JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH

PEDAGOGICKÁ FAKULTA

KATEDRA FYZIKY

Diplomová práce

Autor: Jan Krejčí

Vedoucí práce: Ing. Michal Šerý

2009

Anotace

Tato diplomová práce je zaměřena na návrh a realizaci modulu pro řízení elektronický spotřebičů přes www. Navrhuji a popisuji jednotlivé části: naprogramování mikroprocesoru, návrh modulu pro spojeni s ethernetovým modulem firmy ASIX a využití v praxi.

Annotation

This diploma thesis is focused on the design and the realization of a module for controlling electric appliances over the internet. I have designed and described the following parts: programming of the microprocessor, the design of the module for the connection with the Ethernet module ASIX and its practical use.

Vzdálené řízení elektrických spotřebičů přes WWW

Prohlášení:

Prohlašuji, že jsem tuto práci vypracoval samostatně a že jsem všechny použité zdroje uvedl v seznamu použité literatury na konci této práce. Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své diplomové práce, a to v nezkrácené podobě fakultou elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách.

Datum: Podpis

Poděkování

Tímto bych chtěl poděkovat panu Ing. Michalu Šerému, za vedení při vypracování této práce.

Obsah

Úvod

Pod pojmem webový server si každý může představit velký, drahý a výkonný počítač, který je umístěn v „rackové“ skříni někde v místnosti pro servery. Ukážeme si, že to vždycky takto být nemusí. Tento velký, výkonný a drahý server nahradíme mikroprocesorem jehož cena nepřevýší dvou stovkovou hranici a ethernetovým modulem, který připojíme k mikroprocesoru. Ethernetový modul se bude starat o fyzickou a linkovou vrstvu a mikroprocesor bude řešit protokoly TCP/IP a číst vstupy a ovládat výstupy.

Do mikroprocesoru nahrajeme webové stránky ve strojovém kódu, připojíme k modulu počítač ethernetovým kabelem a přes webový prohlížeč se připojíme na tento náš server. Webový server s námi bude komunikovat přes webové rozhraní, čímž budeme moct číst jednotlivé vstupy a ovládat výstupy, bez použití nějakých dalších aplikací či hardwaru.

Samozřejmě tento webový server nebude srovnatelný s výkonnými servery, ale pro naši činnost bude dostatečně stačit.

V této diplomové práci si ukážeme fyzickou a linkovou vrstvu, síťové protokoly rodiny TCP/IP a návrh mikroprocesorové rozhraní včetně schématu a desky plošných spojů.

2 Fyzická a linková vrstva

Fyzická vrstva popisuje elektrické či optické signály používané při komunikaci mezi počítači.

Základní jednotkou pro přenos dat je na linkové vrstvě datový rámec Datový rámec se skládá ze záhlaví, přenášených dat a zápatí. Datový rámec nese v záhlaví linkovou adresu příjemce, linkovou adresu odesílatele a další řídící informace. V zápatí nese obvykle kontrolní součet z přenášených dat. Pomocí něho lze zjistit, zdali nedošlo při přenosu k porušení dat. Na fyzické vrstvě mohou být pro každý konec spojení použity jiné protokoly. U lokálních sítí se setkáváme s komplikovanějším případem, kdy mezi oba konce spojení je vložen např. přepínač (Switch), který konvertuje linkové rámce jednoho linkového protokolu na rámce jiného linkového protokolu (např. Ethernet na FDDI), což má pochopitelně za následek i použití jiných protokolů na fyzické vrstvě.

2.1 Lokální sítě LAN

Lokální sítě se svou přenosovou rychlostí v Mb/s až Gb/s se řadí mezi středně rychlé sítě. Cílem LAN je propojit mezi sebou počítače v rámci jedné nebo několika budov tak, aby mohly vzájemně mezi sebou komunikovat. Při použití optických rozvodů může LAN pokrývat území i několika kilometrů. V uplynulých deseti letech byla vyvinuta celá řada systémů LAN. Masového rozšíření se však dočkaly jen dva: Ethernet a v menším rozsahu FDDI. Pro připojení stanice na LAN je nutné do stanice vložit příslušnou síťovou kartu. Linkové protokoly LAN jsou realizovány z části přímo v síťové kartě.

Problematika LAN se vždy skládá z problematiky kabeláže, která patří do fyzické vrstvy, problematiky síťových karet, které se vkládají do počítačů a ostatních zařízení. To je součást jak fyzické vrstvy, tak i linkové vrstvy, protože část softwaru pro obsluhu linkové vrstvy je realizována přímo na síťové kartě a problematiky samotného linkového protokolu (včetně obsahu linkových rámců) a jeho realizace programy v počítači.

2.1.1 Ethernet

Protokol Ethernet byl původně vyvinut firmami DEC, Intel a Xerox. Jeho varianta 10 MHz se označuje jako Ethernet II. Později byl Ethernet normalizován institutem IEEE jako norma 802.3. Tato norma byla převzata ISO a publikována jako ISO 8802-3. Formát rámců podle normy Ethernet II se mírně odlišuje od formátu ISO 8802-3. Postupem času vznikla norma IEEE 802.3u pro Ethernet na frekvenci 100 MHz a norma IEEE 802.3z pro frekvenci 1 GHz. Původní rozvod Ethernetu byl prováděn koaxiálním kabelem označovaným 10BASE5. Koaxiální kabel, který mohl být dlouhý maximálně 500 metrů tvořil jeden segment lokální sítě. Segment Ethernetu byl většinou tvořen jedním kusem koaxiálního kabelu. Na koaxiální kabel byly napichovány transceivery, které se propojovaly kabelem na AUI-port ethernetové přídavné karty v počítači. AUI-port zpravidla používá konektor CANNON-15. Označení 10BASE5 vyjadřuje, že se jedná o síť používající přenosovou frekvenci 10 MHz ta je v případě Ethernetu rovná i teoretické přenosové rychlosti sítě.

Masově se Ethernet rozšířil na tzv. tenkém koaxiálním kabelu. Tenký koaxiální kabel je u každé stanice přerušen a na oba konce přerušení je buť napájen nebo speciálními kleštěmi namáčknut BNC-konektor. Mezi dva BNC-konektory se vloží BNC-T-konektor – "odbočka k počítači". Třetí vývod BNC-konektoru se nasadí přímo na ethernetovou síťovou kartu v počítači. Existují však i transceivery pro tenký Ethernet, pak se BNC-T-konektor připojí na transceiver pro tenký Ethernet a kabel z transceiveru se připojí na AUI-port počítače. Tenký Ethernet, označovaný jako 10BASE2 může být tvořen segmentem o maximální délce 185 metrů. Použijí-li se na segmentu stejné síťové přídavné karty, pak v případě některých karet je možné segment zvětšit až na 300-400 metrů.

Délka segmentu LAN je tedy 500 (resp. 185 – 300) metrů. Rozsah LAN je možné zvětšit tím, že použijeme více segmentů, které mezi sebou propojíme tzv. opakovače. Opakovač je tvořen dvěma nebo více síťovými kartami, které jsou vzájemně propojeny. Objeví-li se nějaký datový rámec na jednom rozhraní, pak je automaticky zopakován na všechny ostatní. Opakovač může být osazen AUI i BNC porty, takže některé segmenty mohou používat tlustý a jiné tenký Ethernet. Mezi dvěma opakovači může být použita i dvojice optických kabelů, tento typ Ethernetu se někdy označuje jako 10BASE-F. Délka optického propojení dvou opakovačů může být 1 km. Opakovač může být osazen i porty pro kroucenou „dvojlinku". V případě kroucené „dvojlinky" je

situace trochu odlišná. Kroucená „dvojlinka“ přesněji řečeno dva páry vodičů je rozhraní mezi opakovačem a počítačem. Spíše toto rozhraní připomíná rozhraní mezi transceiverem a AUI-konektorem neobsahuje však napájení. V případě kroucené „dvojlinky“ je jádrem sítě opakovač na rozdíl od koaxiálního kabelu. Z opakovače se hvězdicovitě rozbíhají kroucené „dvojlinky“ k jednotlivým počítačům. Opakovač pro kroucenou „dvojlinku“ se označuje jako HUB, může mít pochopitelně i BNC nebo AUI-porty.

Spoj mezi opakovačem a počítačem je tvořen dvěma páry kroucené dvoulinky. Jedná se o duplexní spoj, kde pro každý kanál je určen jeden pár. Z hlediska počítače je tedy jeden pár “vysílání” a druhý pár “příjem”. HUBy pro kroucenou dvojlinku je možné mezi sebou vzájemně propojovat. Ale pozor, co je pro jeden “vysílání”, je pro druhý “příjem”, takže v propojovací šňůře musí být páry překřížené. Většinou se však dodávají HUBy, kde jeden port je osazen přepínačem, který právě způsobí překřížení párů, takže stačí použít „normální” propojovací šňůru a připojit ji do portu s přepínačem a ten přepnout do vhodné polohy.

Ethernet na kroucené „dvojlince“ se označuje jako 10BASE-T. Existuje i verze desetkrát rychlejšího Ethernetu označovaná 100BASE-TX a gigabitový Ethernet označovaný 1000BASE-CX. (Pomocí opakovačů nelze kombinovat 10BASE-T, 100BASE-TX a 1000BASE-CX propojit je lze až pomocí přepínače). Délka dvojlinky mezi opakovačem a stanicí je standardně do 100 metrů.

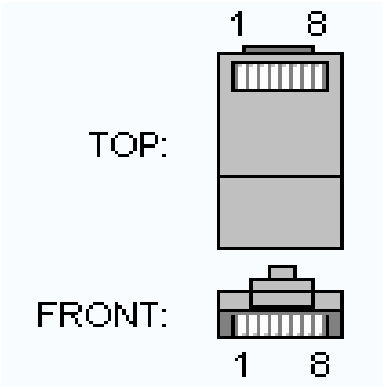
2.1.2 Strukturovaná kabeláž

Strukturovanou kabeláží se rozumí komplexní řešení nízkonapěťových rozvodů v budově. Zahrnuje zejména telefonní rozvody a rozvody pro LAN. Většinou zahrnuje i další rozvody jako jsou bezpečnostní a jiné signalizace. V jednotlivých místnostech budovy jsou umístěny telefonní zásuvky, zásuvky LAN a jiné vývody. Z těchto zásuvek vedou rozvody na propojovací panel budovy. V případě optických rozvodů jsou optická vlákna vyvedena na distribuční box optiky.

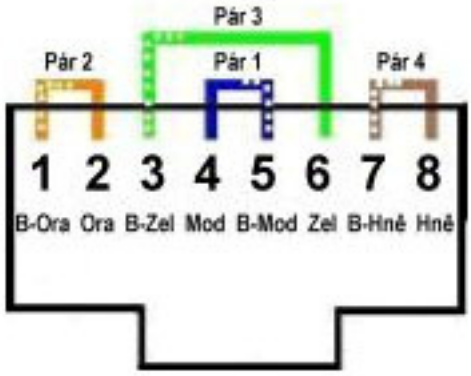
Propojovací panel a distribuční box optiky bývají uzavřeny v jedné skříni spolu s aktivními prvky LAN či dokonce i s telefonní ústřednou. Propojení mezi propojovacím panelem a aktivními prvky se provádí propojovacími kabely. Snahou je rozvod provést maximálně kvalitně, aby se rozvody nemusely často předělávat. Základní filozofií nových protokolů je pak v maximální míře využít stávající kabeláže.

2.1.3 Měděné rozvody

Měděné rozvody se provádí pomocí svazků kroucených „dvoulinek“. Jednotlivé místnosti se opatřují zásuvkami pro konektor RJ 45 (viz obr. 2.1.1). Konektor RJ 45 obsahuje 8 vývodů pro 4 páry. Nejčastěji se používá standart T568B (viz obr. 2.1.2).

