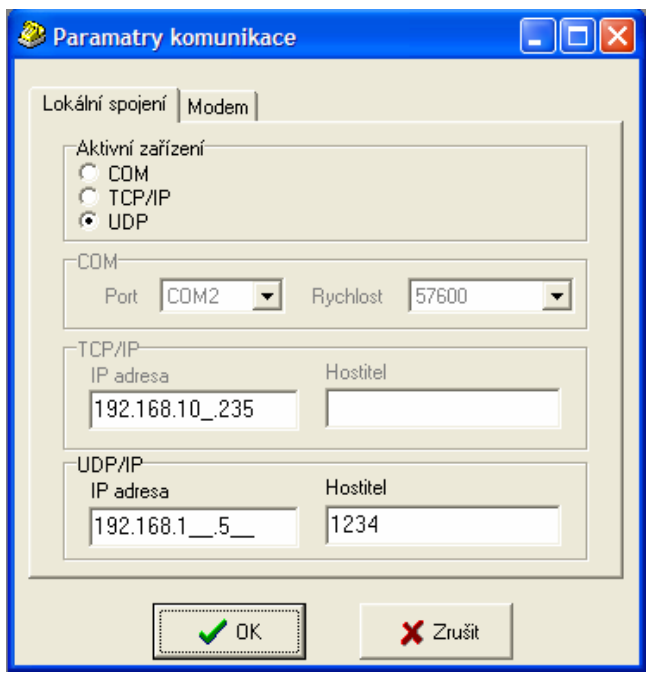
4 Monitorování protokolů

Na obrázku pod textem je vidět prostřednictvím jakých protokolů jednotlivé části modulu komunikují se svým okolím.



Stručný nákres komunikačních protokolů

Omega program umožňuje tři způsoby komunikace s ústřednou: po sériové lince (COM), po UDP a TCP/IP. Nás zajímají poslední dva způsoby realizované na síti TCP/IP.



Náhled konfiguračního okna v Omega programu

4.1 Sériový přenos dat

Jako sériový přenos označujeme takový, kdy se signálové prvky téhož datového proudu předávají za sebou - sériově. Naopak při paralelním přenosu se určitý počet signálových prvků (např. bitů) přenáší současně.

4.1.1 Synchronní a asynchronní sériový přenos

Při přenosu informace (sériovém i paralelním) musíme zabezpečit, aby přijímač správně vyhodnotil okamžiky platnosti jednotlivých značek generovaných vysílačem. Vysílač a přijímač proto spolu musí být nějakým způsobem časově synchronizovány. Právě podle druhu synchronizace rozlišujeme sériový přenos na synchronní a asynchronní.

Synchronní přenos se děje pomocí izochronního signálu, tedy takového, kde odstup dvou libovolných charakteristických okamžiků (např. začátků a konců jednotlivých značek) je celistvým násobkem určitého (apriorně daného) jednotkového intervalu. Komunikační kanál je tedy taktován společným hodinovým signálem (vedeným zvlášť nebo obsaženém v datovém signálu), který vymezuje intervaly platnosti jednotlivých značek. Synchronní přenos se nejčastěji používá u bitově orientovaných protokolů, kde se informace seskupuje do rámců. V datových komunikacích se používá zejména pro přenos větších objemů dat.

Asynchronní (správněji arytmický) přenos - vysílač a přijímač nemají společný hodinový signál, který by vymezoval intervaly platnosti značek. Namísto toho mají obě strany své vlastní hodiny, dostatečně přesné, aby se po fázovém zasynchronizování mohly po několik značkových intervalů považovat za izochronní. Jelikož je třeba hodiny pravidelně synchronizovat, používá se tento způsob nejčastěji pro přenos krátkých bitových posloupností, znaků (5,6,7 nebo 8 bitů). Synchronizace probíhá před každým znakem,i kdy před prvním významovým bitem vždy předchází tzv. startbit, jenž je reprezentován opačnou hodnotou signálu, než je klidová úroveň na lince i a trvá stejnou dobu jako následující bitové intervaly. Arytmický přenos ze své podstaty vhodný např. pro komunikaci s počítačem prostřednictvím terminálu.

Ústředna komunikuje pomocí asynchronního sériového přenosu!!!

4.1.2 Sériová komunikace s ústřednou ATEUS-Omega

Sériová komunikace je použita pro diagnostiku stavu ústředny a jejích částí, programování parametrů, přenos účtovacích dat, a částečně i pro řízení provozu ústředny. Spojení s ústřednou je možné po standardní sériové lince rychlostmi od 9k6 do 57k6 s autodetekcí ze strany ústředny. Komunikace probíhá pomocí poloduplexního posílání paketů.

Hardwarové spojení

Fyzické spojení s ústřednou je realizováno pomocí galvanicky odděleného sériového portu, a to s využitím 4-žilového kabelu se signály Rx,Tx,GND. Tok dat není hardwarově řízen, aplikace na PC tedy musí být schopna přijímat odpovědi od ústředny bez přerušení na zvolené komunikační rychlosti.

Parametry přenosu

Nastavení komunikačního obvodu je vždy 8N1 (bez parity). Přenosová rychlost je ústřednou nastavena při příjmu znaku PREFIX automaticky v rozsahu 9600-57600bps. Na počítačích s obvodem 16550 (UART s FIFO) lze použít vyšší rychlosti.

Protokol sériové komunikace

V ústřednách ATEUS Omega je implementován nový protokol sériové komunikace. Tímto protokolem lze komunikovat po sériové lince i s pomocí modemu. Jedná se o zabezpečený poloduplexní paketový přenos UDP. Ústředna je v každém okamžiku pasivním účastníkem komunikace, tj. nikdy nezačne samovolně vysílat. Komunikace je iniciována aplikací, která vyšle povelový paket a ústředna odpoví po jeho korektním příjmu paketem odpovědním, ve kterém zachová hodnotu identifikátoru NUM. Ústředna nečeká na kladné potvrzení příjmu paketu aplikací, pokud aplikace nedostane odpověď, nebo je odpověď přenosem znehodnocena může poslat nový dotaz. Doba reakce ústředny je závislá na provozním zatížení a pohybuje se v mezích 5-300ms. Během reakční doby ústředna ignoruje jakýkoli další příjem. Během následného vysílání odpovědi je libovolný příjem považován za STOP a ruší vysílání. Ke zrychlení ošetření chyb komunikačního

kanálu jsou zavedeny plně duplexní nepaketové odpovědi (NAK) o délce 1byte, které ústředna vysílá po příjmu každého byte, který nezapadá do protokolu.

A. Struktura paketu UDP:

PREFIX	START	LEN_L	LEN_H	NUM	DATA	CRC32
1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	max. 509 Byte	4 Byte

PREFIX	znak EEh
START	znak 23h
LEN	délka NUM + DATA
NUM	identifikátor paketů (párování povel/odpověď)
DATA	posílaná data (viz formát povelů ústředny)
CRC-32	zabezpečení podle tabulky CRC32 (big endian).

Pozn.:

1) Pokud se kdekoli za znakem START vyskytuje znak EEh (PREFIX) je nahrazen **dvěma** znaky EEh. Na přijímací straně je jeden ze znaků vypuštěn a není samozřejmě ani započítán do CRC.

2) CRC zajišťuje NUM a DATA a posílá se ve formě big endian (LSB první).

3) Délka pole DATA je závislá na velikosti alokované paměti v ústředně a může se v budoucích verzích zvětšit. Hodnota 509B je volena kvůli kompatibilitě se staršími verzemi a umožňuje posílání všech dosud používaných struktur i nových dat pro upgrade SW.

B. Asynchronní odpovědi:

Pro urychlení zotavení z chyby komunikace jsou nově zavedeny asynchronní 1bytové odpovědi. Porušení protokolu může nastat v následujících případech:

Stav rozhranní	Chybný příjem	Reakce	Pozn.
KLID	znak<>PREFIX	NAK	vždy na 57600 bps
PŘÍJEM	PREFIX+START	-	restart příjmu bez odpovědi
PŘÍJEM	PREFIX bez opakování	NAK+KLID	na rychlosti PC
PŘÍJEM	přetečení délky	NAK+KLID	
PŘÍJEM	chybný CRC	NAK+KLID	CRC je porovnáván po bytech
PŘÍJEM	-	-	nedostatečný počet přijatých znaků
REAKCE	cokoli	-	příjem je ignorován
VYSÍLÁNÍ	cokoli	KLID	příjem ruší vysílání (data ztracena)

Stav komunikačního rozhranní ústředny:

KLID očekávání příjmu 1.znaku (PREFIX)

PŘÍJEM pokračující příjem po příjmu nejméně 1 znaku

REAKCE doba určená pro zpracování bezchybně přijatého

VYSÍLÁNÍ stav kdy je odesílán odpovědní paket

Při příjmu nedostatečného počtu znaků (např. při přerušení kabelu uprostřed příjmu) zůstává ústředna ve stavu PŘÍJEM po neomezeně dlouhou dobu, což však není na škodu, neb aplikace po vlastním časovém dohledu posílá opakovaný povel a ústředna je schopna restartovat příjem kdykoli po příjmu sekvence PREFIX+START. Pokud mezitím aplikace změní komunikační rychlost, je 1. paket přijat jako neplatný a ústředna odpoví vysláním NAK na předchozí rychlosti. Opakovaný paket je však již přijat korektně.

Odpověď **NAK = 0E0h** je určena pro urychlení zotavení z chyby komunikace a pro detekci průchodnosti kanálu. Znak NAK je volen tak, aby byl detekovatelný aplikací na všech přípustných rychlostech příjmu i v případě, že dojde ke zkomolení 1. znaku paketu (PREFIX) a ústředna v tomto případě pošle NAK na rychlosti 57600 bps. Aplikace tedy pouze testuje příjem během svého vysílání a při příjmu libovolného byte může vysílání ihned restartovat.

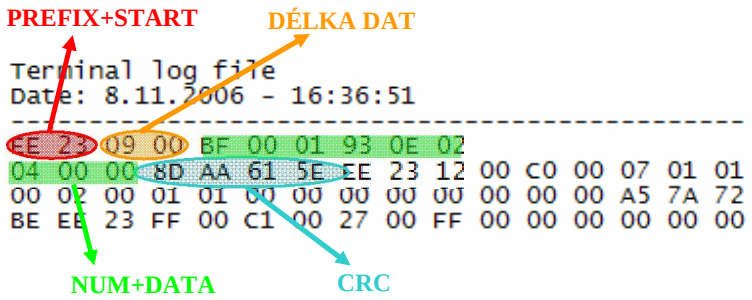
Nastane-li chyba při vysílání z aplikace, ústředna okamžitě odpoví NAK a aplikace by měla být schopna okamžitě restartovat vysílání. Detekuje-li chybu aplikace při příjmu, může zastavit vysílání vysláním STOP (libovolný znak) a po vypláchnutí svého přijímacího bufferu okamžitě opakovat vyslání žádosti (pozor na Windows a 16550). Nastane-li chyba ztrátou znaků na lince, nebo výpadkem jednoho ze směrů komunikace, musí aplikace reagovat časovým dohledem a opakováním žádosti. Nezdaří-li se několikrát opakovaná žádost, je třeba zobrazit příčinu a případně zahájit detekci průchodnosti kanálu, t.j. cyklicky vysílat korektní paket (např. INFO) až se dočká korektní paketové odpovědi. Příčinou se rozumí buď žádná odpověď od ústředny (zkontrolujte stav kabelu a výběr COM), nebo příjem znaků NAK (chybující kanál, snižte komunikační rychlost).

C. Změna komunikační rychlosti:

Ústředna je schopna přepnout se na rychlost zvolenou aplikací na PC během příjmu 1. znaku paketu (PREFIX). Pokud je tento znak zkomolen, odpoví na něj ústředna vysláním NAK na rychlosti 57600 bps. Vzhledem k volbě NAK je i při příjmové rychlosti PC 9600 bps tento znak přijat jalo **0FFh** a aplikace může okamžitě opakovat vysílání. Ústředna detekuje rychlost během každého 1. znaku paketu a pokud je to PREFIX pokračuje v příjmu na zvolené rychlosti a také na této rychlosti odpoví. Po odvysílání odpovědi se vždy přepne na 57600 bps a očekává další paket.

4.1.3 Rozebrání paketu

Ukázka odchycených paketů během programování.



Log z programu Terminal

4.2 SPI rozhraní

SPI je sériové periferní rozhraní. Používá se pro komunikaci mezi řídícími mikroprocesory a ostatními integrovanými obvody (EEPROM, A/D převodníky, displeje...).

4.2.1 Rozdělení zařízení na sériové SPI sběrnici

Master

- řídí komunikaci pomocí hodinového signálu
- určuje se kterým zařízením na sběrnici bude komunikovat pomocí SS-Slave Select(někdy CS-Chip Select)

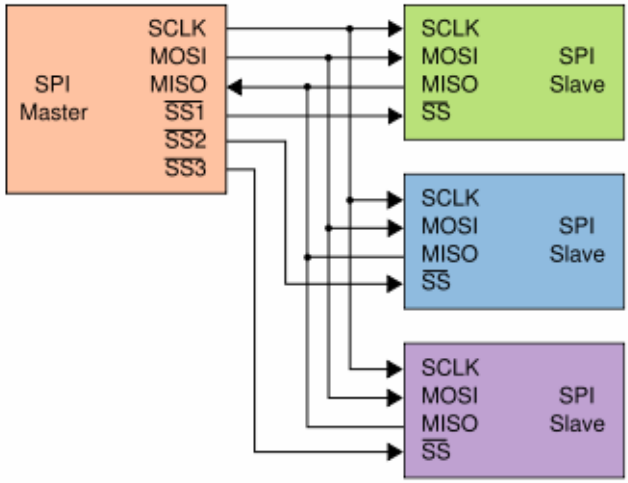
Slave

- vysílá podle hodinového signálu, pokud je aktivován pomocí SS/CS

Komunikace:

Pro komunikaci Master nastaví log.0 na **SS** zařízení, se kterým chce komunikovat.
Pak začne generovat hodinový signál na **SCLK** a v té chvíli vyšlou obě zařízení svoje data, přičemž **MISO** je vždy Master výstup, Slave vstup a **MOSI** je Master Vstup, Slave výstup.
Jakmile jsou data vyslána muže komunikace dále pokračovat: *Master dále dodává hodinový signál, hodnota SS se nemění* nebo může být ukončena: *Master přestane vysílat hodinový signál a nastaví SS do log.1.*

Délka vyslaných dat je buď 8bit (Byte) a nebo 16bit (Word).



Polarita a fáze hodinového signálu

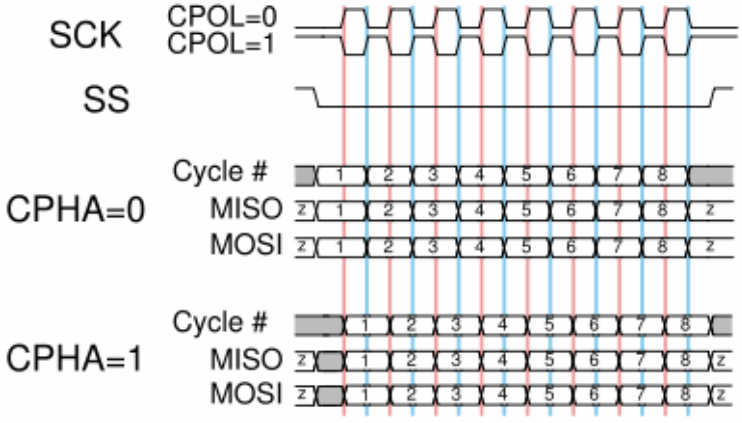
Vztah mezi hodinovým signálem a daty se určuje dvěma konfiguračními bity:CPOL a CPHA.