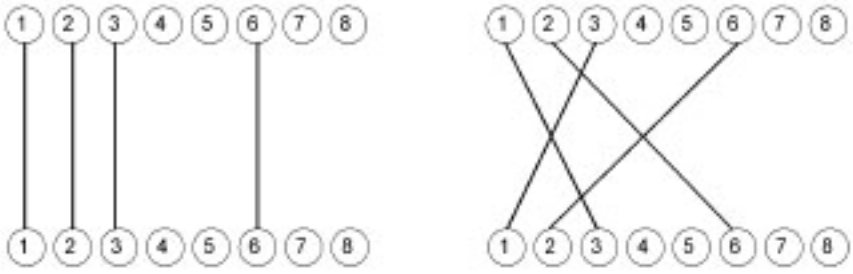


2.1.1 Konektor RJ45 (viz [1])



2.1.2 Zapojení podle standartu T568B (viz [1])

Pro připojení dvou počítačů, bez aktivního prvku, se musí použít tzv. křížený kabel (viz obr. 2.1.3)



2.1.3 Standardní zapojení vlevo, křížený zapojení vpravo (viz [1])

3 Protokol TCP/IP

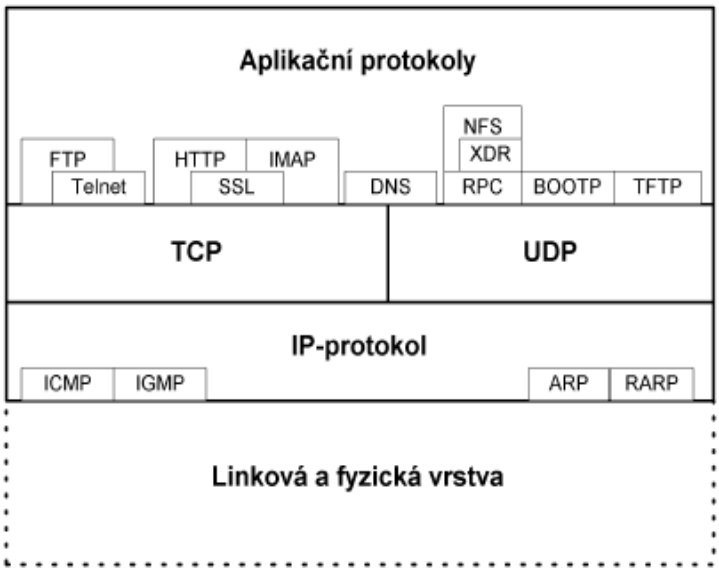
Internet Protokol (dále jen IP-protokol) prakticky odpovídá síťové vrstvě. IP-protokol přenáší tzv. IP-datagramy mezi vzdálenými počítači. Každý IP-datagram ve svém záhlaví nese adresu příjemce, což je úplná směrovací informace pro dopravu IP-datagramu k adresátovi. Takže síť může přenášet každý IP-datagram samostatně. IP-datagramy tak mohou k adresátovi dorazit v jiném pořadí než byly odeslány. Každé síťové rozhraní v rozsáhlé síti Internet má svou celosvětově jednoznačnou IP-adresu (jedno síťové rozhraní může mít více IP-adres, avšak jednu IP-adresu nesmí používat více síťových rozhraní).

Protokol TCP odpovídá transportní vrstvě. Protokol TCP dopravuje data pomocí TCP segmentů, které jsou adresovány jednotlivým aplikacím. Protokol TCP zajišťuje spojení mezi aplikacemi běžícími na vzdálených počítačích, může zajišťovat i komunikaci mezi procesy běžícími na témže počítači. Protokol TCP je tzv. spojovanou službou, příjemce potvrzuje přijímaná data. V případě ztráty dat si příjemce vyžádá zopakování přenosu. Zatímco např. protokol UDP přenáší data pomocí datagramů, odesílatel odešle datagram a už se nezajímá o to, zda-li byl doručen. Adresou je tzv. port. Pro pochopení rozdílu mezi IP-adresou a portem se používá srovnání s poštovní adresou. IP-adresa odpovídá adrese domu a port jménu a příjmení osoby, které má být dopis doručen.

Aplikační vrstva je zredukována do jedné aplikační vrstvy TCP/IP. Aplikačních protokolů je velké množství. Z praktického hlediska je lze rozdělit na:

- Uživatelské protokoly, které využívají uživatelské aplikace (např. pro vyhledávání informací v Internetu). Příkladem takových protokolů jsou protokoly: HTTP, SMTP, Telnet, FTP, IMAP, POP3 atd.
- Služební protokoly, tj. protokoly se kterými se běžní uživatelé Internetu nesetkají. Tyto protokoly slouží pro správnou funkci Internetu. Jedná se např. o směrovací protokoly, které používají směrovače mezi sebou, aby si správně nastavily směrovací tabulky.

Na obrázku 3.1 je zřejmé, že rodina protokolů TCP/IP využívá čtyři vrstvy. Naproti tomu např. protokoly ISO OSI používají vrstev dokonce sedm. Protokoly TCP/IP neřeší (až na výjimky) linkovou a fyzickou vrstvu, proto se i v Internetu setkáváme s linkovými a fyzickými protokoly z modelu ISO OSI. (viz [1])



3.1 Síťový model TCP/IP (viz [1])

3.1 Protokol IP

IP-Protokol dopravuje data mezi dvěma libovolnými počítači v Internetu. Data jsou od odesílatele k příjemci dopravovaná přes směrovače. Data jsou tak předávána od směrovače k směrovači. IP-Protokol je protokol umožňující spojit jednotlivé lokální sítě do celosvětového internetu.

IP-protokol je tvořen několika dílčími protokoly:

Vlastní protokol IP

Protokol ICMP – sloužící k signalizaci mimořádných stavů

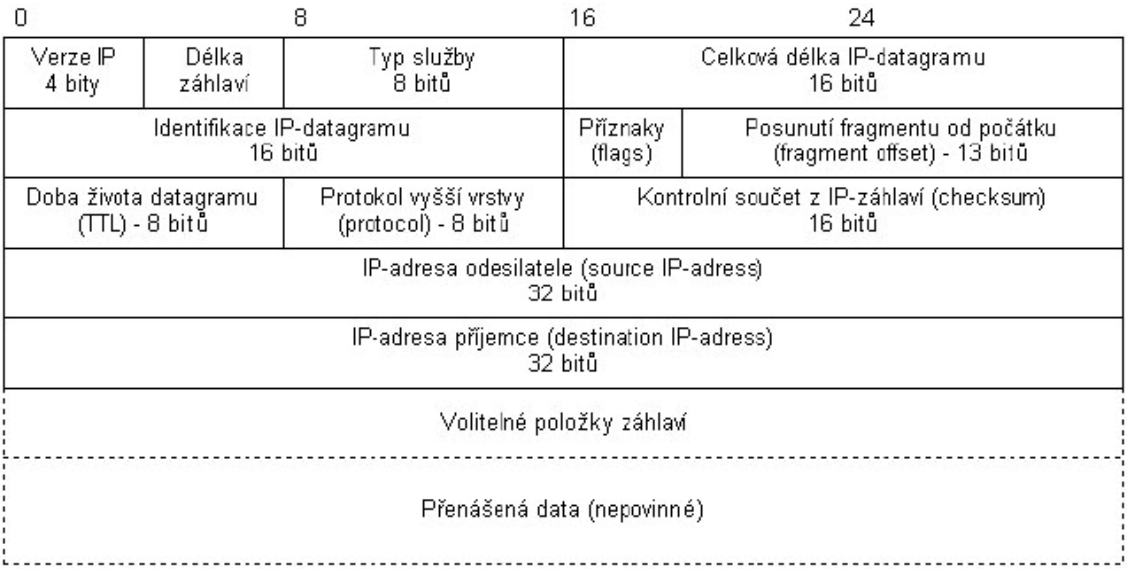
Protokol IGMP – sloužící k dopravě adresných oběžníků

Protokoly ARP a RARP

V IP-protokolu má každé síťové rozhraní alespoň jednu IP-adresu, která může být čtyřbajtová nebo šestnáctibajtová.

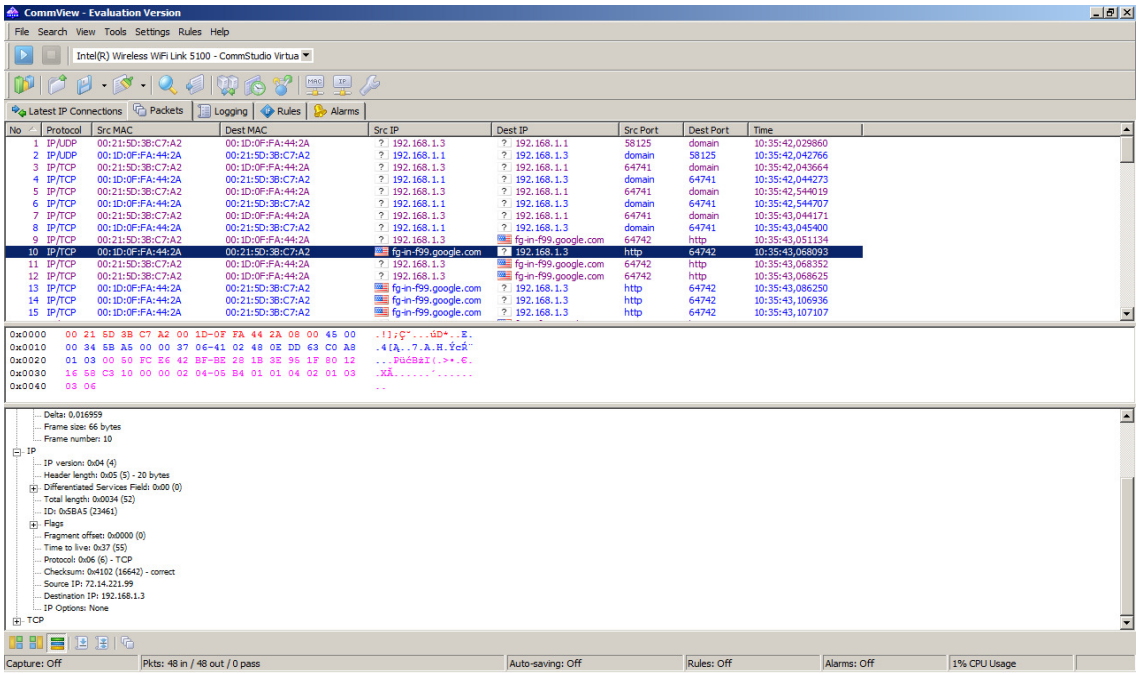
Základním stavebním prvkem WAN je směrovač, kterým se vzájemně propojují jednotlivé LAN do rozsáhlé sítě.

U IP-protokolu je základní jednotkou přenášených dat IP-datagram. Ip-datagram se skládá ze záhlaví a přenášených dat. Záhlaví má zpravidla 20 bajtů, může však obsahovat volitelné položky. Na obrázku 2.1 je zobrazena struktura IP-datagramu. (viz [1])



2.2 Struktura IP-datagramu (viz [1])

Pro znázornění bude jeden datagram programem CommView odchycen a popsán. Při spuštění programu se vybere síťová karta na které se bude monitorovat provoz a spustí se monitorování. V záložce Packets je okno rozděleno do tří částí. V první části jsou jednotlivé pakety. Je zde vidět jaký druh paketu to je, jakou má zdrojovou a cílovou adresu a samozřejmě zdrojový a cílový port. Ve druhé části okna vidíme pouze strojová data, který jsou rozkódována v třetí části okna. (viz obr. 2.3)



2.3 IP datagram odchycený programem CommView

Verze (version) je první položkou záhlaví IP-datagramu. Tato položka dlouhá 4 bity (půl bajtu) obsahuje verzi IP-protokolu. V této kapitole hovoříme o IP-protokolu verze 4, tudíž tato položka je v našem případě rovná hodnotě 4.

Délka záhlaví (header length) obsahuje délku záhlaví IP-datagramu. V případě odchyceného IP-datagramu na obr. 2.2 je délka záhlaví 20, ale jak je vidět z hexadecimálního výpisu z CommView, tak položka délka záhlaví nabývá hodnoty 5 (nikoliv 20). Vysvětlení je prosté. Délka není uváděna v bajtech, ale v čtyřbajtech a 5x4=20. Délka záhlaví musí tak být i v případě použití volitelných položek násobkem čtyř. V případě, že by záhlaví nevyšlo na násobek čtyř, pak se na násobek čtyř doplní nevýznamnou výplní. Maximální délka záhlaví IP-datagramu je tedy omezena tím, že položka délka záhlaví má k dispozici pouze 4 bity . Délka záhlaví IP-datagramu je tedy maximálně 60 B. Jelikož povinné položky mají 20 B, tak na volitelné položky zbývá maximálně 40 B. Typ služby (type of service – TOS) je položka, která v praxi nenašla svého naplnění. V normách RFC-791 a RFC-1349 lze nalézt konkrétní návrhy využití. Záměr spočíval v jistém nedostatku IP-protokolu jehož podstatou je skutečnost, že v Internetu není zaručena šíře přenosového pásma mezi účastníky. Jistého vylepšení se mělo dosáhnout právě touto položkou, pomocí které je možné označit některé IP-datagramy tak, aby byly dopravovány přednostně či aby byla zaručena rychlá odezva atp. (viz [1])

Celková délka IP-datagramu (total length) obsahuje celkovou délku IP-datagramu v bajtech. Jelikož je tato položka pouze dvojbajtová, tak maximální délka IP-datagramu je 65535 bajtů.