CPOL= 0; klidová úroveň hodinového signálu log.0

CPOL = 1; klidová úroveň hod.sig. je log.1

CPHA = 0; hodnota je čtena při vzestupné hraně

CPHA = 1; hodnota je čtena při sestupné hraně



4.3 TCP/IP

(Komunikace Medea modul => User)

4.3.1 Protokoly TCP/IP

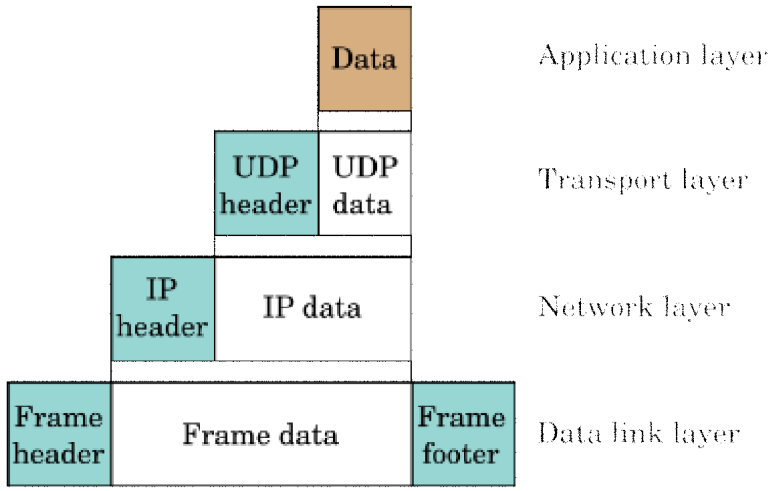
(Stručný popis zaměřený k části použité v naší práci)

Přirovnání k referenčnímu modelu OSI

(Zařazení protokolů do jednotlivých vrstev. Zvýrazněné protokoly jsou obsaženy v naší práci)

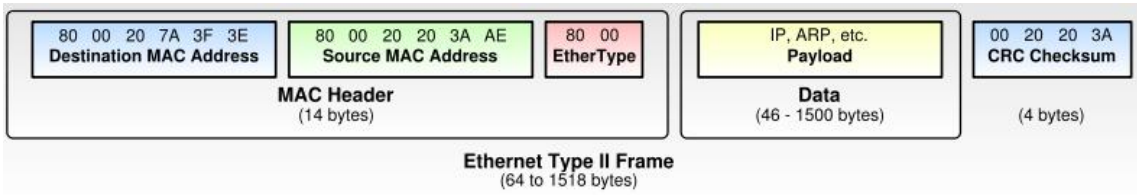
7	**Application**	**HTTP**, SMTP, SNMP, FTP, Telnet, ECHO, SIP, SSH, NFS, RTSP, XMPP, Whois, ENRP
6	**Presentation**	XDR, ASN.1, SMB, AFP, NCP
5	**Session**	ASAP, TLS, SSL, ISO 8327 / CCITT X.225, RPC, NetBIOS, ASP
4	**Transport**	**TCP, UDP,** RTP, SCTP, SPX, ATP, IL
3	**Network**	**IP, ICMP**, **ARP**,IGMP, IPX, OSPF, RIP, IGRP, EIGRP, RARP, X.25
2	**Data Link**	**Ethernet**, Token ring, HDLC, Frame relay, ISDN, ATM, 802.11 WiFi, FDDI, PPP
1	**Physical**	**10BASE-T**, 100BASE-T, 1000BASE-T, SONET/SDH, G.709, T-carrier/E-carrier, various 802.11 physical layers

Příklad stavby rámce



Ethernetový Rámec (Frame)

Adresování: MAC adresa; ochranný kód (Frame check seq.) CRC32 (čtyř bajtový cyklický redundantní kód); max. délka 1518 bajtů



Obsahem dat rámce může být ARP paket, nebo IP záhlaví.

ARP paket

Užívá se většinou k zjištění MAC adresy cílového počítače za předpokladu, že známe jeho IP adresu. Pole Operace zpravidla obsahuje bity poznačující paket buď jako otázku na MAC (v tom případě se paket vysílá broadcast tj. všem) a nebo jako odpověď s vyplněnou požadovanou MAC (pak se paket vysílá unicast, tj. jen tazateli)

+	Bits 0 - 7	8 - 15	16 - 31
0	Typ hardwarového potokolu (HTYPE)		Typ protokolu (PTYPE)
32	Délka hardwarové adresy (HLEN)	Délka protokolové adresy (PLEN)	Operace (OPER)
64	Zdrojová Hardwarová adresa (zpravidla MAC)		
?	Zdrojová protokolová adresa (zpravidla IP)		
?	Cílová Hardwarová adresa (zpravidla MAC)		
?	Zdrojová protokolová adresa (zpravidla IP)		

IP záhlaví

Adresování: IP adresa, záhlaví má proměnou délku díky poli Options; ochranný prvek: kontrolní součet záhlaví viz: Kapitola ENC28J60-Počítání kontrolního součtu

+	Bits 0–3	4–7	8–15	16–18	19–31
0	Verse	Délka záhlaví	Typ služby	Úplná délka paketu	
32	Identification			Flags	Fragment Offset
64	Time to Live		Protocol	Kontrolní součet záhlaví	
96	Zdrojová IP adresa				
128	Cílová IP adresa				
160	Options				
160 or 192+	Data				

Obsahem dat po IP záhlaví může být řada různých protokolů zmíním se jen o ICMP, TCP, UDP.

ICMP protokol

ICMP je protokol určený pro chybové a řídící zprávy. (Echo, Destination Unreachable, Redirect atd.) Typ zprávy „Echo“ se používán programem ping.

0		31
Typ zprávy	Kód	Kontrolní součet zprávy
Identifier		**Sequence Number**
Data :::		

UDP protokol

Protokol UDP poskytuje nespojovou a nespolehlivou službu přenosu zpráv.
Adresování: Porty; ochrana: Kontrolní součet včetně Pseoudo záhlaví.

+	**Bits 0 - 15**	**16 - 31**
0	Zdrojový Port	Cílový Port
32	Délka	Kontrolní součet
64	Data	

TCP protokol

Protokol TCP poskytuje spojovou a nespolehlivou službu přenosu zpráv.
Adresování: Porty; ochrana: Kontrolní součet včetně Pseoudo záhlaví.
Spojení se vytváří synchronizací sekvenčních čísel (Three-way handshake)

Sekvenční číslo vždy obsahuje offset prvního datového bajtu tohoto segmentu vzhledem k celé sekvenci dat určených k vysílání

Potvrzovací číslo obsahuje, pokud je nastaven flag ACK, sekvenční číslo následujícího bajtu, který je přijímač připraven přijmout.

+	Bits 0–3	4–7	8–15	16–31
0	Zdrojový Port			Cílový Port
32	Sekvenční (Sequence) číslo			
64	Potvrzovací (Acknowledgment) číslo			
96	Data Offset	Reserved	Flags	Okno
128	Kontrolní součet			Urgent Pointer
160	Options (optional)			
160/192+	Data			

4.3.2 Data v paketech

Při monitorování se zjistilo, že UDP pakety mají jiný obsah dat, než pakety TCP. TCP pakety neobsahovaly znaky PREFIX, START a CRC (viz. 4.1.2), proto se data musí před odesíláním doplnit. Také chybí nahrazení znaku EE dvěma znaky EE. Je nutné testovat data a když se kdekoliv za znakem START vyskytne znak EE, je zdvojen. Data v UDP jsou kompletní a mají všechny náležitosti.

Pozn: XAPI server neumí komunikovat po UDP.

4.3.3 CRC kódy

CRC neboli Cyclic Redundance Check je speciální hašovací funkce, používaná k detekci chyb během přenosu či ukládání dat. CRC je vypočten před operací, u níž jsou předpokládány chyby. Je odeslán či uložen spolu s daty. Po převzetí dat je z nich nezávisle spočítán znovu. Pokud vyjde různý CRC, je přenos prohlášen za chybový. V určitých případech je možné chybu pomocí CRC opravit. Celý proces je svázán s tzv. generujícím polynomem, který je klíč k celému procesu.

Generující polynom (GP) je obecně n-bitové číslo, kde n je z praktických důvodů mocnina 2. Zásadně tedy počítáme CRC-16bit, CRC-32bit; CRC-64bit nebo CRC-128bit, apod. .
Čím je CRC "více-bitové", tím je šance rozpoznání chyby větší. GP si většinou sami nevymýšlíme, ale používáme již zavedené polynomy s ohledem na typ přenosu a hloubku zabezpečení.

Příklad GP vidíte zde:

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

Tento případ je ukázka GP pro CRC-32bit. Používá se podle IEEE 802 pro Ethernet a ne náhodou i pro naší ústřednu.

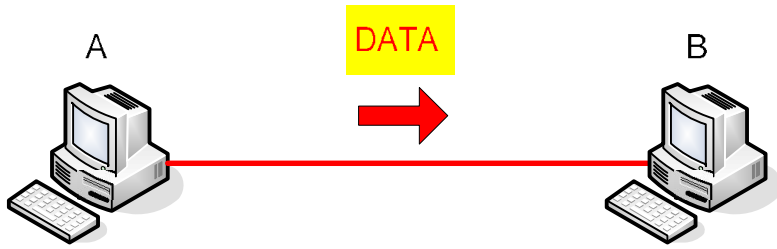
Hlavní operací, kterou budete při počítání CRC potřebovat, je XOR:

Pro připomenutí :

```
0 xor 1 = 1
1 xor 0 = 1
0 xor 0 = 0
1 xor 1 = 0
```

Vysvětlení výpočtu osvětlím na praktickém příkladu:

Posíláme data z bodu A do bodu B :



Volím data: $x^7 + x^6 + x^3 + x^0$ => **11001001**

Volím GP: $x^3 + x^2 + x^0$ => **1101**

BOD A

Máme tedy data 11001001, která zabezpečíme polynomem 1101. První krok je „přidání“ nul na konec dat. Počet nul závisí na nejvyšším členu GP, v našem případě jsou to nuly 3 (x^3). Dalším krokem bude dělení (upozorňuji, že se nejedná o obyčejné dělení, ale o již výše zmíněný XOR!). Důležité je taky to, že po „vydělení“ nás zajímá nikoliv výsledek, ale zbytek, proto si výsledku nebudeme všímat. A teď už čistý výpočet.

Postup výpočtu:

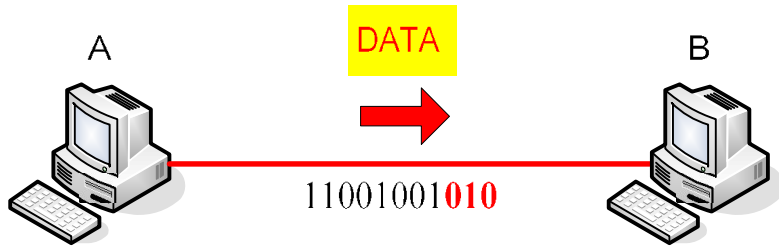
```
11001001000 : 1101 = 10010010
1101
0001100
   1101
   00011
    1101
    1110
    1101
     00110
       1101
       00010
```

11001001000 : 1101 = 10010010, zbytek = **010**

Poté, co nám vyjde zbytek, nahradíme námi přidané nuly za něho.

11001001000 => 11001001**010**

A to už jsou naše zabezpečená data připravena k vyslání.



BOD B

Na přijímací straně se data vydělí stejným GP a když vyjde zbytek nulový, došli data do bodu B nepoškozeny.

11001001010 : 1101 = 10010100
1101
0001**100**
1101
0001**101**
1101
0000

V případě, že zbytek nevyjde nulový, již víme, že něco není v pořádku. Máme několik možností. Jedna z nich je poslat požadavek o zopakování poslední zprávy, nebo pouze dát na vědomí vysílací straně, že data nedošla správně. Na špatné CRC, reaguje ústředna odpovědí NAK pro chybné CRC.

5 Integrovaný obvod ENC28J60

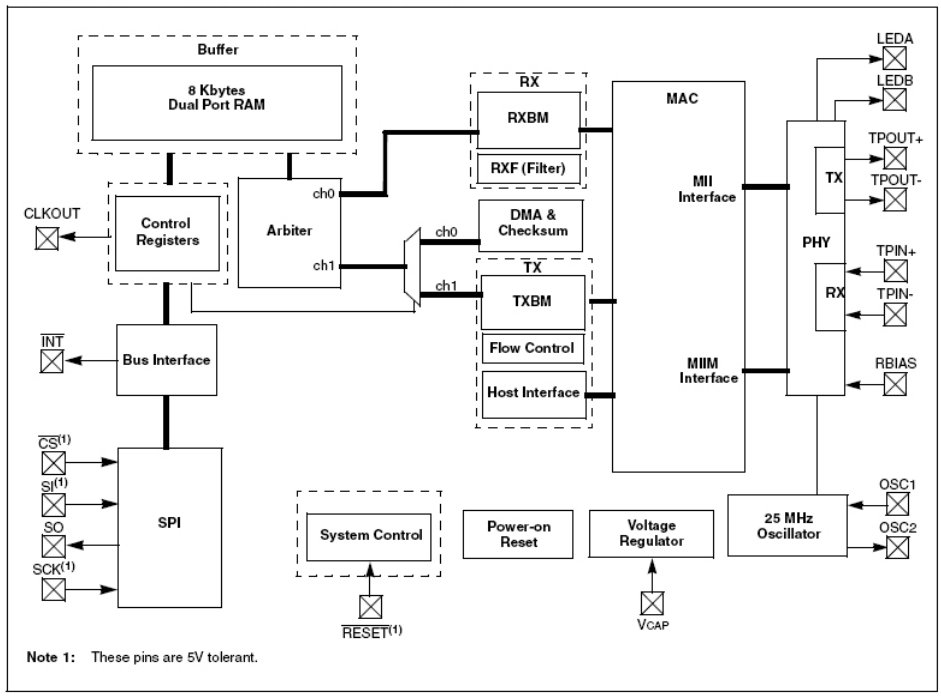
Tato část je překladem výběru některých částí z katalogových listů od výrobce.

5.1 Základní parametry ENC28J60

- ENC28J60(dále v textu pod zkratkou ENC) je integrovaný obvod kompatibilní s průmyslovými standardy IEEE 802.3 a Ethernet operující na první a druhé vrstvě OSI modelu. Je určen pro technologii 10BASE-T, což je standard používající kabeláž UTP a STP (Kroucená dvojlinka) kategorie 3 a konektory RJ45 určené pro hvězdicovou topologii sítě, jejímiž základními síťovými prvky jsou Rozbočovač (HUB) a Přepínač (Switch). Maximální délka segmentu 100m (328feet). Maximální přenosová rychlost je 10 Mbps. Max. impedance:100Ω.
- ENC má …
  - jeden IEEE 802.3 port disponující automatickou detekcí polarity a její následnou korekcí
  - SPI rozhraní pro komunikaci s řídícím mikro-procesorem.
  - vestavenou cyklickou 8KB paměť (Buffer) programovatelně rozdělenou na Vysílací(Transmit) a Přijímací(Recieve) část.
  - implementovaný polynom a mechanismus na počítání 16bit kontrolní součet pro IP protokoly.
- Podporuje oba typy přístupu k médiu; Plně-Duplexní i Polo-Duplexní. Typ přístupu je třeba nastavit ručně. ENC samostatně nedokáže detekovat o který typ se v daném spojení jedná. Při zapnutém Polo-Duplexním typu muže být ENC nastaven tak, aby při vzniklé kolizi automaticky sám vysílal rámec znovu, dokud se ho nepodaří odeslat, nebo dokud nepřesáhne počet kolizí kritický limit.
-
- ENC je schopné také samo přidávat do rámce Padding(vycpávku) a CRC32, který automaticky generuje. Dále obsahuje řadu programovatelných přijímacích filtrů (Unicast, Multicast, Broadcast, CRC ok, Huge Frame (Příliš velký rámec), Pattermatch (Shoda obsahu), Magic Packet a Hash Table Filtr (CRC na cílovou MAC adresu příchozího paketu, kontrola kriteria)
- Elektrické parametry:
  - Napájení: 3.1 – 3,6 [V]

- 5V TTL vstupy, výstupy 3,3V logika

FIGURE 1-1: ENC28J60 BLOCK DIAGRAM



5.1.1 Paměť ENC28J60:

Veškerá paměť ENC je realizovaná jako statická RAM. Rozděluje se do tři typů.

Kontrolní registry(Control reg:

-registry používané pro konfiguraci a zjišťování stavu ENC.

-jsou rozdělují se do 4 stránek

-přistupuje se k nim přímo přes SPI rozhraní

-příkazy Write/Read Control reg.

Ethernet Buffer 8KB

-je vysílací i přijímací paměť, s jednotlivým přístupem

-rozdělení paměti je programovatelné čtyřmi kontrolními registry

-zápis čtení přes přímo SPI(Write/Read Buffer mem.)