Identifikace IP-datagramu (identification) obsahuje identifikaci IP-datagramu, kterou do IP-datagramu vkládá operační systém odesílatele. Tato položka se společně s položkami příznaky (flags) a posunutí fragmentu (fragment offset) využívá mechanismem fragmentace datagramu.

Do češtiny se názvy bitů pole příznaky překládají v negaci. Je-li DF bit nastaven na 1, pak je fragmentace zakázána. Nastavení na 0 naopak znamená, že fragmentace je možná. Je-li nastaven bit MF na jedničku, pak vyjadřuje, že není posledním fragmentem.

Doba života datagramu (time to live – TTL) slouží k zamezení nekonečného toulání IP-datagramu Internetem. Každý směrovač kladnou položku TTL snižuje alespoň o jedničku. Není-li už možné hodnotu snížit, IP-datagram se zahazuje a odesílateli IP-datagramu je tato situace signalizována protokolem ICMP. Jak se hodnota

položky TTL nastavuje? U příkazů ping a traceroute je možné ji explicitně nastavit. Obecně se však jedná o parametr jádra operačního systému, pokud ji tvůrci programu nenastaví explicitně).

Protokol vyšší vrstvy (protocol) obsahuje číselnou identifikaci protokolu vyšší vrstvy, který využívá IP-datagram ke svému transportu. V praxi se nesetkáváme s případem, že by se komunikovalo přímo IP-protokolem. Vždy je použit protokol vyšší vrstvy (TCP nebo UDP) nebo jeden ze služebních protokolů ICMP či IGMP. Protokoly ICMP a IGMP jsou sice formálně součástí protokolu IP, avšak chovají se jako protokoly vyšší vrstvy, tj. v přenášeném paketu je záhlaví IP-protokolu následováno záhlavím protokolu ICMP (resp. IGMP). Čísla protokolů vyšších vrstev přiřazuje tvůrcům protokolů vyšších vrstev organizace IANA. Přiřazená čísla lze najít na http://www.iana.org.

Jako protokol vyšší vrstvy může být i protokol, který je tunelován přes Internet (encapsulation). Tunelovány mohou být např. protokoly, které Internet nepodporuje, jako je např. protokol IPX. Nebo může být tunelován sám protokol IP (IP over IP). Tunelování IP přes IP se může na první pohled jevit jako nesmyslné plýtvání. Avšak v případě, že přes Internet chceme přenášet data mezi dvěma částmi privátní sítě o adrese 10.0.0.0, pak je takový tunel nezbytností. Navíc je možné vnitřní IP-datagramy zabezpečit šifrováním a vznikne nám tak jednoduchá virtuální privátní síť (VPN).
Pokud je třeba transportovat datagramy protokolu IP verze 6 přes síť podporující pouze IP-protokol verze 4, pak také nezbývá opět nic jiného než tunelování.

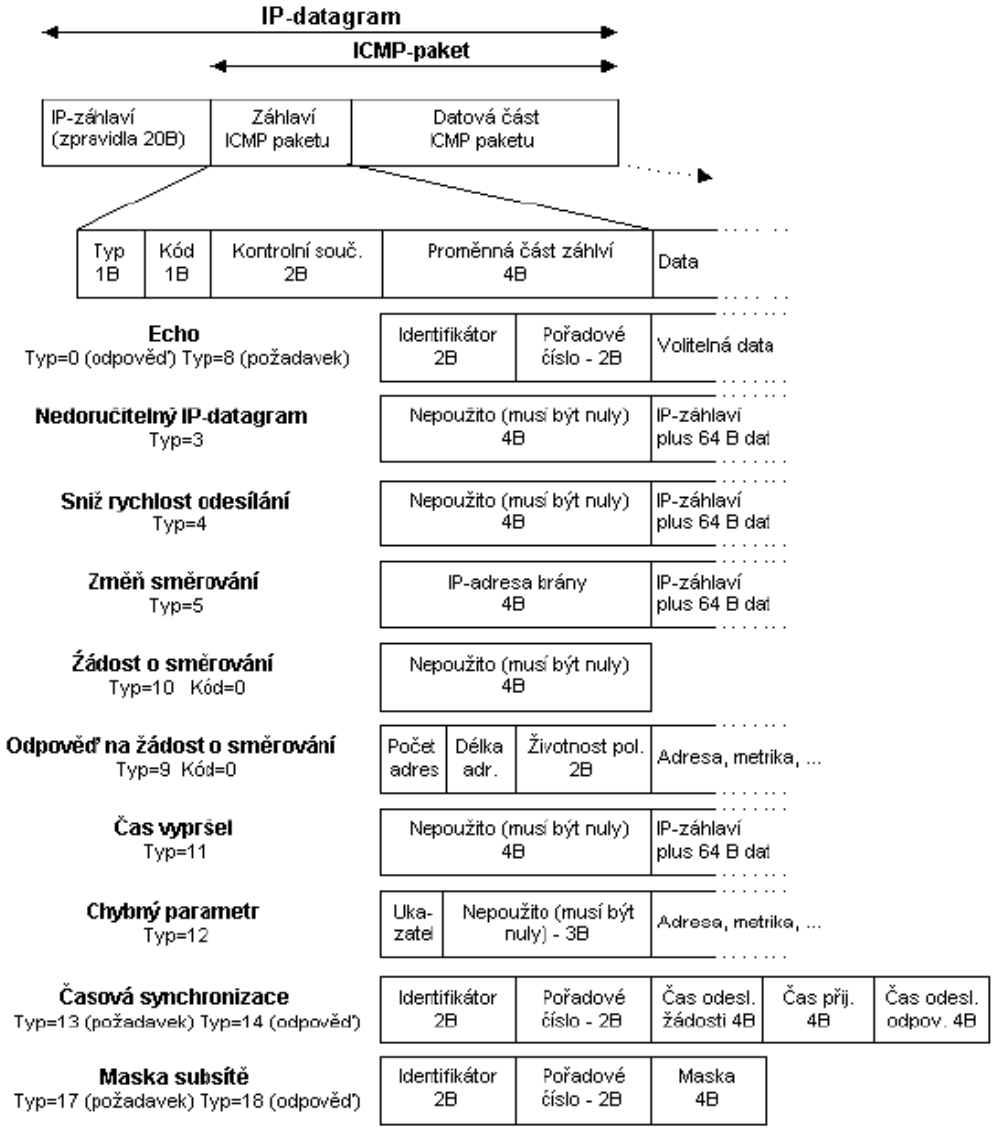
Kontrolní součet z IP-záhlaví (header checksum) obsahuje kontrolní součet, avšak pouze ze záhlaví IP-datagramu a nikoliv z datagramu celého. Jeho význam je tedy omezený. Bližší informace o výpočtu kontrolního součtu lze nalézt v normách RFC-1071 a RFC-1141. Problém s kontrolním součtem spočívá v tom, že když směrovač změní nějakou položku v záhlaví IP-datagramu (např. TTL změnit musí), tak musí změnit i hodnotu kontrolního součtu, což vyžaduje jistou režii směrovače.

IP-adresa odesílatele a IP-adresa příjemce (source and destination adress) obsahuje čtyřbajtovou IPadresu odesílatele a příjemce IP-datagramu.

Volitelné položky jsou využívány ojediněle a zpravidla směrovače bývají nakonfigurovány tak, aby IPdatagramy s použitými volitelnými položkami byly bez okolků zahozeny. (viz [1])

3.2 Protokol ICMP

Protokol ICMP je služební protokol, který je součástí IP-protokolu. Protokol ICMP slouží k signalizaci mimořádných událostí v sítích postavených na IP-protokolu. Protokol ICMP svoje datové pakety balí do IP-protokolu, tj. pokud budeme prohlížet přenášené datagramy, pak v nich najdeme za linkovým záhlavím záhlaví IP-protokolu následované záhlavím ICMP paketu. Protokolem ICMP je možné signalizovat nejrůznější situace, skutečnost je však taková, že konkrétní implementace TCP/IP podporují vždy jen jistou část těchto signalizací a navíc z bezpečnostních důvodů mohou být na směrovačích mnohé ICMP signalizace zahazovány.



2.4 Záhlaví ICMP paketu (viz [1])

Záhlaví ICMP-paketu je vždy osm bajtů dlouhé (viz obr. 2.4). První čtyři bajty jsou vždy stejné a obsah zbylých čtyř závisí na typu ICMP-paketu. První čtyři bajty záhlaví obsahují vždy typ zprávy, kód zprávy a šestnáctibitový kontrolní součet. Formát zprávy závisí na hodnotě pole Typ. Pole Typ je hrubým dělení ICMP-paketů. Pole Kód pak specifikuje konkrétní problém (jemné dělení), který je signalizován ICMP-protokolem. Přehled jednotlivých typů a kódů uvádí obr.2.5

Typ	**Kód**	**Popis**	**Co signalizuje**	**Kdo zpracovává**
0	0	Echo	Odpověď už. aplikaci	Uživ.aplikace
3		**Nedoručitelný IP-datagram (Destination unreacheble)**	Chyba	Uživ.aplikace
	0	Nedosažitelná síť *(Network unreachable)*		
	1	Nedosažitelný uzel *(Host unreachable)*		
	2	Nedosažitelný protokol *(Protocol unreachable)*		
	3	Nedosažitelný port protokolu UDP *(Port unreachable)*		
	4	Fragmentace zakázána, avšak pro další přenos by byla nutná *(Fragmentation needed but don't fragment bit set)*		
	5	Explicitní směrování selhalo *(Source route failed)*		
	6	Adresátova síť je neznámá *(Destination network unknown)*		
	7	Adresátův uzel je neznámý *(Destination host unknown)*		
	9	Adresátova síť je administrativně uzavřena *(Destination network administratively prohibited)*		
	10	Adresátův uzel je administrativně uzavřen *(Destination host administratively prohibited)*		
	11	Nedosažitelná síť pro uvedený typ služby *(Network unreachable for TOS)*		
	12	Nedosažitelný uzel pro uvedený typ služby *(Host unreachable for TOS)*		
	13	Komunikace administrativně uzavřena filtrací *(Communication administratively prohibited by filtering)*		
4	0	**Sniž rychlost odesílání (Source quench)**	Chyba	**1.** Jádro OS pro TCP **2.** Zahazuje se pro UDP

2.5 Přehled zpráv protokolu ICMP (viz [1])

5		**Změň směrování** ***(Redirect)***	Chyba	Jádro OS
	0	Změň směrování pro síť *(Redirect for network)*		
	1	Změň směrování pro uzel *(Redirect for host)*		
	2	Změň směrování pro síť pro daný typ služby *(Redirect for TOS and network)*		
	3	Změň směrování pro uzel pro daný typ služby *(Redirect for TOS and host)*		
8	0	**Žádost o echo** ***(Echo rquest)***	Dotaz už. aplikace	Jádro OS
9	0	**Odpověď na žádost o směrování** ***(router advertisement)***	Odpověď už. aplikací	Uživ.proces
10	0	**Žádost o směrování** ***(router solicitation)***	Dotaz už.aplikace	Uživ.proces
11		**Čas vypršel** ***(time exceeded)***	Chyba	Uživ.proces
	0	Čas vypršel během transportu *(TTL equals 0 during transit)*		
	1	Vypršel čas na sestavení IP-datagramu z jeho fragmentů *(time to live equals 0 during reasembly)*		
12		**Chybný parametr** ***(parameter problem)***	Chyba	Uživ.proces
	0	Chybné IP-záhlaví *(IP header bad)*		
	1	Schází požadovaný volitelný parametr *(required option missing)*		
13	0	**Požadavek na časovou synchronizaci** ***(timestamp request)***	Dotaz už. aplikace	Uživ.proces
14	0	**Odpověď na časovou synchronizaci** ***(timestamp reply)***	Odpověď už. aplikací	Jádro OS
17	0	**Žádost o masku subsítě** ***(address mask request)***	Dotaz už. aplikace	Uživ. proces
18	0	**Odpověď na žádost o masku** ***(address mask reply)***	Odpověď už. aplikace	Jádro OS

2.6 Přehled zpráv protokolu ICMP (pokračování) (viz [1])

Echo je jednoduchý nástroj protokolu ICMP, kterým můžeme testovat dosažitelnost jednotlivých uzlů v Internetu. Žadatel vysílá ICMP-paket „Žádost o echo“ a cílový uzel je povinen odpovědět ICMP-paketem „Echo“. Všechny operační systémy podporující protokol TCP/IP obsahují program ping, kterým uživatel může na cílový uzel odeslat žádost o echo. Program ping pak zobrazuje odpověď. Význam pole identifikátor v záhlaví ICMP-paketu spočívá ve spárování žádosti s odpovědí (aby se dalo zjistit, ke které žádosti patří příslušná odpověď).