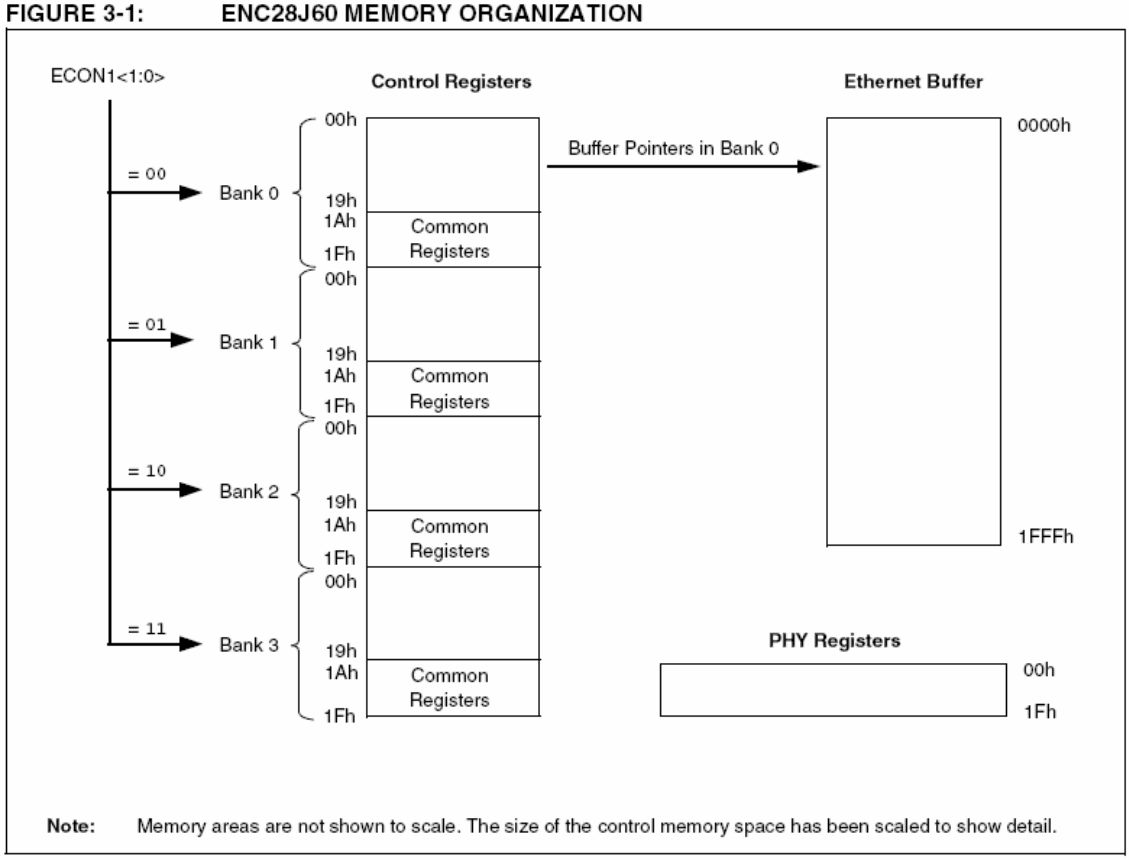
-změna automaticky inkrementujicích pointerů(write/read) přes Kontrolní reg.

PHY registry

-registry používané pro konfiguraci, kontrolu stavu, kontrolu Fyzické stavu vrstvy

-přístup zápisem přes Kontrolní registry

FIGURE 3-1: ENC28J60 MEMORY ORGANIZATION



Rozdělení Kontrolních registrů na stránky

TABLE 3-1: ENC28J60 CONTROL REGISTER MAP

Bank 0		Bank 1		Bank 2		Bank 3	
Address	Name	Address	Name	Address	Name	Address	Name
00h	ERDPTL	00h	EHT0	00h	MACON1	00h	MAADR5
01h	ERDPTH	01h	EHT1	01h	Reserved	01h	MAADR6
02h	EWRPTL	02h	EHT2	02h	MACON3	02h	MAADR3
03h	EWRPTH	03h	EHT3	03h	MACON4	03h	MAADR4
04h	ETXSTL	04h	EHT4	04h	MABBIPG	04h	MAADR1
05h	ETXSTH	05h	EHT5	05h	—	05h	MAADR2
06h	ETXNDL	06h	EHT6	06h	MAIPGL	06h	EBSTSD
07h	ETXNDH	07h	EHT7	07h	MAIPGH	07h	EBSTCON
08h	ERXSTL	08h	EPMM0	08h	MACLCON1	08h	EBSTCSL
09h	ERXSTH	09h	EPMM1	09h	MACLCON2	09h	EBSTCSH
0Ah	ERXNDL	0Ah	EPMM2	0Ah	MAMXFLL	0Ah	MISTAT
0Bh	ERXNDH	0Bh	EPMM3	0Bh	MAMXFLH	0Bh	—
0Ch	ERXRDPTL	0Ch	EPMM4	0Ch	Reserved	0Ch	—
0Dh	ERXRDPTH	0Dh	EPMM5	0Dh	Reserved	0Dh	—
0Eh	ERXWRPTL	0Eh	EPMM6	0Eh	Reserved	0Eh	—
0Fh	ERXWRPTH	0Fh	EPMM7	0Fh	—	0Fh	—
10h	EDMASTL	10h	EPMCSL	10h	Reserved	10h	—
11h	EDMASTH	11h	EPMCSH	11h	Reserved	11h	—
12h	EDMANDL	12h	—	12h	MICMD	12h	EREVID
13h	EDMANDH	13h	—	13h	—	13h	—
14h	EDMADSTL	14h	EPMOL	14h	MIREGADR	14h	—
15h	EDMADSTH	15h	EPMOH	15h	Reserved	15h	ECOCON
16h	EDMACSL	16h	Reserved	16h	MIWRL	16h	Reserved
17h	EDMACSH	17h	Reserved	17h	MIWRH	17h	EFLOCON
18h	—	18h	ERXFCON	18h	MIRDL	18h	EPAUSL
19h	—	19h	EPKTCNT	19h	MIRDH	19h	EPAUSH
1Ah	Reserved	1Ah	Reserved	1Ah	Reserved	1Ah	Reserved
1Bh	EIE	1Bh	EIE	1Bh	EIE	1Bh	EIE
1Ch	EIR	1Ch	EIR	1Ch	EIR	1Ch	EIR
1Dh	ESTAT	1Dh	ESTAT	1Dh	ESTAT	1Dh	ESTAT
1Eh	ECON2	1Eh	ECON2	1Eh	ECON2	1Eh	ECON2
1Fh	ECON1	1Fh	ECON1	1Fh	ECON1	1Fh	ECON1

Souhrn všech Kontrolních registrů, jejich jednotlivé bity a hodnoty po resetu

TABLE 3-2: ENC28J60 CONTROL REGISTER SUMMARY

Register Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Reset	Details on Page
EIE	INTIE	PKTIE	DMAIE	LINKIE	TXIE	r	TXERIE	RXERIE	0000 0000	65
EIR	—	PKTIF	DMAIF	LINKIF	TXIF	r	TXERIF	RXERIF	-000 0000	66
ESTAT	INT	BUFER	r	LATECOL	—	RXBUSY	TXABRT	CLKRDY(1)	0000 -000	64
ECON2	AUTOINC	PKTDEC	PWRSV	r	VRPS	—	—	—	1000 0---	16
ECON1	TXRST	RXRST	DMAST	CSUMEN	TXRTS	RXEN	BSEL1	BSEL0	0000 0000	15
ERDPTL	Read Pointer Low Byte ERDPT<7:0>								1111 1010	17
ERDPTH	—	—	—	Read Pointer High Byte (ERDPT<12:8>)					---0 0101	17
EWRPTL	Write Pointer Low Byte (EWRPT<7:0>)								0000 0000	17
EWRPTH	—	—	—	Write Pointer High Byte (EWRPT<12:8>)					---0 0000	17
ETXSTL	TX Start Low Byte (ETXST<7:0>)								0000 0000	17
ETXSTH	—	—	—	TX Start High Byte (ETXST<12:8>)					---0 0000	17
ETXNDL	TX End Low Byte (ETXND<7:0>)								0000 0000	17
ETXNDH	—	—	—	TX End High Byte (ETXND<12:8>)					---0 0000	17
ERXSTL	RX Start Low Byte (ERXST<7:0>)								1111 1010	17
ERXSTH	—	—	—	RX Start High Byte (ERXST<12:8>)					---0 0101	17
ERXNDL	RX End Low Byte (ERXND<7:0>)								1111 1111	17
ERXNDH	—	—	—	RX End High Byte (ERXND<12:8>)					---1 1111	17
ERXRDPTL	RX RD Pointer Low Byte (ERXRDPT<7:0>)								1111 1010	17
ERXRDPTH	—	—	—	RX RD Pointer High Byte (ERXRDPT<12:8>)					---0 0101	17
ERXWRPTL	RX WR Pointer Low Byte (ERXWRPT<7:0>)								0000 0000	17
ERXWRPTH	—	—	—	RX WR Pointer High Byte (ERXWRPT<12:8>)					---0 0000	17
EDMASTL	DMA Start Low Byte (EDMAST<7:0>)								0000 0000	71
EDMASTH	—	—	—	DMA Start High Byte (EDMAST<12:8>)					---0 0000	71
EDMANDL	DMA End Low Byte (EDMAND<7:0>)								0000 0000	71
EDMANDH	—	—	—	DMA End High Byte (EDMAND<12:8>)					---0 0000	71
EDMADSTL	DMA Destination Low Byte (EDMADST<7:0>)								0000 0000	71
EDMADSTH	—	—	—	DMA Destination High Byte (EDMADST<12:8>)					---0 0000	71
EDMACSL	DMA Checksum Low Byte (EDMACS<7:0>)								0000 0000	72
EDMACSH	DMA Checksum High Byte (EDMACS<15:8>)								0000 0000	72
EHT0	Hash Table Byte 0 (EHT<7:0>)								0000 0000	52
EHT1	Hash Table Byte 1 (EHT<15:8>)								0000 0000	52
EHT2	Hash Table Byte 2 (EHT<23:16>)								0000 0000	52
EHT3	Hash Table Byte 3 (EHT<31:24>)								0000 0000	52
EHT4	Hash Table Byte 4 (EHT<39:32>)								0000 0000	52
EHT5	Hash Table Byte 5 (EHT<47:40>)								0000 0000	52
EHT6	Hash Table Byte 6 (EHT<55:48>)								0000 0000	52
EHT7	Hash Table Byte 7 (EHT<63:56>)								0000 0000	52
EPMM0	Pattern Match Mask Byte 0 (EPMM<7:0>)								0000 0000	51
EPMM1	Pattern Match Mask Byte 1 (EPMM<15:8>)								0000 0000	51
EPMM2	Pattern Match Mask Byte 2 (EPMM<23:16>)								0000 0000	51
EPMM3	Pattern Match Mask Byte 3 (EPMM<31:24>)								0000 0000	51
EPMM4	Pattern Match Mask Byte 4 (EPMM<39:32>)								0000 0000	51
EPMM5	Pattern Match Mask Byte 5 (EPMM<47:40>)								0000 0000	51
EPMM6	Pattern Match Mask Byte 6 (EPMM<55:48>)								0000 0000	51
EPMM7	Pattern Match Mask Byte 7 (EPMM<63:56>)								0000 0000	51

Legend: x = unknown, u = unchanged, — = unimplemented, q = value depends on condition, r = reserved, do not modify.

Note 1: CLKRDY resets to '0' on Power-on Reset but is unaffected on all other Resets.

2: EREVID is a read-only register.

3: ECOCON resets to '---- -100' on Power-on Reset and '---- -uuu' on all other Resets.

TABLE 3-2: ENC28J60 CONTROL REGISTER SUMMARY (CONTINUED)

Register Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Reset	Details on Page
EPMCSL	Pattern Match Checksum Low Byte (EPMCS<7:0>)								0000 0000	51
EPMCSH	Pattern Match Checksum High Byte (EPMCS<15:0>)								0000 0000	51
EPMOL	Pattern Match Offset Low Byte (EPMO<7:0>)								0000 0000	51
EPMOH	—	—	—	Pattern Match Offset High Byte (EPMO<12:8>)					---0 0000	51
ERXFCON	UCEN	ANDOR	CRCEN	PMEN	MPEN	HTEN	MCEN	BCEN	1010 0001	48
EPKTCNT	Ethernet Packet Count								0000 0000	43
MACON1	—	—	—	r	TXPAUS	RXPAUS	PASSALL	MARXEN	---0 0000	34
MACON3	PADCFG2	PADCFG1	PADCFG0	TXCRCEN	PHDREN	HFRMEN	FRMLNEN	FULDPX	0000 0000	35
MACON4	—	DEFER	BPEN	NOBKOFF	—	—	r	r	-000 --00	36
MABBIPG	—	Back-to-Back Inter-Packet Gap (BBIPG<6:0>)							-000 0000	36
MAIPGL	—	Non-Back-to-Back Inter-Packet Gap Low Byte (MAIPGL<6:0>)							-000 0000	34
MAIPGH	—	Non-Back-to-Back Inter-Packet Gap High Byte (MAIPGH<6:0>)							-000 0000	34
MACLCON1	—	—	—	—	Retransmission Maximum (RETMAX<3:0>)				---- 1111	34
MACLCON2	—	—	Collision Window (COLWIN<5:0>)						--11 0111	34
MAMXFLL	Maximum Frame Length Low Byte (MAMXFL<7:0>)								0000 0000	34
MAMXFLH	Maximum Frame Length High Byte (MAMXFL<15:8>)								0000 0110	34
MICMD	—	—	—	—	—	—	MIISCAN	MIIRD	---- --00	21
MIREGADR	—	—	—	MII Register Address (MIREGADR<4:0>)					---0 0000	19
MIWRL	MII Write Data Low Byte (MIWR<7:0>)								0000 0000	19
MIWRH	MII Write Data High Byte (MIWR<15:8>)								0000 0000	19
MIRDL	MII Read Data Low Byte (MIRD<7:0>)								0000 0000	19
MIRDH	MII Read Data High Byte(MIRD<15:8>)								0000 0000	19
MAADR5	MAC Address Byte 5 (MAADR<15:8>)								0000 0000	34
MAADR6	MAC Address Byte 6 (MAADR<7:0>)								0000 0000	34
MAADR3	MAC Address Byte 3 (MAADR<31:24>), OUI Byte 3								0000 0000	34
MAADR4	MAC Address Byte 4 (MAADR<23:16>)								0000 0000	34
MAADR1	MAC Address Byte 1 (MAADR<47:40>), OUI Byte 1								0000 0000	34
MAADR2	MAC Address Byte 2 (MAADR<39:32>), OUI Byte 2								0000 0000	34
EBSTSD	Built-in Self-Test Fill Seed (EBSTSD<7:0>)								0000 0000	76
EBSTCON	PSV2	PSV1	PSV0	PSEL	TMSEL1	TMSEL0	TME	BISTST	0000 0000	75
EBSTCSL	Built-in Self-Test Checksum Low Byte (EBSTCS<7:0>)								0000 0000	76
EBSTCSH	Built-in Self-Test Checksum High Byte (EBSTCS<15:8>)								0000 0000	76
MISTAT	—	—	—	—	r	NVALID	SCAN	BUSY	---- 0000	21
EREVID(2)	—	—	—	Ethernet Revision ID (EREVID<4:0>)					---q qqqq	22
ECOCON(3)	—	—	—	—	—	COCON2	COCON1	COCON0	---- -100	6
EFLOCON	—	—	—	—	—	FULDPXS	FCEN1	FCEN0	---- -000	56
EPAUSL	Pause Timer Value Low Byte (EPAUS<7:0>)								0000 0000	57
EPAUSH	Pause Timer Value High Byte (EPAUS<15:8>)								0001 0000	57

Legend: x = unknown, u = unchanged, — = unimplemented, q = value depends on condition, r = reserved, do not modify.

Note 1: CLKRDY resets to '0' on Power-on Reset but is unaffected on all other Resets.

2: EREVID is a read-only register.

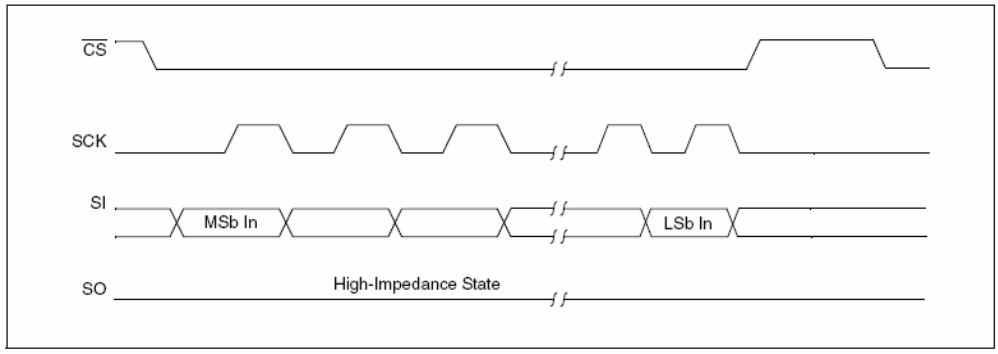
3: ECOCON resets to '---- -100' on Power-on Reset and '---- -uuu' on all other Resets.

5.1.2 Přístup k ENC28J60 SPI rozhraním:

ENC podporuje 8bitový SPI 0,0 slave mód s hodinami v logické nule při klidovém stavu

Vyslaní bajtu po SPI je zahájeno nastavením logické 0 na CS pin ENC (DsPIC PortD8), čímž se uvede ENC strana do stavu očekávání příkazu. Nyní může následovat odeslaní bajtu příkazu/adresy/dat podle tabulky příkazu. Dojde k aktivaci hodin na SPI a vyslání a zároveň přijmutí bajtu. Po skončení vysílání/přijímání vrací se CS pin zpět do logické 1-do klidového stavu

FIGURE 4-1: SPI INPUT TIMING



SPI příkazy:

Funkce ENC28J60 je úplně závislá na pokynech vyslaných z ovládacího zařízení(DsPIC) přes SPI rozhraní. Pokyny jsou vysílány ve formě jedno či více bajtových instrukcí používaných k přístupu k Kontrolní paměti a k Ethernet Bufferu. Instrukce se skládá z nejméně 3bitového operačního kódu(Opcode) následovaného 5ti bitovým argumentem obsahujícím buď adresu registru nebo datovou konstantu. Zapisovací a BitSet instrukce jsou následovány ještě jedno či více bajty dat.

Tabulka instrukcí:

TABLE 4-1: SPI INSTRUCTION SET FOR THE ENC28J60

Instruction Name and Mnemonic	Byte 0								Byte 1 and Following							
	Opcode			Argument					Data							
Read Control Register (RCR)	0	0	0	a	a	a	a	a	N/A							
Read Buffer Memory (RBM)	0	0	1	1	1	0	1	0	N/A							
Write Control Register (WCR)	0	1	0	a	a	a	a	a	d	d	d	d	d	d	d	d
Write Buffer Memory (WBM)	0	1	1	1	1	0	1	0	d	d	d	d	d	d	d	d
Bit Field Set (BFS)	1	0	0	a	a	a	a	a	d	d	d	d	d	d	d	d
Bit Field Clear (BFC)	1	0	1	a	a	a	a	a	d	d	d	d	d	d	d	d
System Reset Command (Soft Reset) (SRC)	1	1	1	1	1	1	1	1	N/A							

Legend: a = control register address, d = data payload.

5.1.3 Zapojení ENC2860:

Oscilátor:

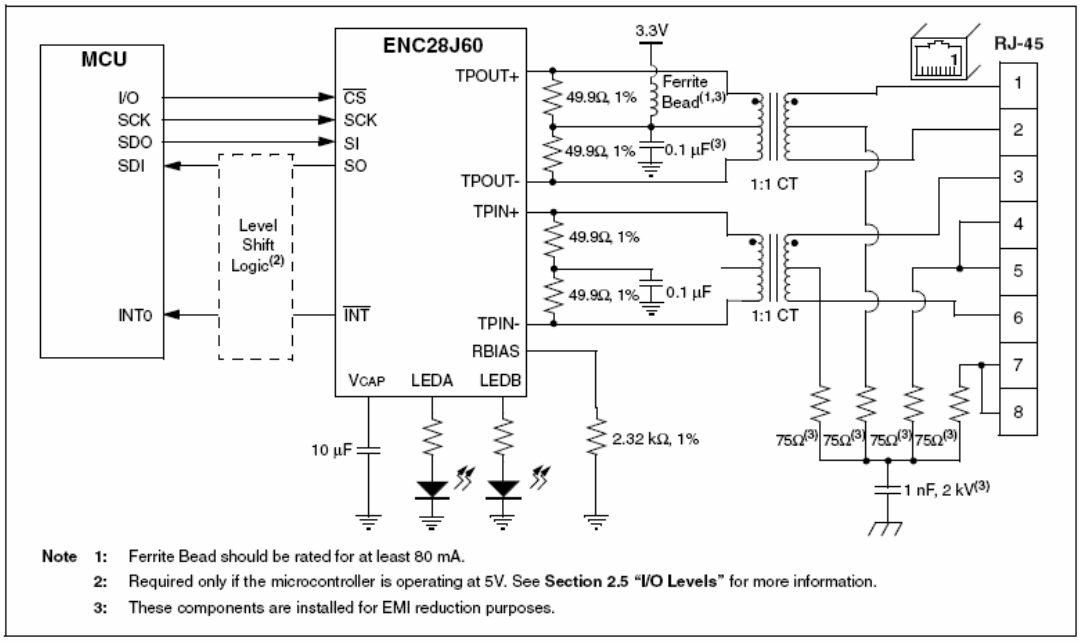
ENC konstruováno na frekvenci 25 MHz. To je možné realizovat připojením buď krystalu s na piny OSC1 a OSC2, nebo použitím externího zdroje hodin a jeho připojením na OSC1.

Po Power-On a Power-down restartu je třeba počkat až se oscilátor stabilizuje. Když se tak stane nastaví se bit CLKRDY v ESTAT (Kontrolní registr v ENC) do log1.

Vnější spoje:

Aby ENC správně fungovalo je třeba ho zapojit podle katalogových listů:

FIGURE 2-4: ENC28J60 ETHERNET TERMINATION AND EXTERNAL CONNECTIONS



Note 1: Ferrite Bead should be rated for at least 80 mA.
2: Required only if the microcontroller is operating at 5V. See **Section 2.5 "I/O Levels"** for more information.
3: These components are installed for EMI reduction purposes.

Modul fyzické vrstvy potřebuje rezistor 2,32KΩ proti zemi na pin RBIAS. Tento Odpor ovlivňuje aplitudu výstupního signálu na pinech TPOUT+/-. Doporučuje se, aby to byl typ pro povrchovou montáž a kvůli rušení byl připojen, co nejkratším spojem na zem.

ENC je napájeno 3,3V, kdežto DsPIC použitý v tomto projektu jako ovládací kontroler je napájen 5V. Napěťový rozdíl je třeba kompenzovat. Vstupy ENC(CS, SCK, SI, RESET) jsou 5V tolerantní, ale výstupy(SO, INT, CLKOUT) nejsou v TTL úrovních, přestože by pravděpodobně DsPIC rozpoznal úrovně správně(3,3 je stále ještě v toleranci) je pro jistotu třeba použít úrovňový zdvih, který je možný realizovat např. pomocí logického integrovaného CMOS obvodu AND (74HCT08), jehož jeden vstup připojíme do log1. viz obr. 2-5.

K ENC je také možné připojit 2 diody, které můžeme mimo jiné nastavit na indikaci Link(kabel připojen k jinému zařízení) a Recieve(příchozí paket) stavu. Zapojením LED diody B je možné hardwarově nastavit Full/Half duplex přístup k médiu. Viz obr. 2-7

FIGURE 2-5: LEVEL SHIFTING USING AND GATES

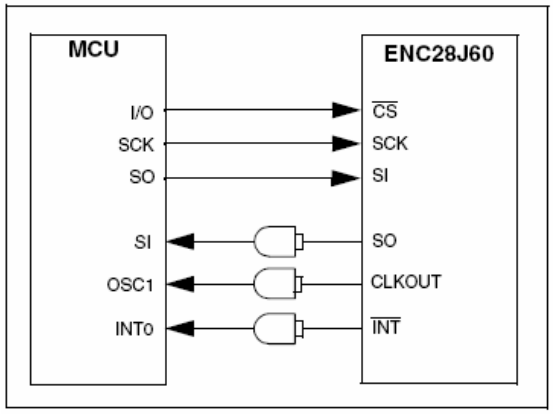
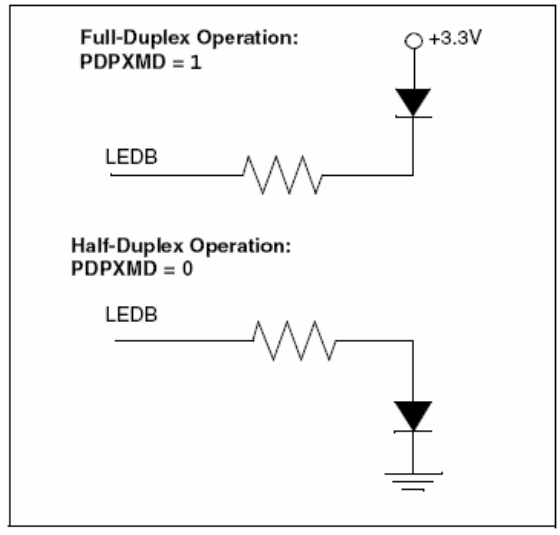


FIGURE 2-7: LEDB POLARITY AND RESET CONFIGURATION OPTIONS



5.1.4 Kalkulace kontrolního součtu v ENC28J60:

ENC disponuje funkcí pro kalkulaci kontrolního součtu (checksum). Dvěma pointery (rozdeluje se na spod. a hor. bajt, EDMAST-pointer na první bajt dat pro checksum, EDMAND-pointer na poslední bajt zahrnuty do checksum) definujeme oblast v cyklickém Ethernet Bufferu, pro kterou má ENC spočítat kontrolní součet. Pak aktivujeme startovacími bity operaci(ECON1.CSUMEN;ECON1.DMAST). Když výpočet skončí nastavi se EIR.DMAIF, který může generovat přerušení. Výsledek je v Kontrolních registrech EDMACS (H a L-horní a spodní bajt).

Kontrolní součet pracuje s označeným obsahem Ethernet Bufferu jako s jednotlivými 16ti bitovými čísly(pokud jde o lichý počet přidá se vycpávka 00h, na doplnění do 16bitů). Tyto 16bit. čísla sečte (Carry bit se u každého jednotlivého součtu automaticky sečte s daným výsledkem) a doplněk celkového výsledku do max. hodnoty(0xFFFFh) je výsledný 16bit. kontrolní součet.

Příklad:

hodnoty v paměti pro kon.souč.:{89h, ABh, CDh}

součet: 89ABh + CD00h = 156ABh

přesun carry bitu: 56ABh + 0001h = 56ACh

doplněk: FFFFh – 56ACh = A953h

kontrolní součet tedy je A953h

Této metody se užívá u protokolů IP, TCP, UDP, ICMP atd.

Nastavení stránky (SetPage podprogram)

Číslo aktuální stránky uvádí dva nejspodnější bity(BSEL0,1) ECON1 kontrolního registru.

REGISTER 3-1: ECON1: ETHERNET CONTROL REGISTER 1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TXRST	RXRST	DMAST	CSUMEN	TXRTS	RXEN	BSEL1	BSEL0
bit 7							bit 0

Změna stránky se musí provést pomocí BitSet(ne write2control) instrukcí, poněvadž změna stránky nesmí ovlivnit ostatní bity v registru!

5.1.5 Zapsání do PHY registru (Write2Physical podprogram)

Do PHY registrů se nedá zapisovat přímo nějakou SPI instrukcí. Musí se tak učinit zapsáním do specielních Kontrolních registrů. Uvádím postup zapisovaní:

- zápis adresy PHY registru do Kontrolního registru MIREGADR
- zápis spodních 8 bitů dat do Kontrolního reg. MIWRL
- zápis horních 8 bitů dat do Kontrolního reg. MIWRH
- počkat až bit MISTAT.BUSY spadne zpět do 0, nebo vyčkat přesně 10,24us, nebo pokračovat dál, ale nezapisovat a nečíst z PHY paměti po danou dobu.

5.2 Inicializace zařízení

(viz 6.0 INITIALIZATION)

Před správnou funkcí zařízení je ho nejprve třeba nastavit. To provedeme zapsáním série hodnot do některých Kontrolních registrů. Pokud se jedná o registry rozdělené na horních a spodních 8 bitů, je nutno vždy nejprve zapsat do hodnotu do spodních 8mi. Doporučuje se dodržovat daný postup.

Rozdělení Ethernet Bufferu na přijímací a vysílací část

Provedeme zapsáním počáteční a koncové adresy přijímací části. Buffer má 8K. Nejvyšší možná adresa je 1FFFh. Pokud budeme chtít rozdělit paměť přesně na polovinu nastavíme Přijímací(Recieve) část uložením

0FFEh do ERXST(16bit reg. ukazuje na začátek přij.části, musí být sudá adresa)

1FFFh do ERXND(16bit reg. ukazuje na konec přij.části)

Veškerá ostatní paměť je považována za vysílací část.

ERXRDPT je nutno nastavit na stejnou hodnotu jako ERXST

Nastavení Přijímacích filtrů (viz Recieve Filters)

Přijímací filtry se nastavují registrem ERXFCON. Sérií povolovacích bitů. Pro většinu aplikací má smysl uvažovat o nastavení UCEN(Unicast), CRCEN(CRC ok), BCEN(Broadcast), ANDOR(logika procházení filtrem). Pro běžné aplikace je vhodné ponechat defaultní hodnotu. (UCEN,CRCEN,BCEN=1; zbytek 0)

REGISTER 8-1: ERXFCON: ETHERNET RECEIVE FILTER CONTROL REGISTER

R/W-1	R/W-0	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1
UCEN	ANDOR	CRCEN	PMEN	MPEN	HTEN	MCEN	BCEN
bit 7							bit 0

- UCEN – propustí rámce s MAC adresou shodou s adresou uloženou v Kontrolních registrech MAADR1-6. Ty se nastavují až později.
- CRCEN – propustí rámce u kterých nedetekuje chybu použitím CRC32, nevztahuje se na něj ANDOR bit
- BCEN – propustí všechny broadcast rámce(MAC= ff:ff:ff : ff:ff:ff). Takové rámce se používají např. u ARP protokolu, tudíž jsou nezbytné pro analýzu a odpověď.

- ANDOR (viz obr 8-1 Recieve Filtering using OR logic)
    - Log 1- rámec musí projít všemi filtry
    - Log 0- rámec musí projít alespoň jedním filtrem

Ať je hodnota v ANDOR jakákoli musí paket zároveň splnit CRCEN filtr, pokud je aktivován.

Paket, který neprojde definovanými filtry bude odmítnut. Uživatel odmítnutý rámec nebude schopný nijak zaregistrovat.

Zapsáním 00h do ERXCON přepne zařízení do promiskuitního mód (všechny příchozí pakety budou přijmuty)

Čekání na Oscillator Start-up Timer (viz Wainting For OST)

Než bude možno pokračovat v inicializaci, je třeba počkat až se stabilizuje oscilátor vždy po Power-on Resetu. Indikátor stabilizace je bit ESTAT.CLKRDY. Jakmile se nastaví do 1 můžeme pokračovat. (Nyní je možno nastavovat MAC a PHY registry)

MAC inicializace

Následuje sled nastavení, který je pochopitelný jen po hlubším porozumění problematiky. Raději se řídím originální katalogovým listem- jeho doporučeným nastavení.

Do některých registrů je nutno zapsat jiné hodnoty pro Plno a pro Polo-Duplexní(Half/Full-Duplex) typ mnohonásobného přístupu k médiu. Uvádím nastavení, které bude v obou případech nezávislé na vnějším zapojením LED diody B, kterou je možné způsobem zapojením typ spojení hardwarově nastavit. (Viz Vnější spoje; LED polarity)

Ve všech případech následujících zápisů se jedná o registry na stránce 2.

Full Duplex	**Half Duplex**
0Dh do MACCON1	01h do MACCON1
B3h do MACCON3	B2h do MACCON3
40h do MACCON4	40h do MACCON4
EEh do MAMFLL	EEh do MAMFLL
05h do MAMFLH	05h do MAMFLH
15h do MABBIPG	12h do MABBIPG
12h do MAIPGL	12h do MAIPGL
0100h do PHY.PHCON1	0C do MAIPGH
	0000h do PHY.PHCON1
	0100h do PHY.PHCON2

Nastavení MAC adresy:

Pro správnou funkci Unicast filtru je nutné do ENC uložit vlastní MAC adresu, ta není dodávána s IO. Unikátní MAC adresa se dá například převzít od starých nebo rozbitých síťových karet a které je pak možno zlikvidovat, abychom se vyhnuli problémům s kolizemi dvou totožných MAC v síti. MAC adresa se zapisuje do kont. reg. na stránce 3 MAADR1-6, přičemž do MAADR6 se zapisuje první bajt(LSB).

Přerušení při přijímání paketů:

ENC může generovat přerušení při každém příchozím paketu, který projde Filtry, pokud nastavíme povolovací bity EIE.PKTIE a EIE.INTIE nebo pokud chceme, aby generoval přerušení při každém zahozeném paketu z důvodu přeplněnosti Bufferu, je třeba nulovat EIR.RXERIF a nastavit bity EIE.RXERIE a EIE.INTIE.

Povolení přijímání paketů:

Po nastavení RXEN bitu neměl by se měnit Duplexní mód, pointer označujicí začátek a konec přijímacího bufferu a pro předcházení přijmutí nevhodných paketů je doporučeno před změnou filtrů ERXFCON a MAC adresy dočasně nuloval RXEN. Jakmile bude přijímání povoleno, všechny pakety vyhovující filtrům budou zapsány do paměti přijímacího bufferu. Odmítnuté budou odmítnuty a uživatel nebude mít možnost to zjistit.