Nedoručitelný IP-datagram - nemůže-li být IP-datagram předán dále směrem k adresátovi, pak je zahozen a odesílatel je protokolem ICMP o tom uvědoměn zprávou „Nedoručitelný IP-datagram“.

Sniž rychlost odesílání - jestliže je síť; mezi odesílatelem a příjemcem v některém místě přetížena, pak směrovač, který není schopen předávat dále všechny IP-datagramy signalizuje odesílateli „Sniž rychlost odesílání“. Odesílatel pak v případě,

že používá protokol TCP snižuje rychlost odesílání TCP segmentů. V případě protokolu UDP se zprávy „Sniž rychlost odesílání“ ignorují.

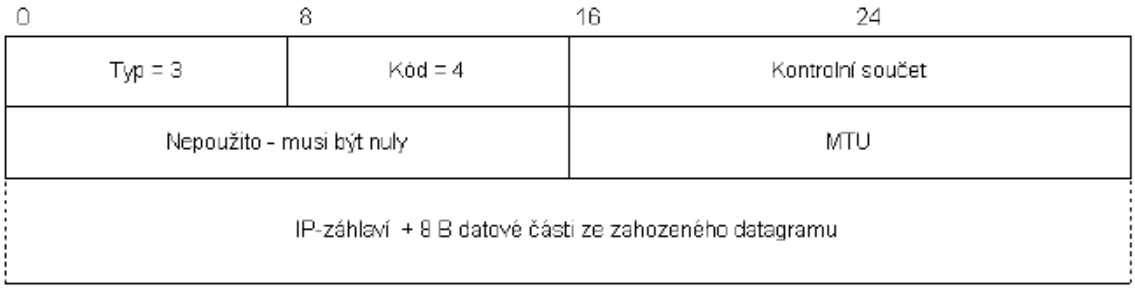
Žádost o směrování - jedná se o poměrně novou záležitost, pomocí které nemusíme do směrovací tabulky počítačů na LAN ručně konfigurovat vůbec žádnou položku default. Počítač po svém startu vyšle oběžníkem „Žádost o směrování“ a směrovač mu odpoví ICMP-paketem „Odpově1 na žádost o směrování“, která obsahuje: počet adres směrovače, délku adresy a pak dvojice IP-adresa a preference. Z odpovědi může počítač automaticky vygenerovat položku default. Čím má preference vyšší hodnotu, tím je IP-adresa více preferována. Hodnota preference $(80000000)_{16}$ signalizuje, že tato adresa se má ze směrovací tabulky vypustit. Směrovače odpovídají na žádost o směrování, avšak v náhodném intervalu mezi 450 a 600 vteřinami by měly oběžníkem samy do lokální sítě generovat ICMP-pakety „odpově1 na žádost o směrování“. Položka „doba života“ udává čas, po který je informace platná, tj. po který má být položka ve směrovací tabulce udržována.

Čas vypršel - tento typ zahrnuje dva velmi odlišné případy. Pro kód=0 signalizuje, že položka TTL by byla na směrovači snížena na nulu, tj. že je podezření, že IP-datagram v Internetu zabloudil, proto bude zlikvidován. Pro kód=1 signalizuje, že počítač adresáta není schopen v daném čase sestavit z fragmentů celý IP-datagram. ICMP-paket čas vypršel kód=0 využívá ke své činnosti program tracert. Tento program odesílá ze zdrojového počítače na cílový uzel ICMP-pakety „Žádost o echo“, avšak v prvním paketu nastaví položku TTL na jedničku. První směrovač na cestě paket zahodí a vrátí ICMP-paket „Čas vypršel“, protože musí zmenšit TTL alespoň o jedničku, ale tímto zmenšením už dostane nulu. Zdrojový počítač tak od prvního směrovače na cestě dostane v IP-datagramu ICMP-paket „čas vypršel“. Z položky adresa odesílatele v IP-záhlaví lze zjistit adresu prvního směrovače na cestě. Změří se časový interval od odeslání po příjem paketu a zjistí se tak čas procházky paketu od odesílatele k příjemci a zpět. Toto se opakuje třikrát a všechny tři časy se zobrazí. Na konec řádku ještě zobrazí jméno směrovače a v hranatých závorkách jeho IP-adresu. Jméno získá z reverzního překladu v DNS. Nezíská-li v časovém limitu odpově1, zobrazí místo času hvězdičku (*). Poté vše opakuje s hodnotou TTL=2 atd. Svou činnost ukončí, když od cílového uzlu obdrží ICMP-zprávu „Echo“. K ukončení může pochopitelně také dojít, když nějaký směrovač nezná cestu k cílovému počítači, pak zdrojovému počítači zašle zprávu „nedoručitelný IP-datagram“. (viz [1])

3.3 Fragmentace

IP-datagramy jsou baleny do linkových rámců. Linkové protokoly umožňují přenášet ve svých datových rámcích data pouze do určité maximální velikosti. Tato maximální velikost dat, která lze vložit do jednoho linkového rámce se označuje MTU (Maximum Transfer Unit). Pro ethernet je to 1500MTU

Linkové protokoly mají nejčastěji MTU řádkově v jednotkách KB. Na linkách spojujících vzdálené lokality se někdy dokonce setkáváme s MTU menším než 1 KB. Pole celková délka IP-datagramu je však dlouhé 16 bitů, takže teoreticky je možné vytvořit IP-datagram až 64 KB dlouhý. Co se však stane, když IP-datagram na své pouti od odesílatele k příjemci dorazí na směrovač z něhož směrem k příjemci vede linka, která má menší MTU než je velikost našeho IP-datagramu. Směrovač není schopen takový IP-datagram poslat dále. Směrovač se rozhoduje co dále na základě příznaku „Fragmentace možná" (DF bit) v záhlaví IP-datagramu (ponecháváme stranou možnost, že k příjemci vede ještě jiná linka, byť s horší metrikou). Příznak „Fragmentace možná" může být nastaven nebo ne. Jsou tedy dvě možnosti: Fragmentace je možná, pak se provede fragmentace, jak je popsáno dále v této kapitole. Fragmentace není možná, pak směrovač IP-datagram zahodí a odesílatele o tom informuje ICMP signalizací. Zakážeme-li příznakem fragmentaci, pak můžeme i zjistit jaké nejmenší MTU je mezi odesílatelem a příjemcem, tj. jak velké IP-datagramy nebude nutné fragmentovat. Podstatně jednodušší by bylo, kdyby ICMP-signalizace obsahovala hodnotu MTU, která platí pro inkriminovanou linku. Původně nebylo s touto možností počítáno, avšak později byl ICMP-paket pro tento případ doplněn o pole MTU. Tato možnost je jen zřídka implementována. V ICMP-paketu byly využity druhé dva bajty z nevyužitých čtyř bajtů záhlaví. Stuktura ICMP-paketu je znázorněna na obr. 2.7.



2.7 Struktura ICMP paketu (viz [1])

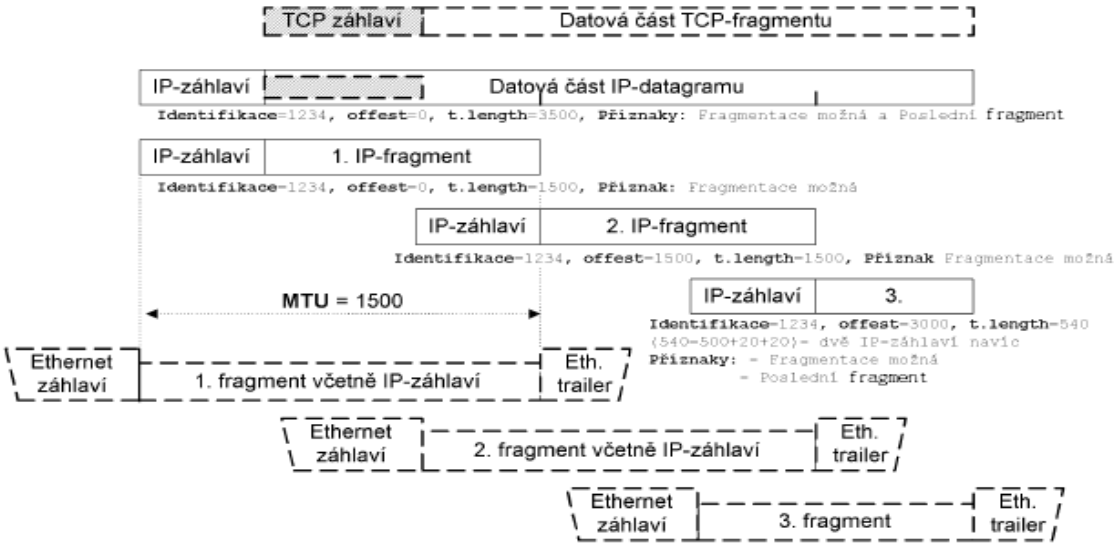
Pokud je pole MTU nulové, pak směrovač nepodporuje toto nové rozšíření. Nyní se vraťme k situaci, kdy v IP-paketu je nastaveno, že fragmentace je možná, pak směrovač dělí delší IP-datagramy na fragmenty, jejichž celková délka (total length) je menší nebo rovná MTU následující linky.

Každý IP-datagram má ve svém záhlaví svou identifikaci, kterou dědí i jeho fragmenty. Díky identifikaci příjemce pozná, ze kterých fragmentů má datagram složit. Nikdo jiný než příjemce není oprávněn z fragmentů skládat původní datagram, ani např. směrovač ze kterého vede linka s takovým MTU, do kterého by se celý datagram již vešel. Důvod je prostý, Internet negarantuje, že jednotlivé fragmenty půjdou stejnou cestou (ani negarantuje pořadí v jakém dojdou). Takže směrovač, který by se pokoušel datagram sestavit by mohl být na závadu spojení, protože fragmentů, které by šly jinou cestou by se nikdy nedočkal.

Identifikace IP-datagramů může být jednoznačná pouze v rámci jednoho protokolu vyšší vrstvy, protože záhlaví IP-datagramu obsahuje ještě pole „Protokol vyšší vrstvy". Globální identifikace může být brána jako zřetězení polí identifikace a protokol vyšší vrstvy (+ pochopitelně IP-adresa odesílatele a příjemce). Takže teoreticky mohou být za sebou odeslány dva IP-datagramy o stejné identifikaci, jeden však nese TCP-paket a druhý UDP-paket. Toto je opět méně běžná implementace. Každý fragment tvoří samostatný IP-datagram. Při fragmentaci je nutné vytvořit pro každý fragment nové IP-záhlaví. Některé údaje (jako protokol vyšší vrstvy, či IP-adresa odesílatele a příjemce) se získají ze záhlaví původního IP-datagramu.

Při fragmentaci vstupuje do hry pole „Posunutí fragmentu od počátku IP-datagramu (fragment offset)", které vyjadřuje kolik bajtů datové části původního IP-datagramu bylo vloženo do předchozích fragmentů. Pole „Celková délka IP-datagramu (total length)" obsahuje délku fragmentu, nikoliv délku původního datagramu. Aby příjemce poznal jak je původní datagram dlouhý, tak je poslední fragment opatřen příznakem „Poslední fragment". Celý mechanismus je znázorněn na obr. 2.8.

Síť nerozlišuje mezi přenosem fragmentu a přenosem celého (nefragmentovaného) IP-datagramu. Nefragmentovaný datagram je fragment s posunutím nula a příznakem „Poslední fragment". Proto se často slova IP-datagram a fragment zaměňují. Mechanismus fragmentace umožňuje i fragmentovat fragment dorazí-li na směrovač jehož odchozí linka má ještě menší MTU.



2.8 Fregmentace IP-datagramu (viz [1])

Důležité je, že každý další fragment znamená zatížení o minimálně 20 B jeho záhlaví. Pro zajímavost je na obrázku 2.8 znázorněn také TCP-paket, který je vložen do IP-paketu. Co je na tom zajímavého? Zajímavé je to, že TCP-záhlaví je obsaženo pouze v prvním IP-fragmentu. Takže pokud se provádí na směrovači filtrace IP-datagramů na základě nejen informací z IP-záhlaví, ale též na základě informací z TCP-záhlaví, pak lze filtrovat pouze první fragment a ostatní se propouští. Příjemce pak po určeném časovém intervalu zjistí, že mu schází první fragment z IP-datagramu a signalizuje to příjemci ICMP zprávou „Vypršel čas na sestavení IP-datagramu z jeho fragmentů (time to live equals 0 during reasembly)". Takže při filtraci TCP-paketů je nutné nezapomenout v protisměru filtrovat i tyto ICMP-pakety, pokud útočníkovi nechceme poskytnout informaci o tom, že se chráníme filtrací. Fragmentace je považována za jakési nutné zlo, aplikace vyžadující extrémně bezpečnou komunikaci fragmentaci zakazují. (viz [1])

3.4 Protokoly ARP a RARP

Jsem-li stanice na lokální síti a chci protokolem IP komunikovat s jinou stanicí na téže síti, pak ji v protokolu IP adresuji čtyřbajtovou IP-adresou. Pro komunikaci znám IP-adresu odesílatele (svou IP-adresu) a IP-adresu příjemce. Jsem tedy schopen sestavit IP-datagram. Jenže potíž je v tom, že tento IP-datagram musí být zabalen do linkového rámce – např. do ethernetového rámce. Abych vytvořil ethernetový rámec, tak potřebuji linkovou (6B) adresu příjemce i odesílatele. odesílatel jsem já a svou linkovou adresu znám, avšak neznám linkovou adresu příjemce. Jak takovou adresu zjistím? To řeší protokol ARP.

Protokol ARP (Address Resolution Protocol) řeší problém zjištění linkové adresy protější stanice ze znalosti její IP-adresy. Řešení je jednoduché, do LAN vyšle linkový oběžník (linková adresa FF:FF:FF:FF:FF:FF) s prosbou: „Já stanice o linkové adrese HW1, IP-adrese IP1, chci komunikovat se stanicí o IP-adrese IP2, kdo mi pomůže s nalezením linkové adresy stanice o IP-adrese IP2? Stanice IP2 takovou žádost uslyší a odpoví. V odpovědi uvede svou linkovou adresu HW2. ARP-paket je balen přímo do Ethernetu, tj. nepředchází mu žádné IP-záhlaví. Protokol ARP je vlastně samostatný, na IP nezávislý protokol. Proto jej mohou používat i jiné protokoly, které s protokoly TCP/IP nemají nic společného.

Pole typ linkového protokolu specifikuje linkový protokol používaný na LAN. Linkovému protokolu Ethernet II je vyhrazeno číslo 1. Seznam přidělených čísel je uveřejněn na http://www.iana.org. Typ síťového protokolu specifikuje typ síťového protokolu, používají se stejná čísla jako pro pole protocol v protokolu Ethernet II, tj. IP-protokol má přiděleno číslo 800_{16}. Pole HS určuje délku linkové adresy a pole PS délku síťové adresy. Standardně je tedy HS=6 a PS=4. Pole operace určuje o jakou operaci jde. Žádost (ARP request) má hodnotu 1 a odpově1 (ARP reply) má hodnotu 2. Toto pole je definováno rovněž pro reverzní překlad (protokol RARP), kdy RARP žádost používá hodnotu 3 a RARP odpově1 hodnotu 4. Pak již následuje linková adresa odesílatele, IP-adresa odesílatele, linková adresa příjemce (v dotazu vyplněna nulami) a IP-adresa příjemce.

Protokolem ARP je také možné odeslat žádost s vyplněnou IP-adresou odesílatele i příjemce a také s oběma vyplněnými linkovými adresami. Takovou žádost je možné chápat jako: „Neexistuje náhodou na LAN ještě jiná stanice, která používá stejnou IP-adresu jako já?”. V případě, že se obdrží odpověď, tak se uživateli

signalizuje zpráva „Duplicate IP address sent from Ethernet address xx:xx:xx:xx:xx:xx”. To pochopitelně signalizuje chybu v konfiguraci jedné ze stanic používajících tuto adresu. Zatímco protokol ARP slouží k překladu IP-adres na linkové adresy reverzní ARP označované jako RARP slouží k překladu linkové adresy na IP-adresu. Avšak proč takový překlad provádět? Smysl protokolu RARP je u bezdiskových stanic. Bezdisková stanice po svém zapnutí nezná nic jiného než svou linkovou adresu (tu má uloženu výrobcem v paměti ROM). Po svém zapnutí se potřebuje dozvědět svou IP-adresu. Proto do LAN vyšle oběžník s prosbou: „Já mám linkovou adresu HW1, kdo mi řekne jakou mám IP-adresu”. Na LAN pak musí být RARP-server, který jí IP-adresu přidělí a sdělí v odpovědi. Protokol RARP používá stejný formát paketu jako protokol ARP. Pouze hodnota pole operace je zvětšena o jedničku. V RARP žádosti pochopitelně není vyplněna ani IP-adresa žadatele. Protokol RARP se v praxi téměř nepoužívá, nahradil jej protokol DHCP, který je komplexnější. (viz [1])

3.5 Protokol IGMP

Protokol IGMP je podobně jako protokol ICMP služebním protokolem (podmnožinou) protokolu IP. Pakety IGMP-protokolu jsou baleny do IP-datagramů. Protokol IGMP slouží k šíření adresných oběžníků (multicasts). Nyní je aktuální protokol IGMP verze 2 podle normy RFC-2236.

2.9 Struktura IGMP protokolu (viz [1])

Hodnota (šestnáctkově)	Význam
11	Dotaz směrovače: "Jsou na LAN ještě nějací členové" *(Membership query)*
16	Požadavek na členství ve skupině *(Membership report)*
17	Opuštění skupiny *(Leave group)*
12	Požadavek IGMP v1 na členství ve skupině *(Version 1 membership report)*

2.10 Pole typ nabývá těchto hodnot (viz [1])

Pole MRT (Maximum response time) se používá pouze v dotazu směrovače a specifikuje v desetinách sekundy čas do kterého musí členové skupiny opakovat požadavky na členství ve skupině. Ve všech ostatních případech má pole MRT hodnotu 0. Kontrolní součet se počítá stejně jako u protokolu ICMP. Pole IP-adresa adresného oběžníku je nulové u všeobecného dotazu, v ostatních případech specifikuje konkrétní IP-adresu adresného oběžníku. IP-adresy adresných oběžníků jsou v intervalu 224.0.0.0 až 239.255.255.255. Interval 224.0.0.0 až 224.0.0.255 je určen pro vyhrazené účely na LAN. Jelikož jsou oběžníky s těmito adresami určeny výhradně pro LAN, tak mívají v položce TTL nastavenu hodnotu 1.

Všechny IGMP pakety mají v IP-záhlaví nastavenu položku TTL=1. Pakety protokolu IGMP verze 2 používají volbu (rozšíření) IP-záhlaví „Upozornění pro směrovač (IP Router Alert Option)". Jádrem Internetu je tzv. Mbone (zkráceno z Mulicast Backbone), kde je zabezpečeno šíření adresných oběžníků. Např. internetová rozhlasová stanice šíří svá data pomocí adresných oběžníků. Kdyby se oběžníky šířily nekontrolovaně lavinovitě, tak by se mohla data postupně duplikovat.

Protokol IGMP řeší šíření adresných oběžníků v rámci LAN. Představme si situaci, kdy jsme na LAN a některé směrovače přijímají adresné oběžníky z MBONE a řeší otázku, zdali je mají šířit dále na LAN. Obecně, pokud žádný počítač na LAN adresné oběžníky nepotřebuje, pak je zbytečné je šířit – pouze by se zvětšilo zatížení LAN. Jsme tedy v situaci, kdy některé směrovače na LAN mohou LAN zásobovat adresnými oběžníky, ale nečiní tak, protože adresné oběžníky na LAN nejsou vyžadovány.

Pro každou IP-adresu adresného oběžníku se na LAN definuje tzv. skupina členů adresného oběžníku. Směrovače udržují seznam skupin. V případě, že se nějaký počítač na LAN přihlásí do konkrétní skupiny, pak směrovače začnou daný oběžník na LAN šířit. V případě, že poslední člen skupinu opustí, pak se šíření adresného oběžníku na LAN zastaví. Čili existence skupiny znamená šíření oběžníků. Přitom není důležité, kolik má skupina členů, ale jestli má alespoň jednoho člena nebo nikoliv. (viz [1])

3.6 Všeobecné oběžníky a linkový protokol

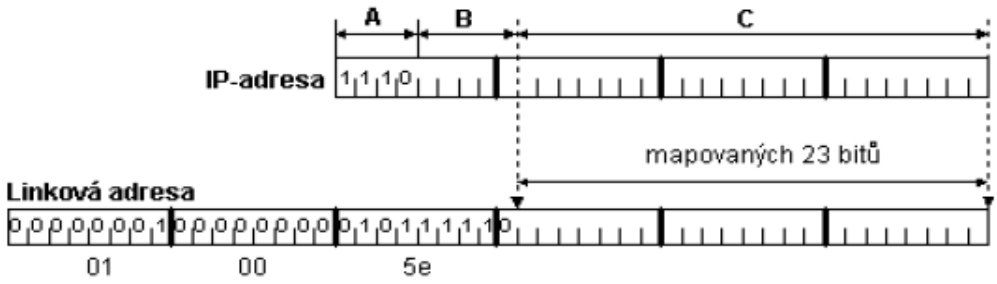
Zatím jsme popisovali odesílání všeobecných oběžníků, ale problém na LAN spočívá v určení linkové adresy jejich příjemce. Protokol ARP určil jednoznačný vztah mezi jednoznačnou IP-adresou příjemce (unicast) a linkovou adresou příjemce. To je možné tehdy, když mezi IP-adresami a linkovými adresami existuje jednoznačný vztah. Tento vztah se anglicky nazývá mapping, který se česky (asi ne zcela správně) označuje za mapování IP-adres na linkové adresy. Jiným případem na LAN je všeobecný oběžník (broadcast), který se posílá všem systémům na LAN. Pro tyto účely slouží linkovému protokolu všeobecný oběžník, který pro Ethernet, FDDI apod. je FF:FF:FF:FF:FF:FF.

Jenže jak to udělat na LAN v případě adresného oběžníku (multicast), který není určen jednou adresátovi na LAN ani všem systémům na LAN, nýbrž několika konkrétním adresátům? V čem je problém? Příjemce za normálních okolností zpracovává pouze rámce, které jsou všeobecnými oběžníky nebo adresovány příjemcovou linkovou adresou. (Je možné síťovou kartu realizující síťové rozhraní přepnout do tzv. promiskuitního módu, kdy přijímá vše, ale tento případ není považován za běžný.)

Linkové protokoly umožňují též linkové adresné oběžníky (multicast). Jsou to takové linkové adresy, kde nejnižší bit prvního bajtu linkové adresy je nastaven na 1, tj.

všeobecný linkový oběžník je zvláštním případem takovéhoto adresného oběžníku. Jenže jak mapovat IP-adresu adresného oběžníku na linkový adresný oběžník. Není to však tak jednoduché jak to vypadá na první pohled. Šestibajtová linková adresa se skládá ze tří bajtů specifikujících výrobce a tří bajtů čísla karty v rámci výrobce.

IANA (nejvyšší autorita Internetu) se nechala zaregistrovat jako fiktivní výrobce síťových karet a obdržela pro sebe identifikaci 00:00:5E. První polovinu těchto adres použila pro mapování adresných IP oběžníků na adresné linkové oběžníky (viz obr. 2.11). Bohužel tato polovina má pouze 23 bitů, takže mapování nemůže být jednoznačné.



2.11 Mapování adresných oběžníků na linkové adresy (viz [1])

První bajt linkové adresy musí mít nastaven nejnižší bit na jedničku, protože se jedná o adresný linkový oběžník. Takže prefix ve skutečnosti nebude 00:00:5E ale 01:00:5E. Část A IP-adresy specifikuje adresný oběžník, je tedy vždy konstantní. Část B není mapována. Takže pokud se dva adresné oběžníky liší pouze v části B, pak jsou mapovány na stejnou linkovou adresu. Např. IP-adresy 224.0.1.1, 224.128.1.1 a 225.0.1.1 jsou mapovány vždy na 01:00:5E:00:01:01.