Pro povolení přijímání
Je nutno nastavit ECON1, RXEN

5.3 Vyslání paketu

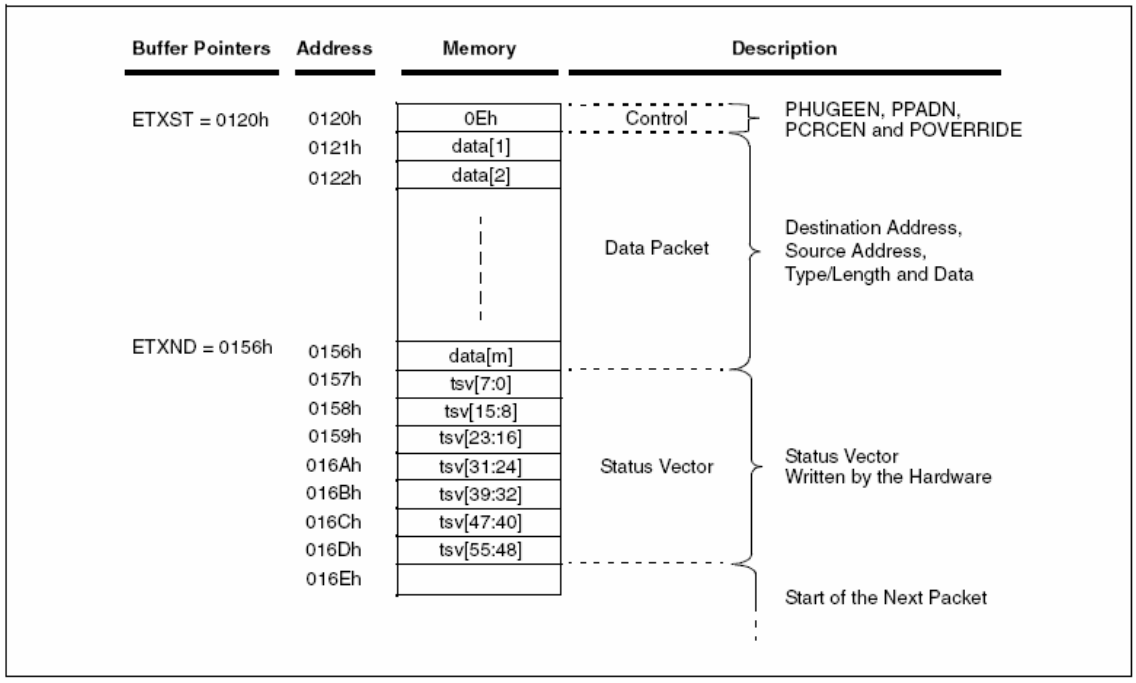
Pokud je ENC správně inicializován generuje automaticky Preambuli, Start-of-frame, vycpávku(60B nebo 64B) a CRC32. Toto nastavení se dá měnit i jen pro jednotlivý paket použitím Per-Packet-Control bajtu(viz. 7.1 datasheet). Uvádím doporučený katalogový postup pro odeslání v paketu:

1. Nastavit ETXST pointer na sudou nepoužívanou(Ne-přijímací) část paměti, tam kde bude paket začínat. *Pokud program nepotřebuje chrlit hodně dat najednou a nemůže čekat až se pakety odešlou, postačí nastavit tento pointer vždy na stejnou adresu.*
2. Zapsat do bufferu pod hodnotu v ETXST, Per-Packet-Control bajt (pro většinu aplikaci 00h)
3. Zapsat DATA (mac cílov. adresa,mac zdrojova adresa, type/lenght, data rámce)
4. Nastavit ETXND pointer na poslední bajt dat zahrnutý do paketu
5. Nulovat bit EIR.TXIF, nastavit EIE.INTIE pro povolení přerušení
6. Vyslat paket nastavením ECON1.TXRTS

Paket bude odeslán jakmile to bude možné- jakmile skončí případná DMA operace (např. Checksum kalkulace). Stejně tak DMA operace by počkala pokud by se vysílal paket.

Jakmile bude paket odeslán nebo pokud bude jeho vysílání zrušeno TXRTS bude vynulován a sedmibajtový *Status Vektor*(viz. kapitola) bude zapsán za vysílaný paket. Pokud bude ESTAT.TXARB v 1, paket se neodeslal správně. ESTAT.LATECOL určuje, že kolize nastala po odvysílání 64bajtů. (tzv. pozdní kolize, viz. Half-duplex problematika, ale tento problém by neměl být příliš častý a ve většině případů není třeba ho ošetřovat)

FIGURE 7-2: SAMPLE TRANSMIT PACKET LAYOUT



5.4 Přijímání paketů

Jakmile bude přijímání v inicializaci povoleno, všechny pakety vyhovující filtrům budou zapsány do paměti přijímacího bufferu. Odmítnuté budou smazány a uživatel nebude mít možnost to nijakým způsobem zjistit.
Jakmile je paket kompletně přijmut a zapsán v bufferu registr EPKCNT inkrementuje, EIR.PKTIF (určuje, že v paměti je jeden nebo více nezpracovaných paketů) se nastaví do 1 a bude vyvoláno přerušení, pokud je povoleno a Hardwarový Write Pointer (ukazuje na adresu kam bude zapisován následující paket; softwarově je nezapisovatelný) automaticky inkrementuje o 1. EIR.PKTIF nuluje hardware po uvolnění místa paketu *viz. kapitola Uvolňování místa v přijímací paměti*.

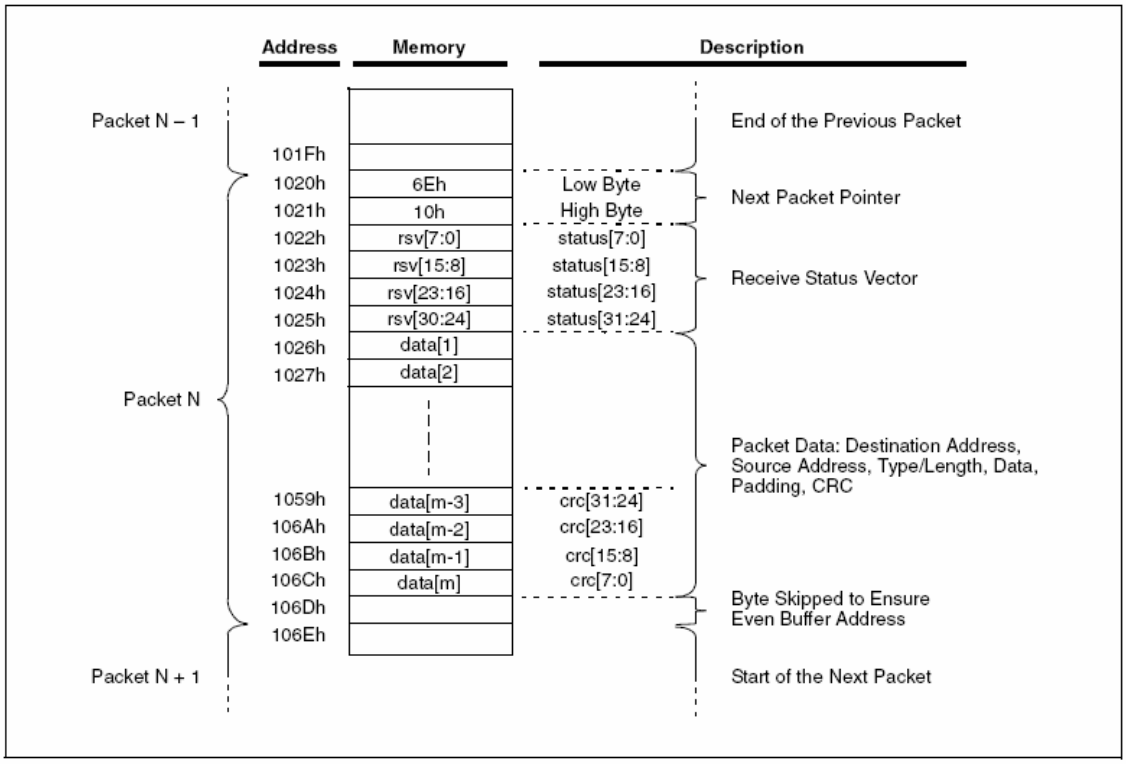
Přijmutý paket (viz. obr 7-3) zapsaný v bufferu má předřazených 6 informačních bajtů.

1. První dva určují tzv. Next Paket Pointer. (tj. adresa prvního bajtu následujícího paketu) Hodnota Next Paket Pointer je velice důležitá pro běh celého přijímacího programu - doporučuji ji zálohovat do řídícího procesoru (DsPIC).

2. Následuje 4 bajtový Status vektor (viz kapitola)
3. Za 6ti informačními bity jsou v paměti data rámce(cílová mac. ad., zdrojová mac. ad., type/length, data, pading a CRC32 rámce)

Na konci ještě může být přidán vycpávkový bit tak, aby první adresa následujícího paketu(next paket pointer) byla sudá.

FIGURE 7-3: SAMPLE RECEIVE PACKET LAYOUT



5.4.1 Uvolňování místa v přijímací paměti

Uvolňování cyklické paměti(buffer)

Poté, co uživatel zpracuje paket (nebo jeho část) a chce uvolnit místo zaplněné zpracovanými daty, musí zvýšit hodnotu Recieve Buffer Read Pointeru (ERXRDPT). ENC bude zapisovat cyklicky do Bufferu pokud hodnota Hardware Write Pointeru bude jiná než hodnota ERXRDPT, tudíž všechna data pod a na adrese v bufferu na kterou ukazuje ERXRDPT pointer jsou chráněna proti přepsání daty novými. Když se ENC pokusí přepsat buňku chráněnou ERXRDPT, zápis se zruší, hodnota v buňce zůstane a pokud je povoleno přerušení vyvolá se. (nastaví se EIR.RXERIF, přerušení se povoluje EIE.RXERIE). Všechny příchozí pakety budou do uvolnění paměti zrušeny.

Změna pointeru bude provedena po zapsaní nejprve spodního a teprve pak horního bajtu.Pokud uživatel zpracuje veškerá data v určitém paketu a chce uvolnit jeho místo, musí zvýšit hodnotu ERXRDPT, tak aby ukazoval za zpracovaný paket (pozor paměť je cyklická, nejlepší nastavit hodnotu z Next Paket Pointeru), pak je nutno nastavit bit ECON2.PKTDEC. Když toto uživatel provede hodnota v EPKTCNT (registr obsahující hodnotu nezpracovaných paketů) dekrementuje o 1. Pokud EPKTCNT dosáhne nuly hardware vynuluje bit EIR.PKIF (určuje, že v paměti je jeden, nebo více nezpracovaných paketů).

Určení velikosti volného místa v cyklické paměti

Odečtením hodnot registrů ERXRDPT a ERXWRPT(Hardware Write Pointer) získá uživatel počet volných bajtů, připravených k zapsaní nových paketů.

6 Program v dsPIC30F3013

Tato část shrnuje nejdůležitější funkce programu použitého v projektu Médea.

6.1 Základní podprogramy:

Pro z zjednodušení, urychlení a zpřehlednění programu jsme pro nejčastěji používané skupiny instrukcí vytvořily podprogramy. Některé z nich mají vstupní parametry a některé také vrací hodnotu. Pro zadávání vstupní a vracení výstupních hodnot používáme wregistry, kterých je v našem DsPICovi použitelných celkem 15 (0-14). Pro vstupní/výstupní hodnoty používáme povětšinou W registry 1-3, zbytek je určen pro pracovní výpočty.

Podprogramy SPISendW1, SPISendW2:

Jsou v hierarchii nejníže položené podprogramy určené pro přímou komunikaci s SPI rozhraním. Vyšlou spodních 8bitů z W1 nebo W2 a zároveň přijmou 8bitů vyslaných z cílového zařízení (ENC2860).

Podprogramy přímo užívající SPISend, nastavující CS pin

Podprogram Write2Control:

Tento podprogram je určený pro ukládání hodnoty ve W2 do Kontrolního registru v ENC na adrese uložené ve W1 na současné stránce. Vysílá kompletní příkaz write control register. Může být předcházen prodprog. SetPage pro nastavení stránky.

Podprogram ReadControl:

Je určen ke čtení z Kontrolních registru. Velmi se podobá Write2Control. Má variantu pro čtení Kontrolních reg. typu MAC

Podprogram BitSet/Clear

Nastavují vybranou hodnotu maskováním hodnotou ve W2 Kontrolního registru na adrese ve W1.

Podprogramy Write2Buffer/ReadBuffer

Jsou určeny pro čtení a zápis z Ethernet Bufferu. Vysílají kompletní příkazy write/read buffer mem. Vysílají/Vrací hodnoty ve W1. Může ho přecházet podprogram SetWrite/ReadPointer nastavující pointer na vybranou hodnotu.

Podprogram SoftReset

Tento podprogram vyvolá „měkký“ reset ENC vysláním příkazu po SPI.

Podprogramy přímo užívající některé z výše uvedených podprogramů

Podprogramy SetPage, Write2Physical, SetWrite/ReadPointer, atd…

6.2 Start programu

Start programu probíhá ve 3 fázích:

- Nastavení procesoru DsPIC (Porty, SPI, Uart, ..)
- Reset ENC28J60 (Hardware reset, detekce stavu přepínače Half/Full Duplex)
- Inicializace ENC28J60 (Načtení IP z EEPROM, Inicializace, spuštění Ethernet rozhraní)

Tím se uvede Médea modul do stavu očekávání, je to pasivní zařízení (nikdy nezačne vysílat jako první). Program v DsPIC je uveden do cyklu očekávající změnu, buď stisknutí některého s tlačítek a nebo přijetí paketu.

6.3 Běh programu

Po startu se program uvede do stavu očekávání (jde o pasivní zařízení) buď příchozího paketu změny stavu na jumperu Half/Full Duplex, změny stavu tlačítka Reset IP.

- Po příchodu paketu odskočí program do podprogramu a paket zpracovává, případně poté odešle vhodnou odpověď. (více v kapitole Identifikace příchozího paketu)
- Jakmile je detekována změna stavu na jumperu Half/Full Duplex program se resetuje. Přičemž může dojít ke ztrátě některých příchozích paketů, které právě mohou být přijímány obvodem ENC28J60, proto je nutné provádět změnu duplexu jen v klidovém stavu zařízení, nejlépe při úplném vypnutí.

- Stiskem tlačítka Reset IP způsobíme načtení základního nastavení IP adresy (192.168.1.5) Jestliže mají být ještě přijmuty některé pakety na původní IP, proveďte Reset IP až po jejich zpracování.

6.4 Half/Full Duplex jumper

ENC28J60 nedisponuje funkcí autonegotiation (automatické nastavení duplexního provozu používající synchronizační impulsy) a tedy je duplexní provoz nastavit ručně pomocí jumperu přičemž máme dvě možnosti:

- Half-Duplex – zařízení je připojeno k HUBu, nebo k routeru nepodporujícím Full duplex
    - jumper je zapojen uzavřen
- Full-Duplex – zařízení je připojeno ke Switchi nebo k routeru podporujícím Full duplex
    - jumper není zapojen

Duplexní změnu provádějte jen v klidovém stavu zařízení.

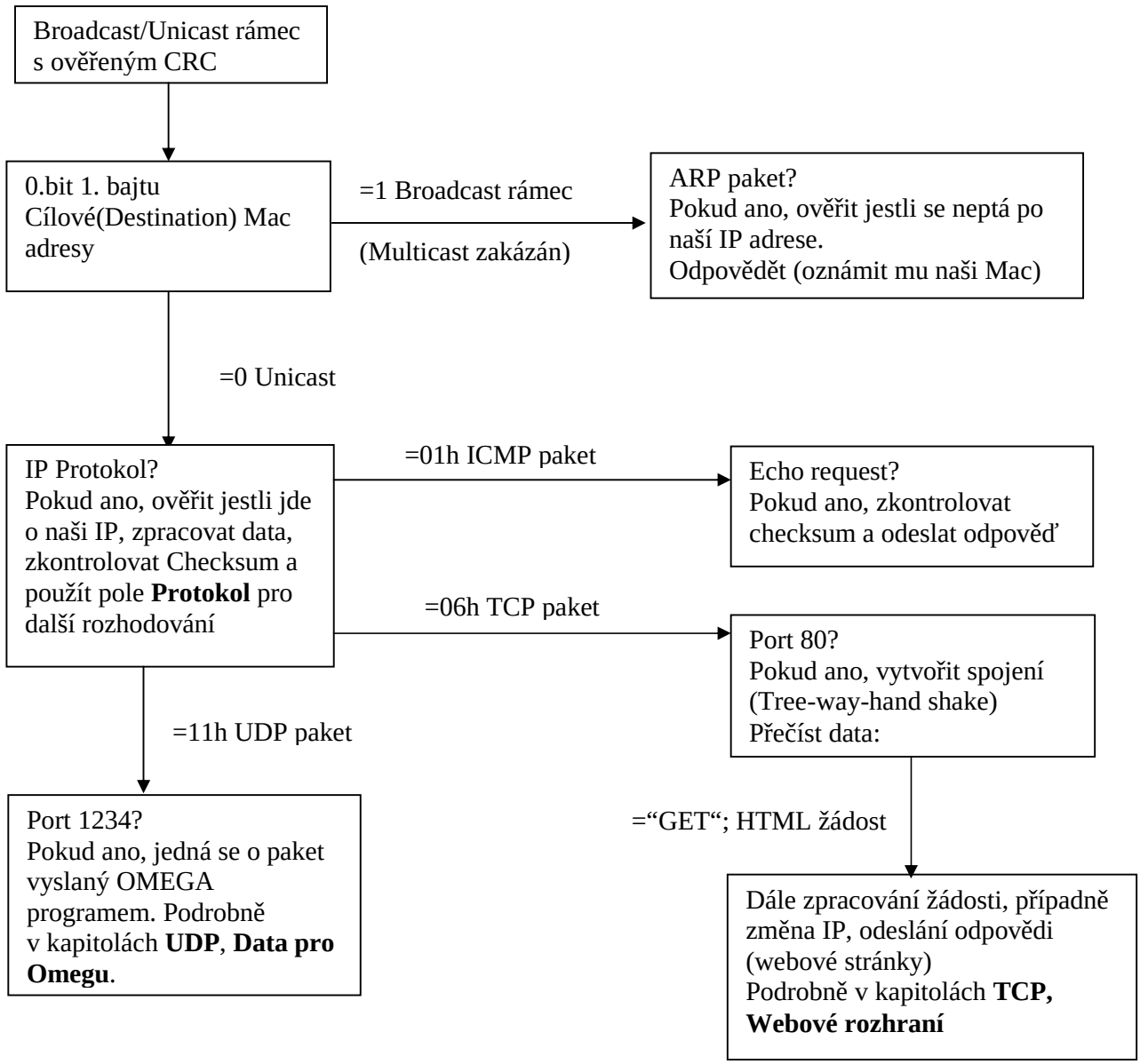
6.5 Reset IP

Za normálních okolností je při každém startu programu načtena z paměti EEPROM poslední nastavená IP adresa, jejíž změna se provádí pomocí webového rozhraní (viz. kapitola Webové rozhraní) Pokud však uživatel z nějakého důvodu tuto adresu nezná a tudíž ani nemá přístup k webové stránce, musí provést Reset IP, čímž se adresa zpět nastaví na základní 192.168.10.111 a poté ji může změnit na požadovanou hodnotou.