Seznam přijímaných adresných linkových oběžníků zahrnuje adresu 224.0.0.1 a s každým oběžníkem i všechny adresné oběžníky, které vznikly díky nejednoznačnosti mapování. Nadbytečné oběžníky musí odfiltrovat IP-protokol. Některé implementace softwaru přepnou síťovou kartu do promiskuitního módu pro interval všech adresných oběžníků a vše ponechají na IP-protokolu, to však zbytečně zvyšuje zatížení operačního systému. (viz [1])

3.7 IP-adresa

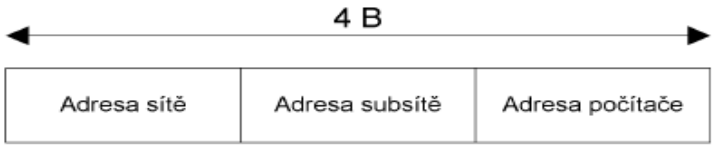
Protokol IP verze 4 používá IP-adresu o délce čtyři bajty. IP-adresa adresuje jednoznačně síťové rozhraní systému. Anglicky se takováto jednoznačná adresa nazývá unicast. Pokud má systém více síťových karet (více síťových rozhraní) a na všech je provozován protokol IP, pak každé rozhraní má svou IP-adresu. Je to podobné jako s adresou domu. Pokud má dům vchod ze dvou ulic, pak má i dům dvě adresy.

Je možná i opačná varianta, kdy na jedné síťové kartě (fyzicky jednom síťovém rozhraní) podporujeme několik IP-adres. První adresa se obvykle nazývá primární a další adresy pak sekundární nebo aliasy. Využití sekundárních IP-adres je běžné např. pro WWW-servery, kdy na jednom počítači běží WWW servery několika firem a každý se má tvářit jako samostatný WWW-server. V praxi se však využívání sekundárních IP-adres pro WWW-servery považuje za plýtvání – používají se tzv. virtuální WWW-servery, kdy mnoha WWW-serverům stačí jedna společná IP-adresa. Specifikace serveru se pak provádí na aplikační úrovni v protokolu http (pomocí hlavičky host). Jelikož má většina počítačů jedno síťové rozhraní, tak se přeneseně místo IP-adresa rozhraní říká IP-adresa počítače.

IP-adresa je tvořena čtyřmi bajty. IP-adresa se zapisuje notací, kde jednotlivé bajty se mezi sebou oddělují tečkou. Rozeznáváme: Dvojkovou notaci, kde jednotlivé bity každého bajtu se vyjádří jako dvojkové číslo, např.: 10101010.01010101.11111111.11111000 nebo desítkovou notaci – čtyři osmiciferná dvojková čísla se převedou do desítkové soustavy např: 170.85.255.248 a v poslední řadě šestnáctkovou notaci – jednotlivé bajty IP-adresy se vyjádří šestnáctkově (hexadecimálně), např.: AA.55.FF.F8

IP-adresa se skládá ze dvou částí. Adresy (lokální) sítě a adresy počítače v (lokální) síti. Problém je v tom jak zjistit, která část IP-adresy je adresou sítě a která adresou počítače. Není ani zcela jasné co to znamená slovo síť, protože jeho význam se postupně měnil a kromě slova síť se zavedly pojmy subsíť a supersíť.

V roce 1993 vyšly normy RFC-1517 až 1520. Tyto normy od základu změnily pohled na slovo síť jak je chápáno v Internetu. Přestalo se na sítě hledět přes třídy, ale výhradně přes síťové masky. Jenže ono úplně nejde abstrahovat od sítí, takže v podstatě dělení IP-adresy na adresu sítě a adresu počítače zůstalo, pouze část IP-adresy dříve odpovídající adrese počítače se rozdělila na dvě části: na adresu subsítě a adresu počítače.



3.12 Struktura IP-adresy (viz [1])

Z hlediska síťové masky je adresa sítě i subsítě jeden celek. Ta část IP-adresy, kde jsou v masce jedničky je prostě síť. Jenže nyní dochází k nejednoznačnosti v terminologii. Jednou slovo síť označuje ve smyslu třídy (A, B, nebo C) a podruhé je sítí myšleno obecně část IP-adresy, kde v odpovídající masce jsou jedničky. Pokud na čas zapomeneme na třídy a budeme používat libovolné masky, pak už nestačí mluvit o síti např. 192.168.0.0 ale vždy k ní musíme dopsat masku, abychom vyjádřili co touto sítí míníme. Pokud bychom uvažovali, pak se pro tuto síť použije vždy maska 255.255.255.0, protože se jedná o síť třídy C. Maska 255.255.255.0 pro síť 192.168.0.0 se nazývá standardní síťovou maskou.

Kromě subsítí se používají i supersítě, u kterých je počet jedniček masky menší než u standardní síťové masky. Jako příklad je na obr. 4.2 uvedeno dělení sítě 192.168.0.0 na subsítě s různými maskami (standardní maska je zobrazena tučně).

Adresy s maskami majícími méně jedniček než je standardní maska se nazývají adresy supersítí (na obrázku nahoře) a adresy s maskami o více jedničkách než má standardní maska se nazývají adresy subsítí (dolní část obrázku). Jelikož dvojkové vyjádření síťové masky je tvořeno zprava souvislou řadou jedniček, tak se místo vyjádření „síť 192.168.0.0 s maskou 255.255.255.252" častěji zkracuje na 192.168.0.0/30, kde číslo 30 vyjadřuje počet jedniček masky. (viz [1])

Maska	Počet jedniček v masce (zleva)	Síť je tvořena intervalem IP-adres	Zkrácený zápis sítě (včetně masky)
255.248.0.0	13	192.168.0.0 až 192.175.255.255	192.168.0.0/13
255.252.0.0	14	192.168.0.0 až 192.171.255.255	192.168.0.0/14
255.254.0.0	15	192.168.0.0 až 192.169.255.255	192.168.0.0/15
255.255.0.0	16	192.168.0.0 až 192.168.255.255	192.168.0.0/16
255.255.248.0	21	192.168.0.0 až 192.168.7.255	192.168.0.0/21
255.255.252.0	22	192.168.0.0 až 192.168.3.255	192.168.0.0/22
255.255.254.0	23	192.168.0.0 až 192.168.1.255	192.168.0.0/23
255.255.255.0	24	192.168.0.0 až 192.168.0.255	192.168.0.0/24
255.255.255.128	25	192.168.0.0 až 192.168.0.127	192.168.0.0/25
255.255.255.192	26	192.168.0.0 až 192.168.0.63	192.168.0.0/26
255.255.255.224	27	192.168.0.0 až 192.168.0.31	192.168.0.0/27
255.255.255.240	28	192.168.0.0 až 192.168.0.15	192.168.0.0/28
255.255.255.248	29	192.168.0.0 až 192.168.0.7	192.168.0.0/29
255.255.255.252	30	192.168.0.0 až 192.168.0.3	192.168.0.0/30
255.255.255.254	31	192.168.0.0 až 192.168.0.1 Pozor, takováto síť je nesmysl, protože má jen dvě IP-adresy, tedy adresu sítě samotné a adresu oběžníku, nedostávají se už adresy pro počítače na této síti.	192.168.0.0/31
255.255.255.255	32	Adresa samostatného počítače (host address) 192.168.0.0	192.168.0.0/32

3.13 Příklad dělení sítě 192.168.0.0 na subsítě (viz [1])

3.8 Protokol TCP

Protokol TCP je proti protokolu IP protokolem vyšší vrstvy. První otázkou každého začátečníka vždy je: „Proč jsou třeba dva protokoly". Zatímco protokol IP přepravuje data mezi libovolnými počítači v Internetu, tak protokol TCP dopravuje data mezi dvěma konkrétními aplikacemi běžícími na těchto počítačích. Pro dopravu dat mezi počítači se využívá protokol IP. Protokol IP adresuje IP-adresou pouze síťové rozhraní počítače. Pokud bychom použili přirovnání k běžnému poštovnímu styku, pak IP-adresa odpovídá adrese domu a port (adresa v protokolu TCP) pak odpovídá jménu konkrétního obyvatele domu. Protokol TCP je spojovanou službou (connection oriented), tj. službou která mezi dvěma aplikacemi naváže spojení – vytvoří na dobu spojení virtuální okruh. Tento okruh je plně duplexní (data se přenášejí současně na sobě nezávisle oběma směry). Přenášené bajty jsou číslovány. Ztracená nebo poškozená data jsou znovu vyžádána. Integrita přenášených dat je zabezpečena kontrolním součtem. Jinými slovy aplikace používající protokol TCP si nemusí dělat starosti s tím, zdali náhodou nebyla nějaká data během přenosu ztracena nebo díky chybě přenosu modifikována. Toto zabezpečení je účinné pouze proti poruchám technických prostředků. Neklade si za cíl zabezpečit data proti inteligentním útočníkům, kteří mohou data modifikovat a současně také přepočítat kontrolní součet. Ochranou přenášených dat proti takovýmto cíleným útokům se v rodině protokolů TCP/IP zabývají např. protokoly SSL, S/MIME.

Konce spojení ("odesílatel" a „adresát") jsou určeny tzv. číslem portu. Toto číslo je dvojbajtové, takže může nabývat hodnot 0 až 65535. U čísel portů se často vyjadřuje okolnost, že se jedná o porty protokolu TCP tím, že se za číslo napíše lomítko a název protokolu (TCP). Pro protokol UDP je jiná sada portů než pro protokol TCP (též 0 až 65535), tj. např. port 53/TCP nemá nic společného s portem 53/UDP. Cílová aplikace je v Internetu adresována (jednoznačně určena) IP-adresou, číslem portu a použitým protokolem (TCP nebo UDP). Protokol IP dopraví IP-datagram na konkrétní počítač. Na tomto počítači běží jednotlivé aplikace. Podle čísla cílového portu operační systém pozná které aplikaci má TCP segment doručit.

Základní jednotkou přenosu v protokolu TCP je TCP segment. Někdy se také říká TCP paket. Proč segment? Aplikace běžící na jednom počítači posílá protokolem TCP data aplikaci běžící na jiném počítači. Aplikace potřebuje přenést např. soubor velký 2 GB. Jelikož TCP segmenty jsou baleny do IP datagramů, který má pole délka

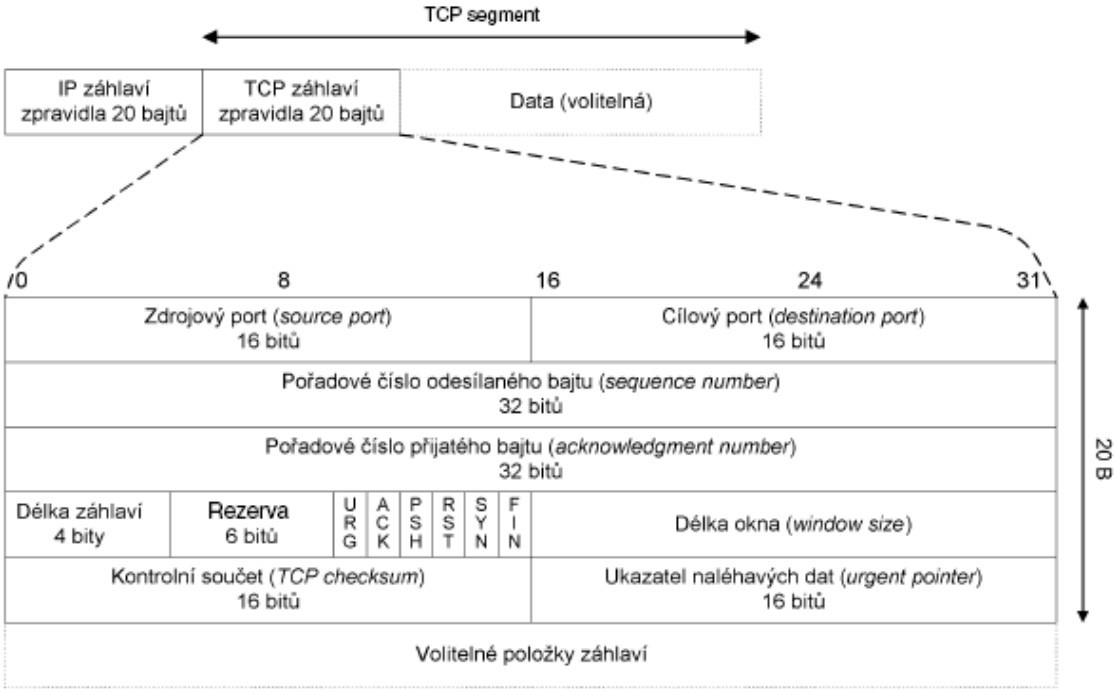
dlouhé 16 bitů, tak TCP segment může být dlouhý maximálně 65535 minus délka TCP-záhlaví. Přenášené 2 GB dat musí být rozděleny na segmenty, které se vkládají do TCP paketu – přeneseně se pak místo TCP paket říká TCP segment. TCP segment se vkládá do IP-datagramu. IP-datagram se vkládá do linkového rámce. Použije-li se příliš velký TCP-segment, který se celý vloží do velkého IP-datagramu, který je větší než maximální velikost přenášeného linkového rámce (MTU), pak IP protokol musí provést fragmentaci IP-datagramu. Fragmentace zvyšuje režii, proto je cílem vytvářet segmenty takové velikosti, aby fragmentace nebyla nutná.

Zdrojový port (source port) je port odesílatele TCP segmentu, cílový port (destination port) je portem adresáta TCP segmentu. Pětice: zdrojový port, cílový port, zdrojová IP-adresa, cílová IP-adresa a protokol (TCP) jednoznačně identifikuje v daném okamžiku spojení v Internetu. TCP segment je část z toku dat tekoucích od odesílatele k příjemci. Pořadové číslo odesílaného bajtu je pořadové číslo prvního bajtu TCP segmentu v toku dat od odesílatele k příjemci (TCP segment nese bajty od pořadového čísla odesílaného bajtu až do délky segmentu). Tok dat v opačném směru má samostatné (jiné) číslování svých dat. Jelikož pořadové číslo odesílaného bajtu je 32 bitů dlouhé, tak po dosažení hodnoty 2^{32}-1 nabude cyklicky opět hodnoty 0. Číslování obecně nezačíná od nuly (ani od nějaké určené konstanty), ale číslování by mělo začínat od náhodně zvoleného čísla. Vždy když je nastaven příznak SYN, tak operační systém odesílatele začíná znovu číslovat, tj. vygeneruje startovací pořadové číslo odesílaného bajtu, tzv. ISN (Initial Sequence Number). Naopak pořadové číslo přijatého bajtu vyjadřuje číslo následujícího bajtu, který je příjemce připraven přijmout, tj. příjemce potvrzuje, že správně přijal vše až do pořadového čísla přijatého bajtu mínus jedna.

Délka záhlaví vyjadřuje délku záhlaví TCP segmentu v násobcích 32 bitů (4 bajtů) – podobně jako u IP-záhlaví. Délka okna vyjadřuje přírůstek pořadového čísla přijatého bajtu, který bude příjemcem ještě akceptován. Ukazatel naléhavých dat může být nastaven pouze v případě, že je nastaven příznak URG. Přičte-li se tento ukazatel k pořadovému číslu odesílaného bajtu, pak ukazuje na konec úseku naléhavých dat. Odesílatel si přeje, aby příjemce tato naléhavá data přednostně zpracoval.

Příkazem ABORT signalizuje uživatel žádost o zrušení procesu. Uživatel například odeslal na server velké množství dat, která čekají ve vyrovnávací paměti serveru na zpracování. Pokud uživatel chce proces ukončit bez čekání na ukončení zpracování dat a nezpracovaná data zahodit, pak vyšle příkaz ABORT. Pokud by server zpracovával veškerá data sekvenčně, pak by se příkaz ABORT vykonal až po

zpracování všech dat. Uživatel však chce proces ukončit okamžitě. V TCP segmentu nesoucím příkaz ABORT se nastaví příznak URG a vyplní se ukazatel naléhavých dat ukazující na příkaz ABORT.

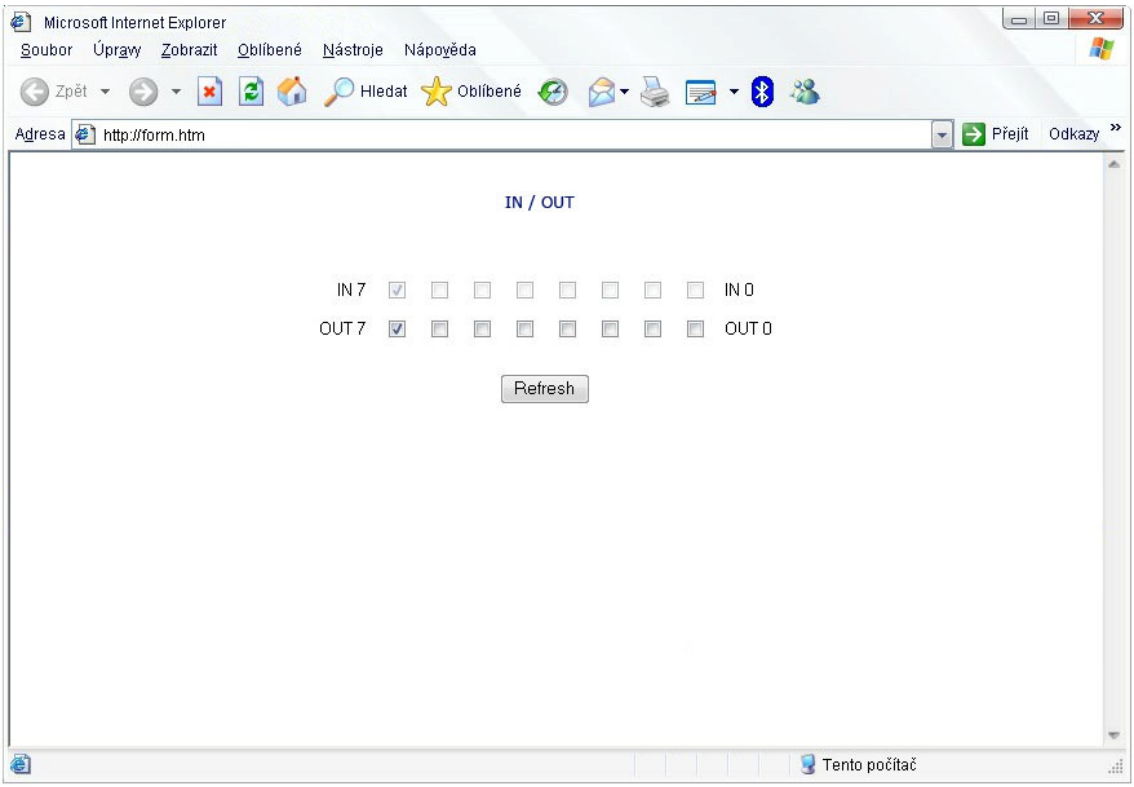


3.14 TCP segment (viz [1])

Server podle příznaku URG zjistí, že TCP segment obsahuje naléhavá data, přičtením ukazatele naléhavých dat k pořadovému číslu odeslaného bajtu zjistí konec naléhavých dat. Nyní začne procházet vyrovnávací paměť od konce naléhavých dat směrem k počátku vyrovnávací paměti až narazí na bajt obsahující <IAC>. Nyní již má zjištěna celá naléhavá data a může je začít interpretovat, tj. v našem případě zrušit proces. (viz [1])

4 Mikroprocesorové rozhraní

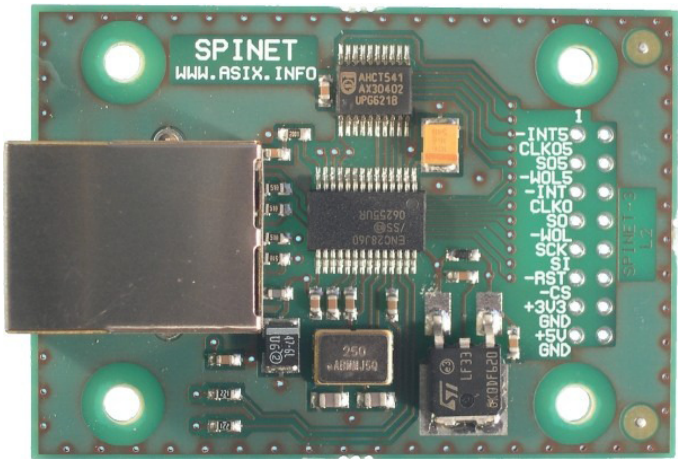
Základem všeho je ethernetový modul, který řeší komunikaci po fyzickou vrstvu. Dále je potřeba mikroprocesor, který bude řešit komunikaci protokolu TCP/IP a ovládat vstupy a výstupy. Toto jsou dva základní kameny, které budou jednotlivě popsány v následujících kapitolách. V poslední řadě ethernetový a mikroprocesorový modul propojíme a přes rozhraní ethernet připojíme k počítači. Jelikož zařízení bude mít svoji vlastní IP adresu, můžeme se na zařízení připojit pomocí webového prohlížeče využitím IP adresy. Při programováním mikroprocesoru se do něj uloží jednoduché webové stránky v podobě strojového kódu. Tyto stránky budou zobrazeny po dotazu webového prohlížeče na IP adresu zařízení. Pomocí označovacích políček, můžeme tak ovládat jednotlivé vstupy a výstupy. Označené políčko tak reprezentuje logickou jedna na daným vstupu či výstupu, tzn. 5V. viz obr. 4.1



4.1 Ukázka webových stránek pro ovládání

4.1 Ethernetový modul

SPINET je modul založený na Ethernetovém kontroléru ENC28J60 firmy Microchip, který umožňuje připojení uživatelské aplikace k síti Ethernet, aniž by byla nutná předchozí znalost problematiky fyzické vrstvy tohoto rozhraní.



4.1 Spinet (viz [2])

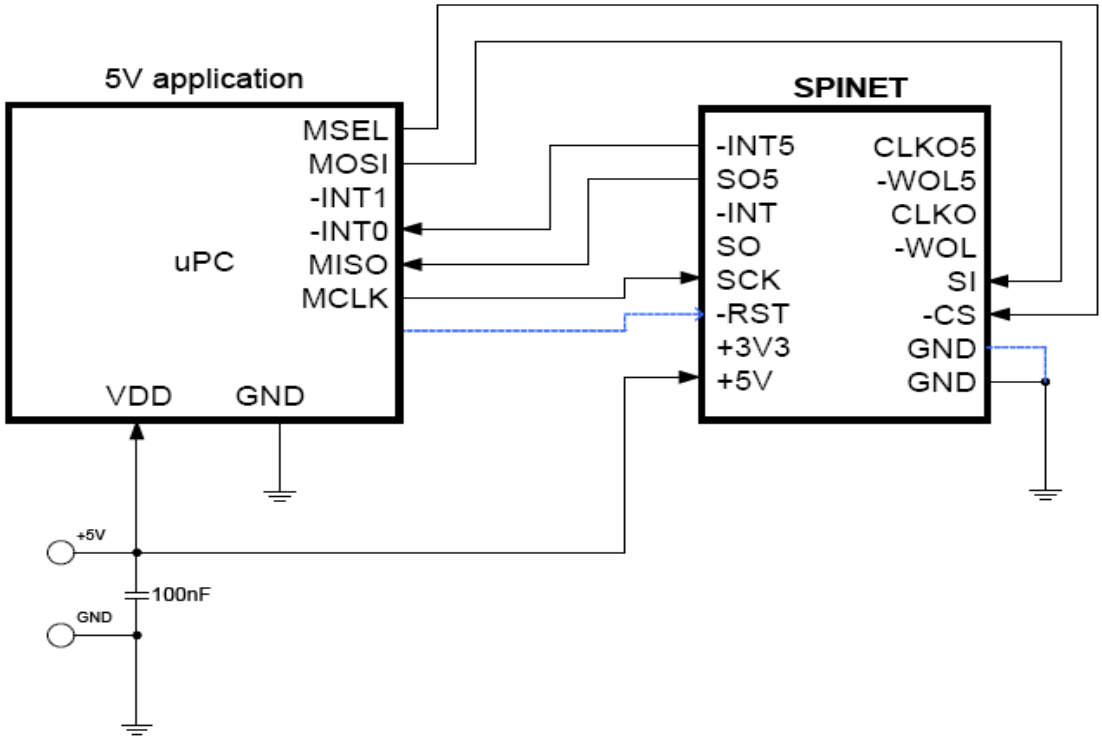
SPINET obsahuje fyzickou vrstvu 10BASE-T (PHY), včetně RJ45 konektoru s galvanickým oddělením, a řízení přístupu k médiu (MAC). Modul umožňuje poloduplexní i duplexní přenos dat rychlostí 10 Mb/s a je vybaven 8 kB dvouportové SRAM pro příchozí a odchozí data. Kromě jiného je integrována hardwarová podpora pro kopírování bloků paměti, výpočet kontrolních součtů v aritmetice s jedničkovým doplňkem (IP checksum) a CRC.