6.6 Identifikace příchozího paketu

(průchod vrstvami referenčního modelu OSI)

Jakmile je přijat paket Médea modul ho začne zpracovávat. Nejprve je třeba zjistit o jaký druh paketu se jedná – začíná proces třídění, proces čtení dat, které paket buď vyřadí jako nežádoucí/nepodporovaný a nebo ho identifikuje jako žádost o určitou službu a dále ho zpracuje a odešle odpověď.



6.7 Přerušení

Používáme přerušení UARTu jak pro vysílání, tak pro příjem, umožní to zrychlení programu.

6.7.1 Inicializace UARTu

Inicializace UARTu se provádí podprogramem _UartInit. Tedy **call _UartInit.**

- Zápis do U1BRG. Rychlost 57600, počítá se s osminásobným fázovým závěsem.
- Zápis do U1MODE číslo 0x8400, tzn. 8 bitu data, parita zadna, 1 stop bit
- Aktivace UARTu => v registru U1STA nastavení bitu UTXEN do 1. Ostatní bity registru zůstávají v nule.

6.7.2 Přerušení .global prijem

Uložení dat do paměti se děje v přerušení .global příjem

- Pro UDP: Přijetí a uložení do paměti
- Pro TCP: Odříznutí hlavičky EE 23, odříznutí CRC

6.7.3 Přerušení .global vysilani

Vysílání dat z paměti do ústředny se děje v přerušení .global vysilani

- Pro UDP: Přeposlání UDP paketu do ústředny
- Pro TCP: Přidání hlavičky, zdvojování znaku EE za znakem START = 23, počítání CRC

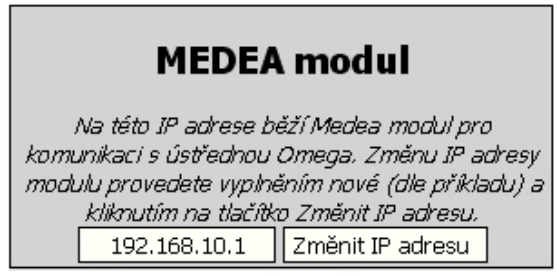
6.7.4 Počítání CRC32

Pro TCP pakety je nutno počítat CRC. Podprogram pro CRC se volá příkazem call CRC32. Postup pro užití podprogramu je takový:

- Vložit adresu začátku dat do registru w13 a délku dat do buňky data_length
- Načíst data z paměti, vložit do W0 a zavolat podprogram call crc32_update
- Po posledním bajtu dat pokračuje program na crc_end a následně dostáváme výsledek v buňkách crc32_H a crc32_L

Pozn.: Vznikly dva programy, jeden počítá CRC podle způsobu uvedeného v 4.3.3, tento způsob však při implementaci na data z ústředny, nevycházel. Vznikl tedy program druhý, a to přepisem programu pro počítaní CRC32 z jazyka C. Program vytvořil pan profesor Tomáš Kubalík, vedoucí naší práce. Bohužel jsme nepřišli na to, jak a proč první způsob nefunguje.

7 Webové rozhraní



autoři: Košař, Kapic

7.1 Změna IP adresy

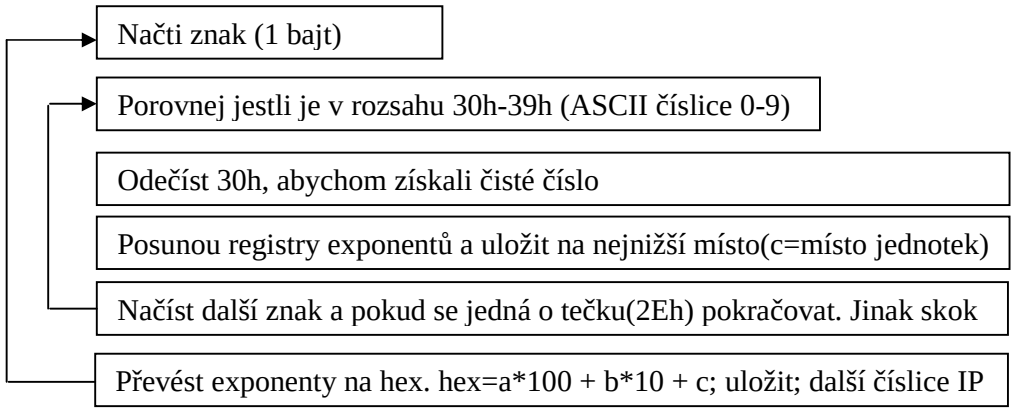
Změna IP adresy se provádí pomocí webového rozhraní – pomocí libovolného webového prohlížeče. Nejprve se mezi prohlížečem a Médea modulem vytvoří TCP/IP spojení na portu 80 (http protokol). Jakmile obdrží Médea modul požadavek na určitou akci pomocí parametrů http příkazu GET odešle vhodnou html odpověď a případně změní IP adresu v paměti DsPIC-EEPROM.

Princip:

- Data TCP segmentu obsahují ASCII text GET / http
    - Médea modul odešle zpět standardní webovou stránku, k vyplnění pole pro změnu IP adresy použije uložený ascii podobu
- Data TCP segmentu obsahují ASCII text GET /?IP= za kterým následuje ***správně*** zadaná IP v ASCII podobě
    - načte se ascii podoba IP
    - převede se na čtyři hexadecimální čísla
    - obě varianty vyjádření IP se uloží do EEPROM
    - odešle se odpověď „IP adresa byla úspěšně uložena“
    - provede se změna IP adresy v RAM
    - program dále reaguje jen na novou IP adresu
- Data TCP segmentu obsahují ASCII text GET /?IP= za kterým následuje ***nesprávně*** zadaná IP v ASCII podobě
    - Médea modul odešle standardní webovou stránku s upozorněním „Chybná IP!“

Převod ASCII IP -> HEX IP

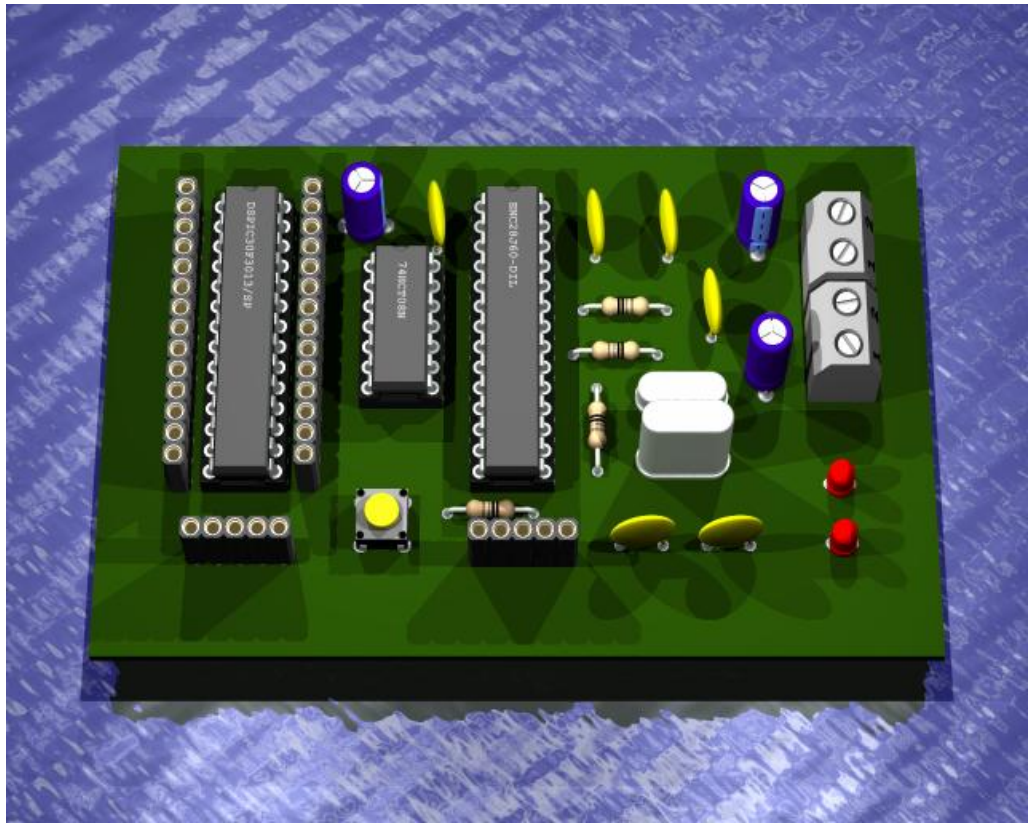
Běžně se IP adresa zapisuje v decimálním tvaru, přičemž jednotlivé číslice jsou odděleny tečkou. Díky vynechávání nul má nepevnou délku. Tak tomu je i když je IP adresa vyjádřena v ASCII znacích. V této podobě je snadno čitelná a zapamatovatelná pro člověka. Avšak v paketech, které analyzujeme a vysíláme, se vyjadřuje IP adresa čtyřmi hexadecimálními, ničím neoddělenými číslicemi.



8 Tvorba plošného spoje

Celkem vznikly desky dvě. První byla zaměřena čistě na tréninkové účely => Training board. A druhá je již připravena na použití v ústředně.

8.1 Training board



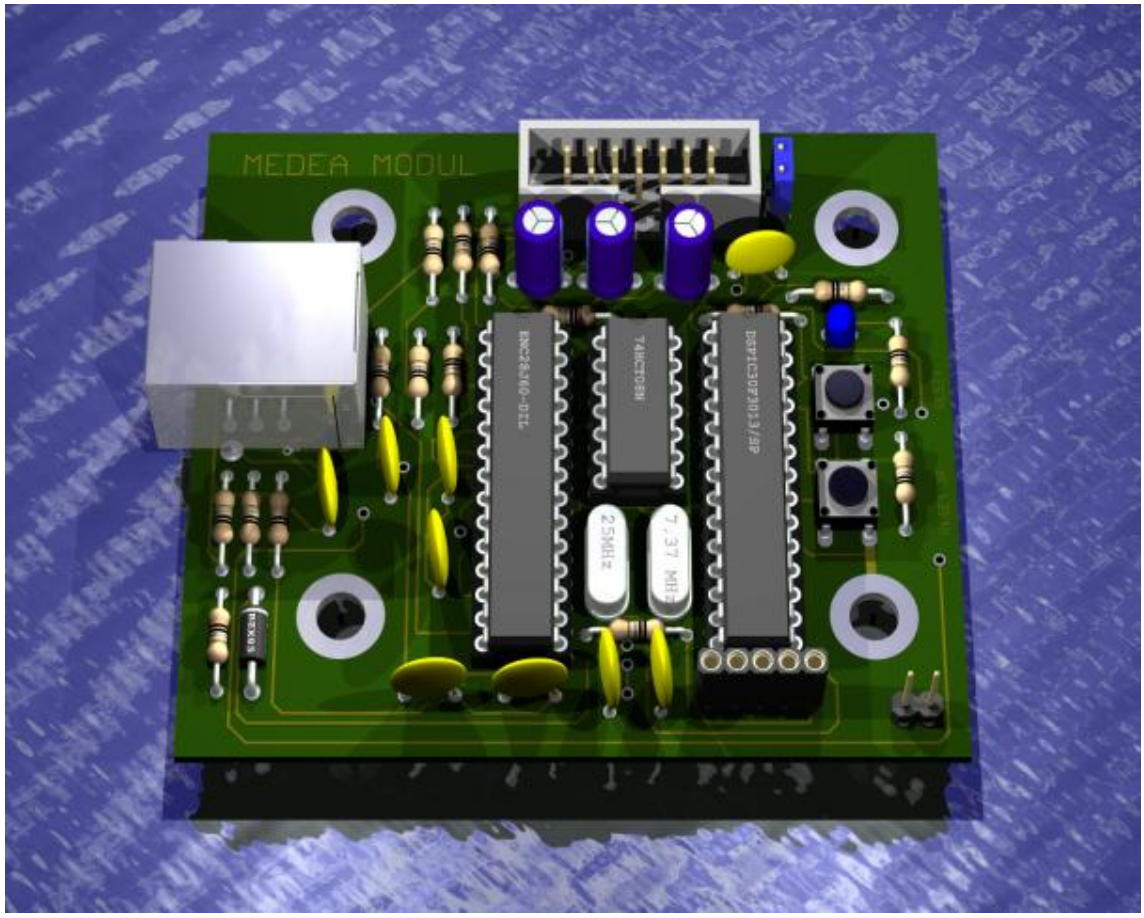
Rozmístění součástek na desce v 3D

Na této desce jsme si ověřovali komunikaci SPI mezi PICem a ENC, dále UART a všechno potřebné k realizaci naší desky v sítích TCP/IP. Jsou zde zavedeny lámací lišty pro obvyklé „kde může být chyba“. Prostřednictvím lámací lišty se také programuje. Pod PICem (nalevo) jsou vývody pro připojení Tomášova převodníku, pod ENC vývody pro připojení trafa a konektoru RJ-45. Vývody napájení jsou zde řešeny pomocí svorkovnic, jedna na 5V a druhá na 3,3V (ENC vyžaduje), země jsou spojeny. Tlačítko resetuje ENC. Jsou použity krystaly 25 MHz (ENC) a 7,3728 MHz (dsPIC). Dále CMOS hradlo 74HCT08 na převod úrovní mezi PICem a ENC.

8.2 Medea modul verze 1.1

Druhý z výrobků jsme nazvali Medea modul. Výrobek je již připraven k použití v ústředně, pro upevnění je vybaven montážními otvory. Modul napájí ústředna, z které leze 5V, napájení 3,3V je vyřešeno Zenerovou diodou (zapojení níže). Lišta pod PICem dole je určena na programovaní. Deska má dvě tlačítka, tlačítko umístěné nejvýše je RESET (reset PICa – vede na MCLR) a tlačítko umístěné pod ním RESET IP (nastavuje defaultní IP adresu modulu 192.168.10.111). Pro lepší orientaci během přenosu informací jsou tu 3 diody, dvě posazené ve sloupku nad sebou vedle konektoru RJ-45 a jedna nad tlačítky. Modul komunikuje s ústřednou prostřednictvím 14-pinového konektoru WSL 14G (úplně nahoře-popis níže). Posledním důležitým prvkem je jumper (vedle WSL 14G), nastavuje half-duplex a full-duplex.

Zapojení jumperu: **HALF = JUMPER CLOSED**, **FULL = WITHOUT JUMPER**



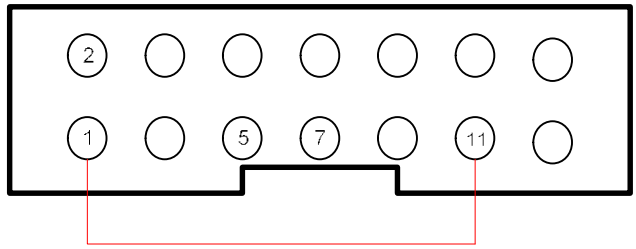
Medea modul verze 1.1

Při realizaci desky došlo k chybě. **Byly prohozeny piny RX a TX** u 14-pinového konektoru. Chyba je opravena redukcí, která je součástí modulu. Popis konektoru viz. níže.

Hned po odhalení chyby vznikl další návrh desky, verze 1.2, kde je vše opraveno.

8.3 Popis WSL 14G

Modul je spojen s ústřednou pomocí 14-žilového kabelu. Na obrázku dole je vidět konektor na našem modulu z pohledu součástek.



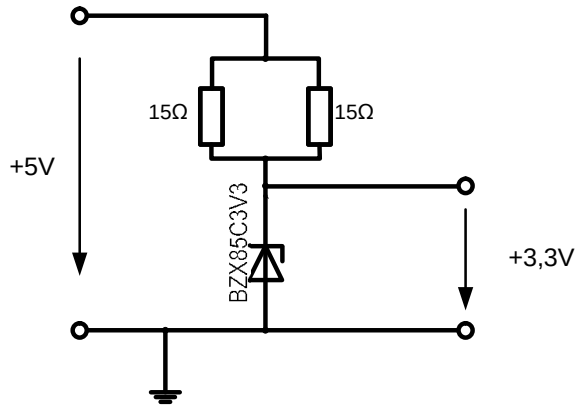
Náhled na WSL 14G zasazeného na desce z pohledu součástky

PIN	POPIS
1	GND
2	+5V
5	RX dsPIC, RX ústředna
7	TX dsPIC, TX ústředna
11	GND

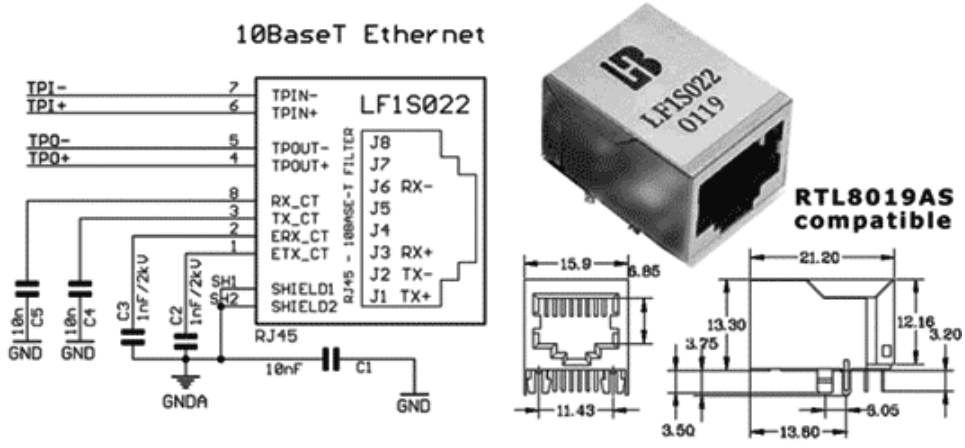
Pozn. :

1) Pin 1, tedy GND, je spojen s pinem 11.
2) V tabulce je vidět, že nedošlo ke křížení cest, jak mělo (tzn. TX dsPIC, RX ústředna). Redukce to řeší prohozením drátů. Tzn. RX na TX.

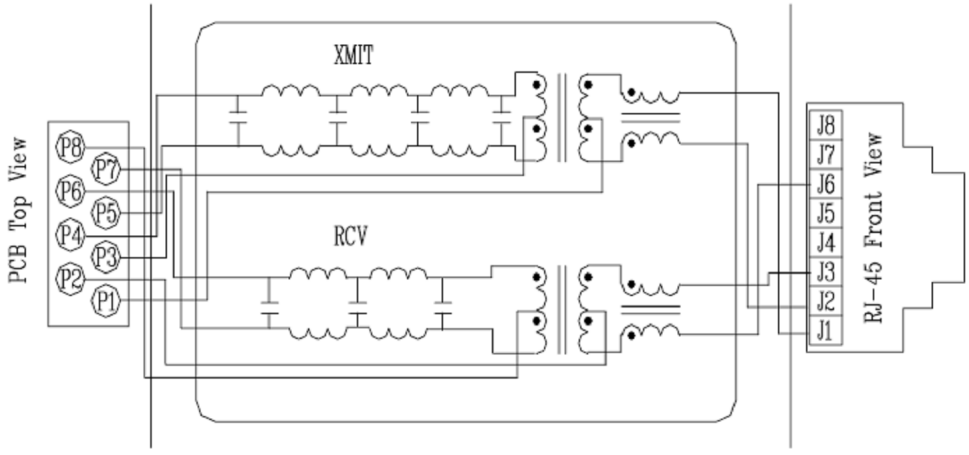
8.4 Napájení 5V na 3,3V



8.5 LF1S022



Při výběru trafa a konektoru, jsme se rozhodli pro integritu a zvolili LF1S022, oddělovací signálové trafo pro ethernet 10Base-T v konektoru RJ-45.



Vnitřní schéma zapojení

9 Závěr

Výsledkem naší dlouhodobé maturitní práce je plně funkční hardwarový modul do ústředny Ateus Omega. Zkonstruováním tohoto výrobku jsme se zdokonalili v řadě dovedností, mezi které patří: programováni v Assembleru, zhotovování plošných spojů, monitorování a aplikování paketových komunikačních protokolů v praxi, spolupráce v týmu. Naučili jsme se řešit problémy spojené s realizací produktu. Do budoucna by možným vylepšením mohlo být snížení energetických ztrát stabilizátoru napětí. Na závěr bychom chtěli poděkovat panu profesorovi Ing. T. Kubalíkovi za jeho podporu a pomoc v dobách, kdy se situace zdála neřešitelná.

10 Odkazy a citace:

Literatura:

Šmrha, P. – Rudolf, V.: Internetworking pomocí TCP/IP vydání České Budějovice, Kopp 1997, 126 s.

Katalogové listy-pdf:

obvody: DSP30F3013, ENC28J60 Microchip

LF1S022

Sériová komunikace s ústřednou Ateus Omega 2N

Odkazy Internet:

http://cs.wikipedia.org/wiki/TCP/IP TCP/IP protokoly

http://www.cs.vsb.cz/grygarek/LAN/sem/sercomm.html sériový přenos

http://cs.wikipedia.org/wiki/Cyklick%C3%BD_redundantn%C3%ADsou%C4%8Det CRC

11 Použitý software

MPLAB IDE v. 7.42

ASIX UP v. 2.7

EAGLE 4.16 Professional

EAGLE 3D ulp modul do EAGLE 4.16 Professional

POV-Ray v. 3.6

Visual Basic 6

Wireshark

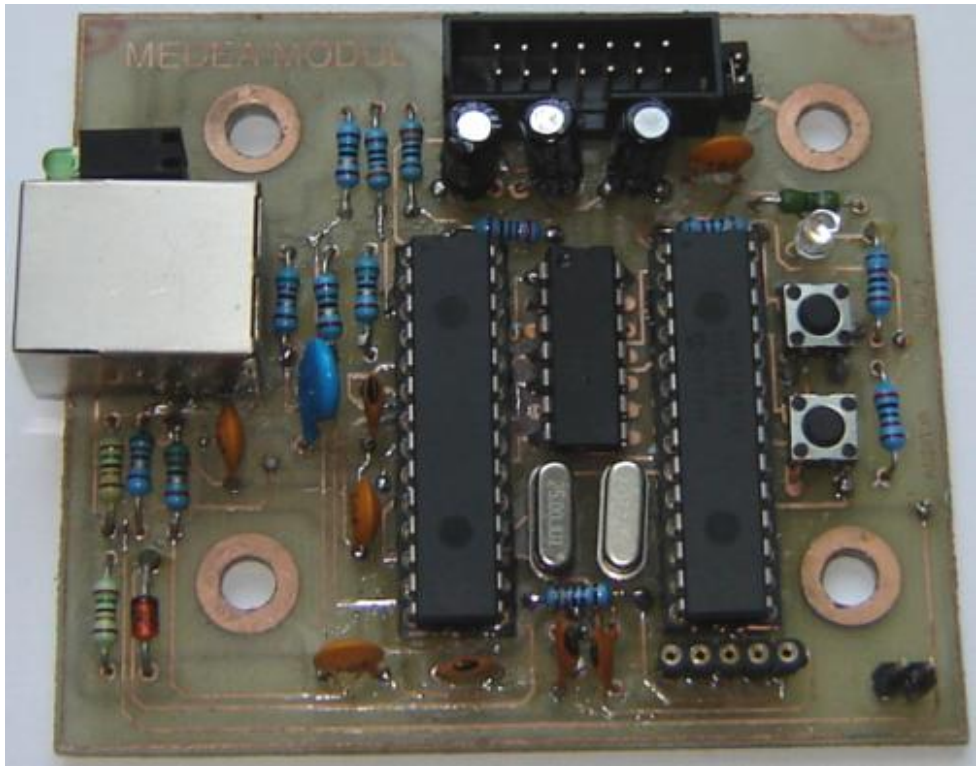
Terminal

IrfanView

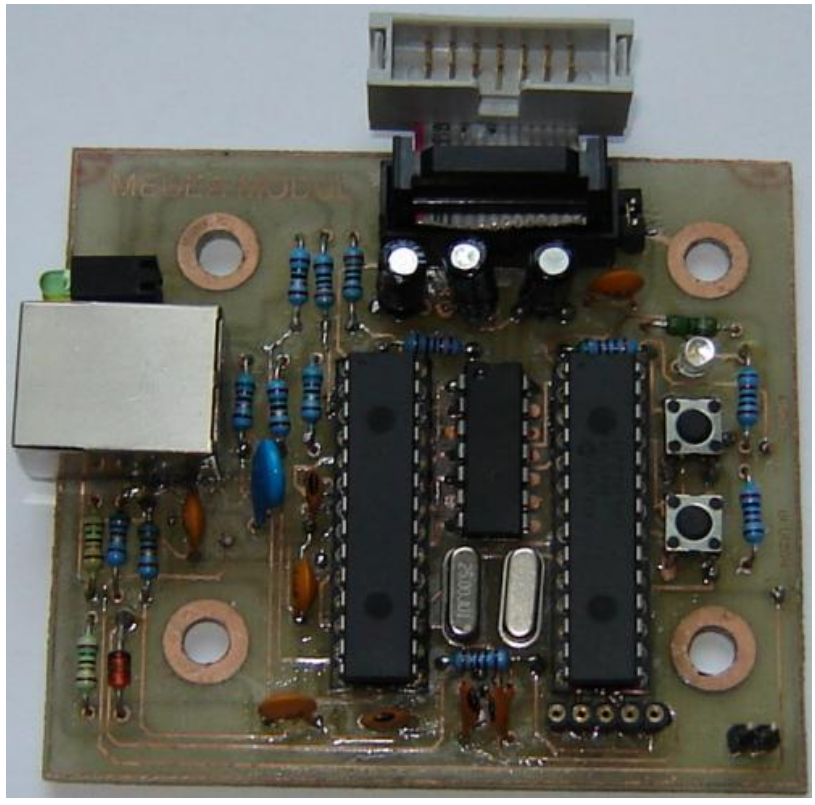
Balík programů firmy microsoft => Word, Visio, PowerPoint

12 Přílohy

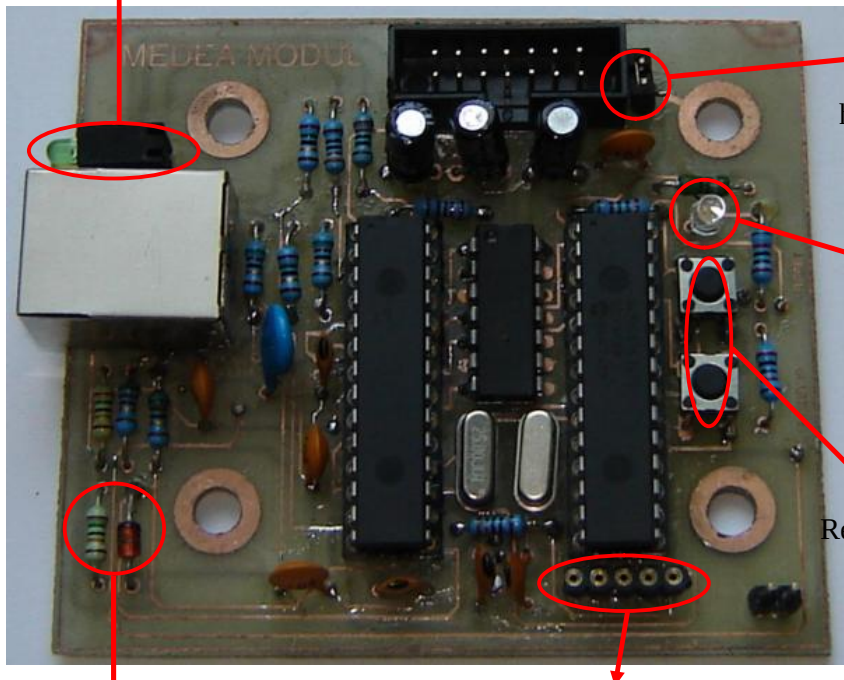
12.1 Fotky MEDEA MODULU



Redukce k MEDEA MODULU

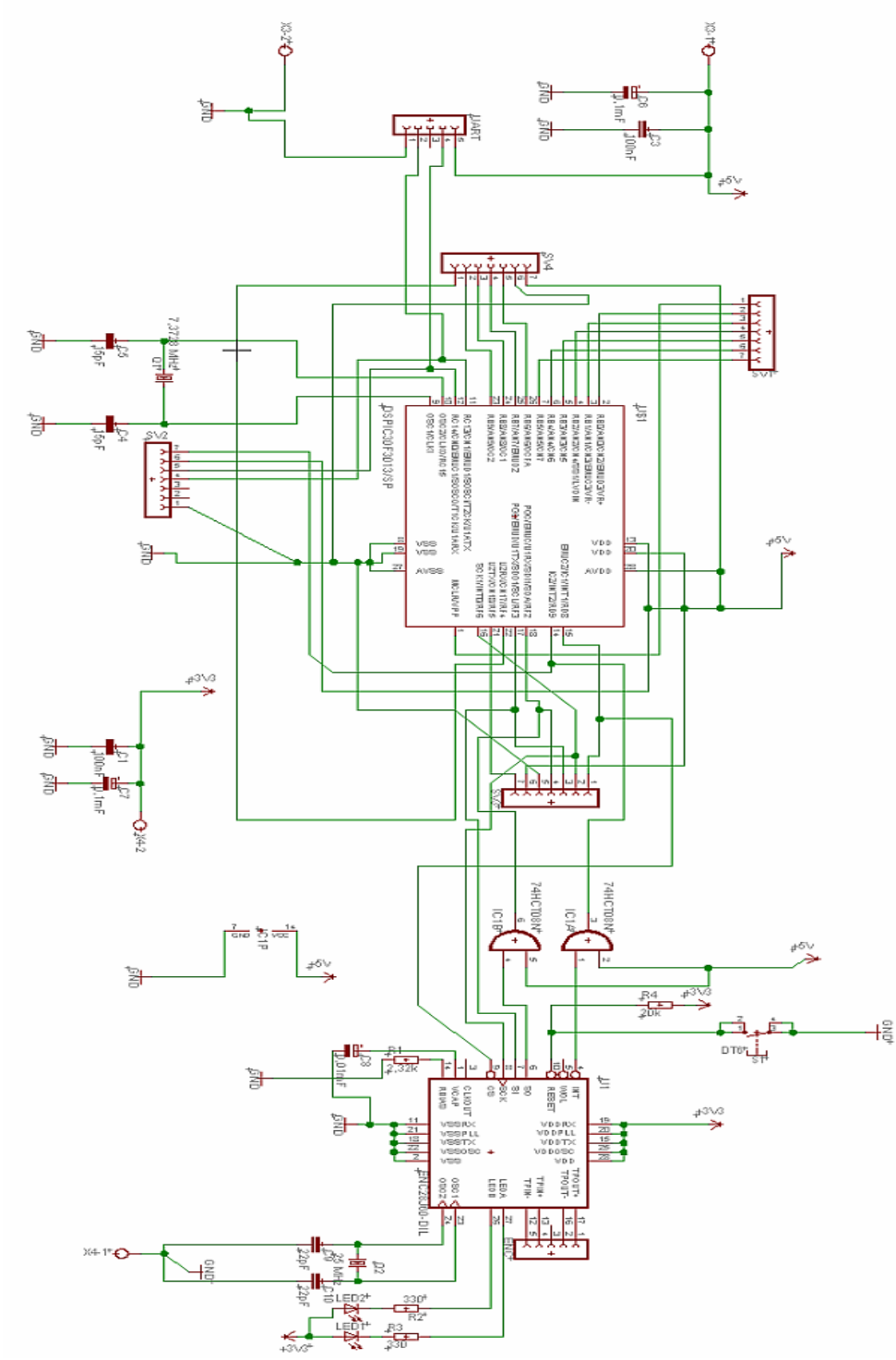


Diodový sloupek

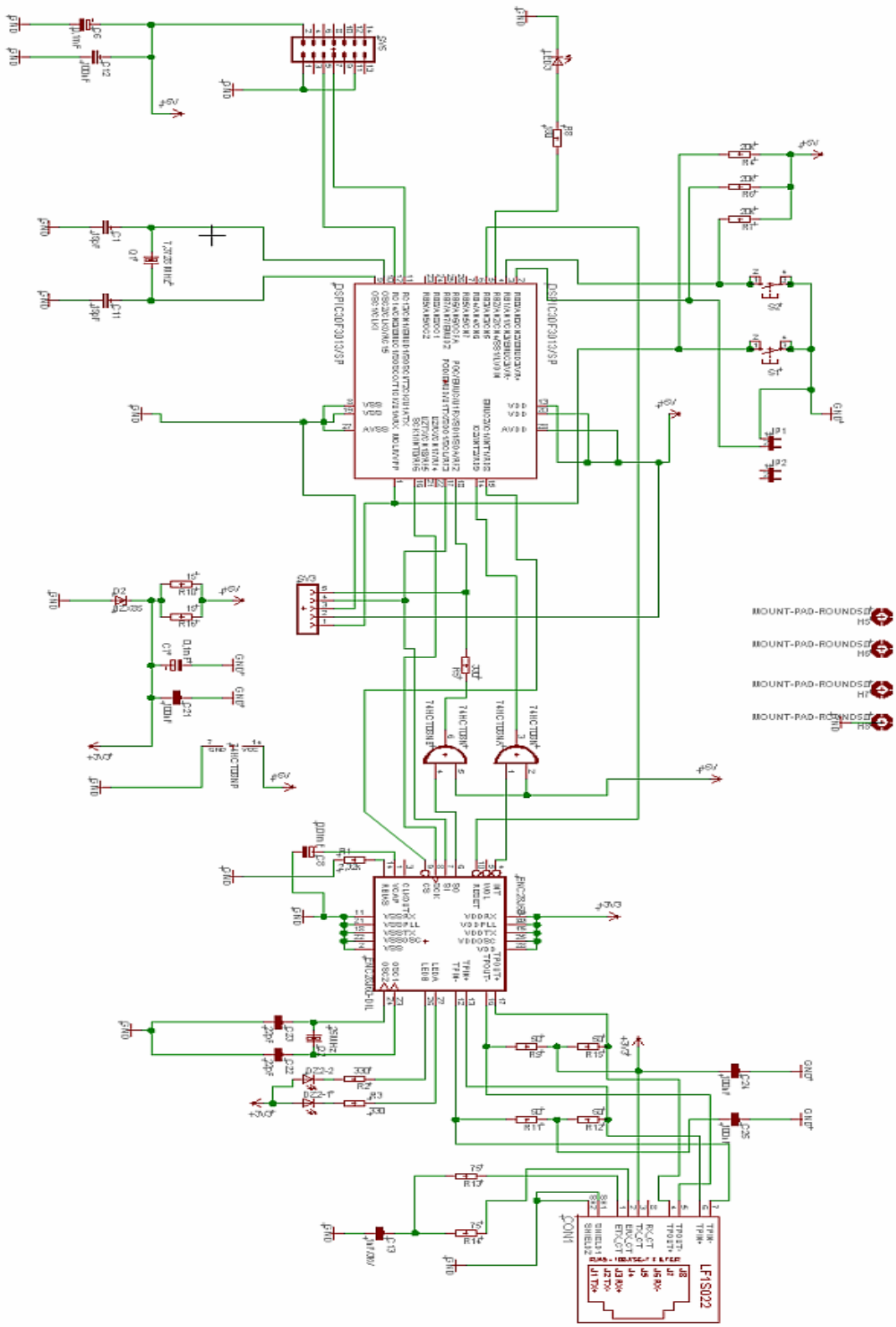


12.2 Schémata a návrhy desek

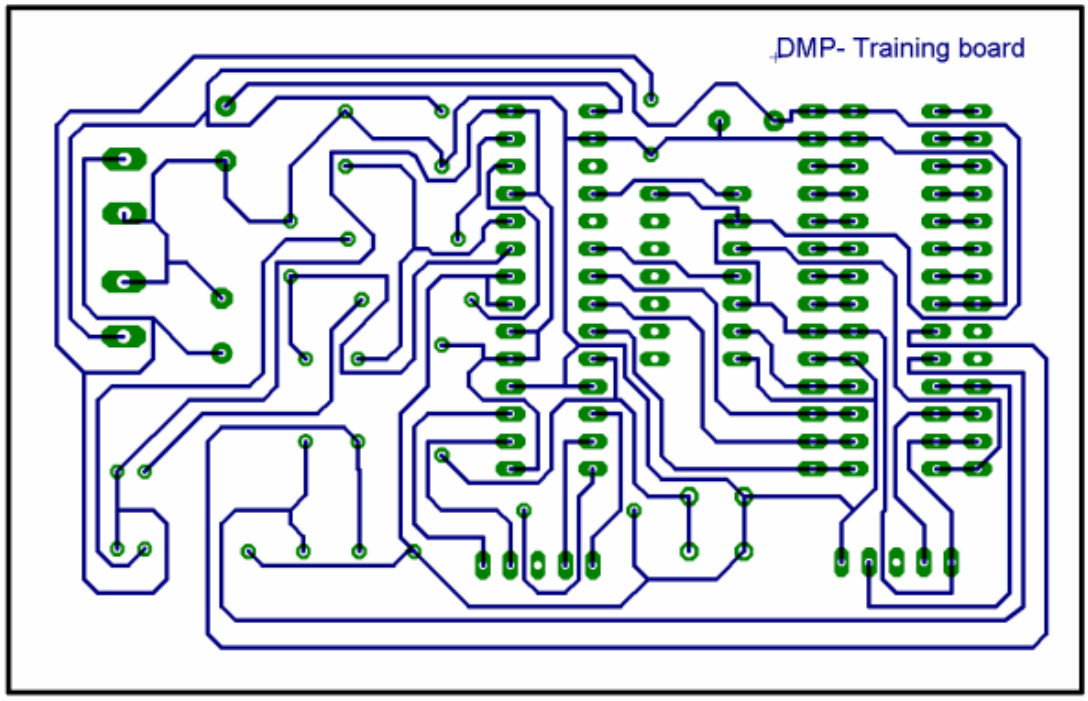
12.2.1 SCHÉMA – TRAINING BOARD



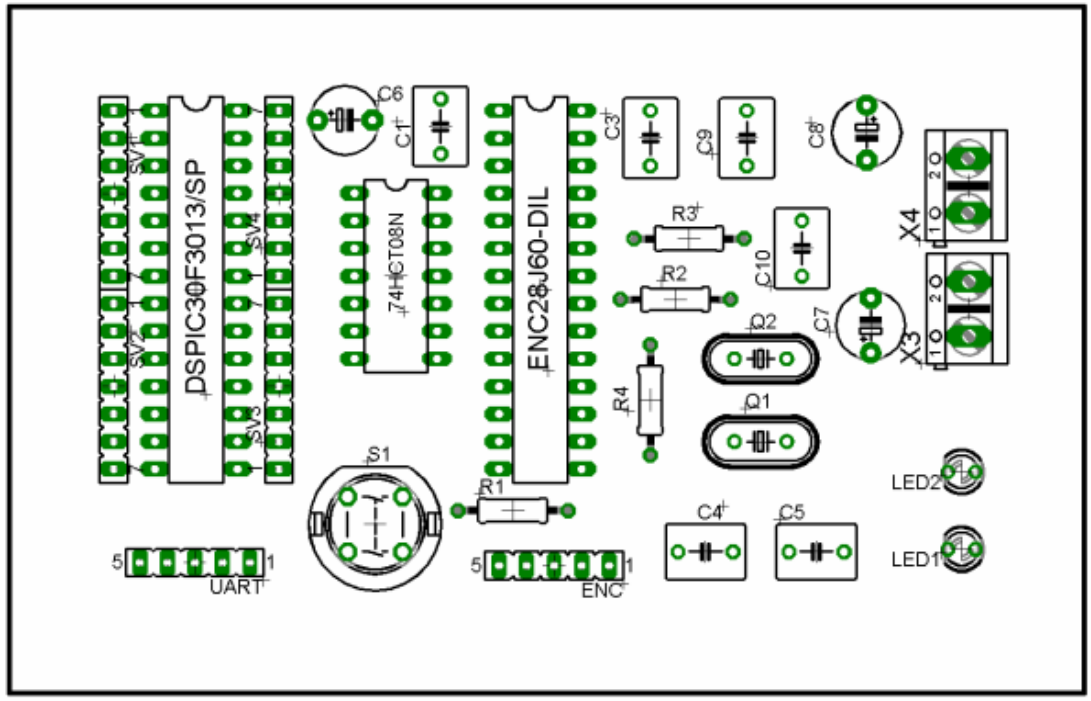
12.2.2 SCHÉMA – MEDEA MODUL



12.2.3 PLOŠNÉ SPOJE – TRAINING BOARD

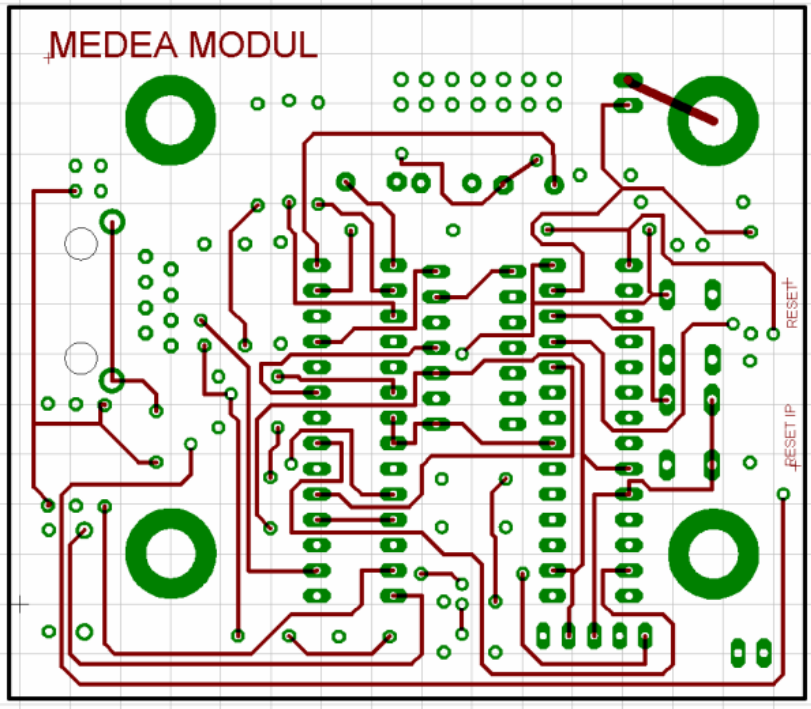


Náhled plošného spoje na vrstvu BOTTOM Training Board

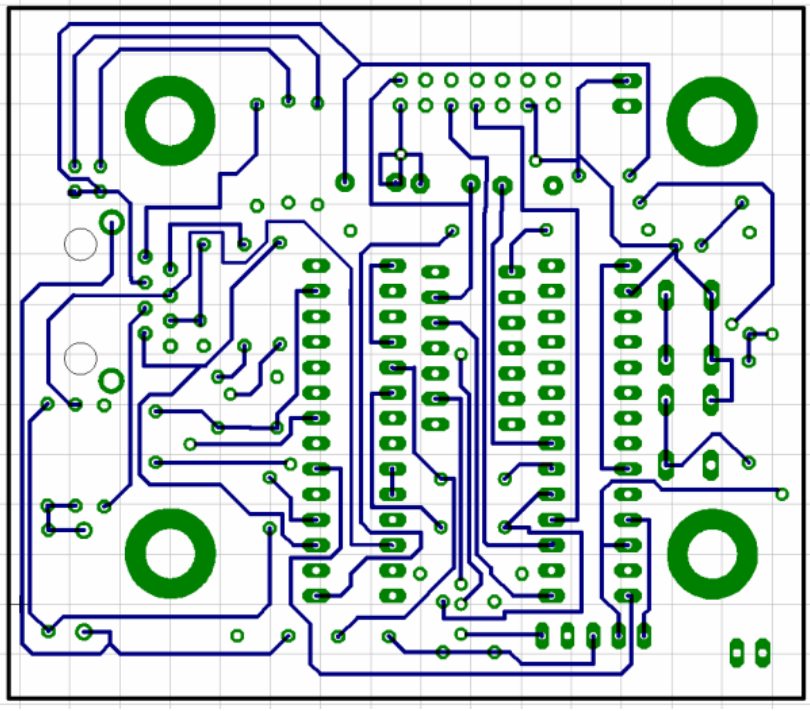


Rozmístění součástek na desce Training Board

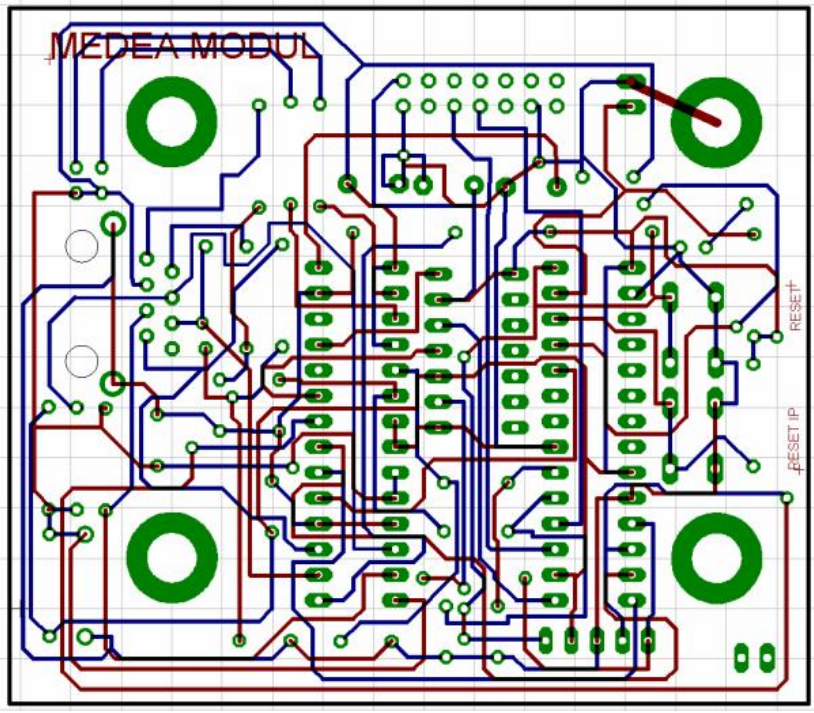
12.2.4 PLOŠNÉ SPOJE – MEDEA MODUL



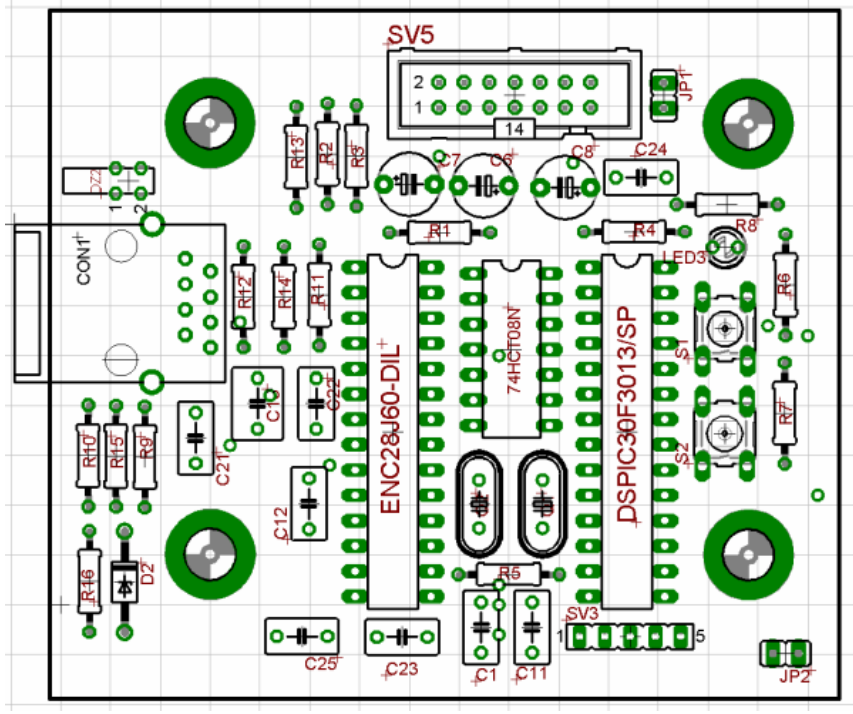
Náhled plošného spoje na vrstvu TOP Medea modulu(z pohledu součástek)



Náhled plošného spoje na vrstvu BOTTOM Medea modulu (z pohledu součástek)



Náhled plošného spoje na vrstvu TOP i BOTTOM zároveň (z pohledu séučástek)



Rozložení součástek na desce MEDEA MODUL