Pin	Potisk	Typ	Popis	3.3 V	5 V
1	-INT5	OUT	Interrupt pin		✔
2	CLKO5	OUT	Výstup uživatelských hodin		✔
3	SO5	OUT	Sériový výstup dat		✔
4	-WOL5	OUT	Wake On LAN *)		✔
5	-INT	OUT	Interrupt pin	✔	
6	CLKO	OUT	Výstup uživatelských hodin	✔	
7	SO	OUT	Sériový výstup dat	✔	
8	-WOL	OUT	Wake On LAN *)	✔	
9	SCK	IN	Vstup sériových hodin	✔	✔
10	SI	IN	Sériový vstup dat	✔	✔
11	-RST	IN	Reset obvodu	✔	✔
12	-CS	IN	Výběr čipu	✔	✔
13	+3V3	PWR	Napájení +3,3 V	✔	
14	GND	PWR	Zem	✔	✔
15	+5V	PWR	Napájení +5 V		✔
16	GND	PWR	Zem	✔	✔

4.2 Popis vstupů, výstupů (viz [2])

Modul vyžaduje pouze jedno napájecí napětí, buď +3,3 V nebo +5 V přivedené na příslušný pin 13 nebo 15. Všechny vstupy modulu jsou LVTTL a jsou 5 V tolerantní. Výstupy, jejichž jméno začíná znakem "-", jsou aktivní v log. 0, ostatní signály jsou aktivní v log. 1. Všechny výstupy jsou zdvojené. Výstupy na pinech 5 až 8 vedou přímo z obvodu ENC28J60 a mají 3,3 V úrovně. Při použití 5 V napájení jsou k dispozici výstupy na pinech 1 až 4 s 5 V úrovněmi. Dle napájení aplikace zapojte výstupní signály s požadovanou úrovní, ostatní zůstanou nezapojené, (viz obr. 4.2)

V případě použití modulu v 5 V aplikaci jsou výstupní signály vedeny přes převodník úrovní, zatímco vstupní jsou připojeny přímo k obvodu ENC28J60 (který je 5 V tolerantní). Pro napájení obvodu ENC28J60 je použit 3,3 V napěťový regulátor na modulu.

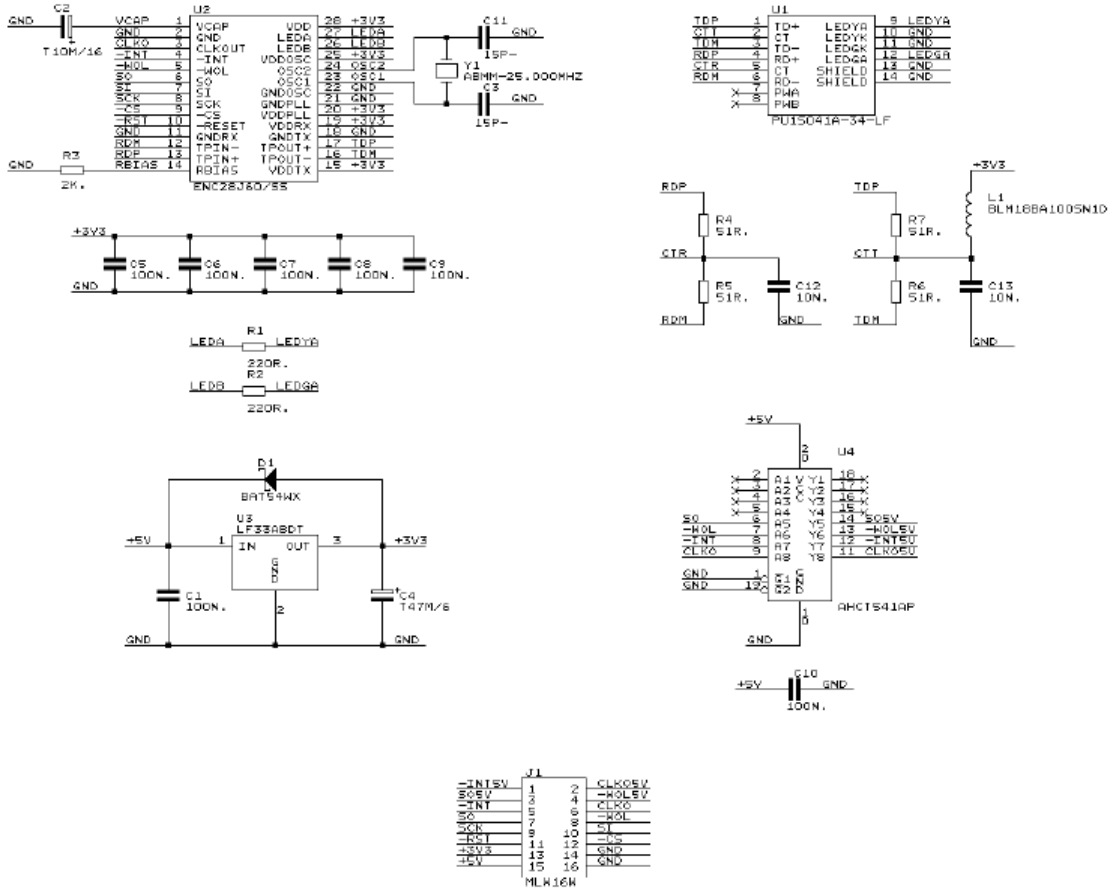


4.3 Zapojení k 5V aplikaci (viz [2])

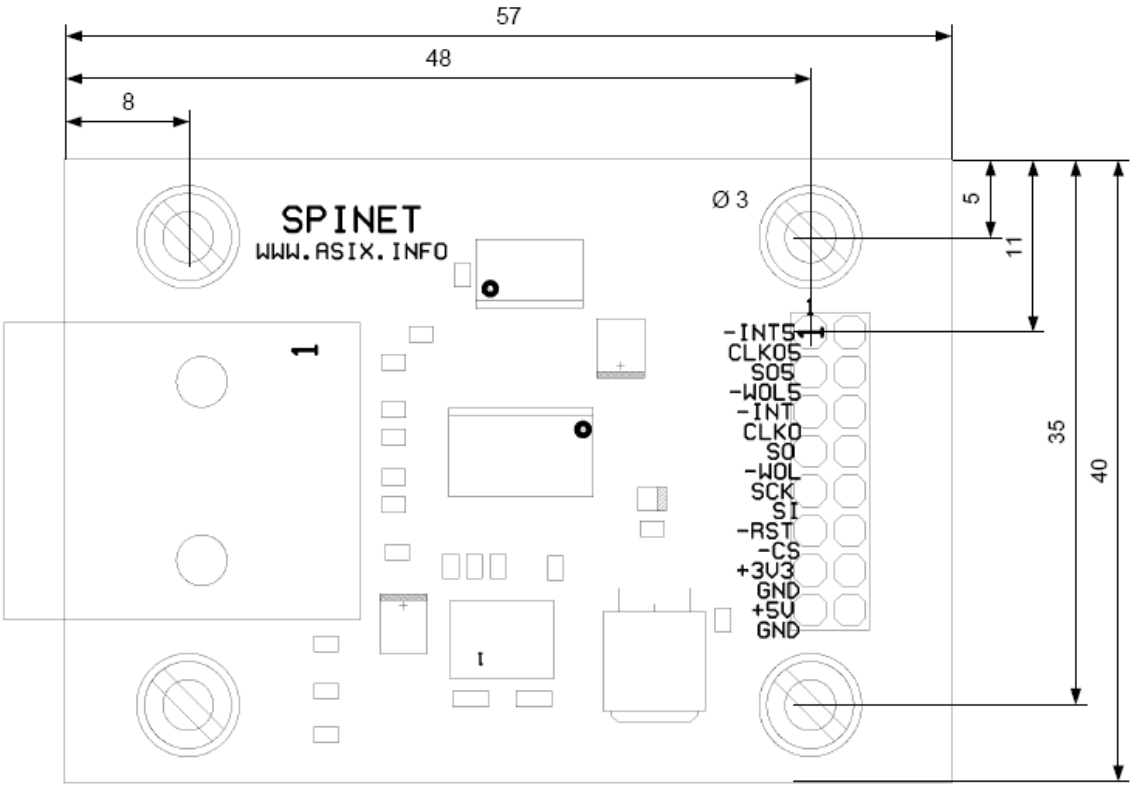
Při konfiguraci obvodu je nutné modulu přidělit vhodnou MAC adresu. Pokud není organizací IEEE přidělena vlastní rozsah adres (OUI), doporučuje se použít adresu z tzv. locally administered rozsahu, tedy takovou, která má nastaven bit 1 (druhý nejnižší) v prvním bytu, např. 02-00-00-00-00-78.

Modul řeší autonomně komunikaci po úroveň MAC vrstvy sítě Ethernet, přenáší tedy Ethernetové rámce. Pokud je třeba použít vyšší komunikační protokoly (ARP, IP,

ICMP, UDP, TCP, HTTP...), je třeba je implementovat v aplikaci (ve firmware připojeného mikrokontroléru). (viz [2])



4.4 Schéma modulu (viz [2])



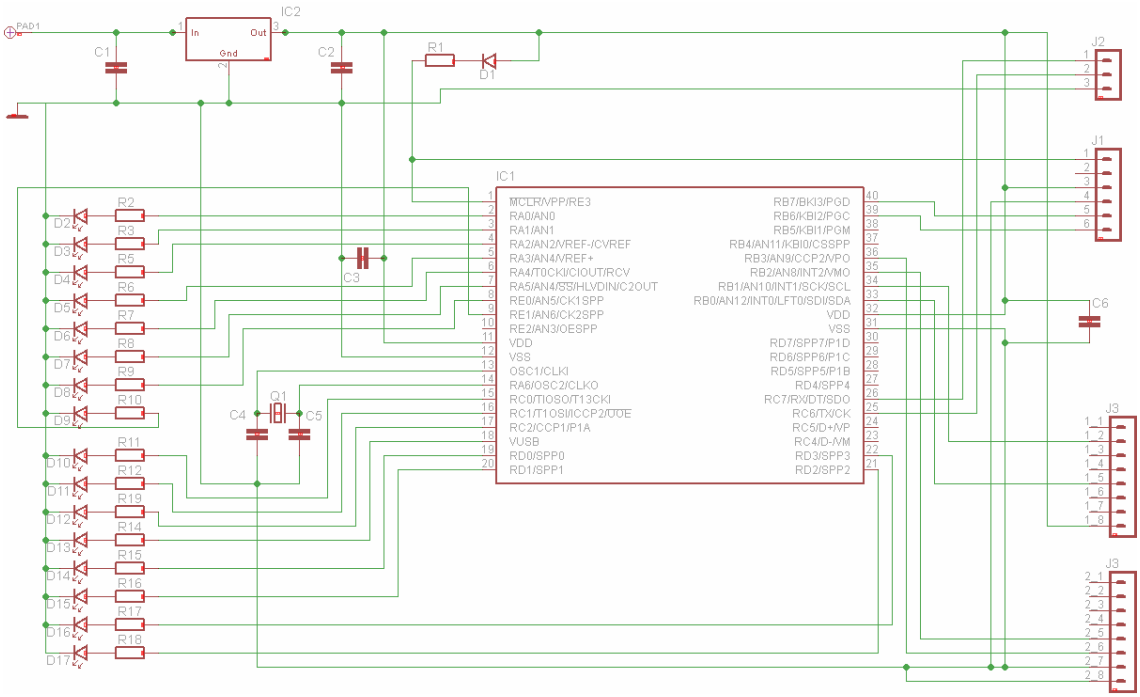
4.5. Mechanické provedení (viz [2])

Doporučené hodnoty pro 5V aplikaci

Napájecí napětí, 5 V	VCC5	min. 4,5 V	max. 5,5 V
Spotřeba, +5V	ICC5		max. 200 mA
Vstupní napětí log.1	VIH	min. 2,25 V	max. 5,5 V
Vstupní napětí log.0	VIL		max. 1 V
Výstupní napětí log.1, 5 V	VOH5	min. 4,4 V	
Výstupní napětí log.0, 5 V	VOL5		max. 0,1 V
Výstupní proud signálů, 5 V	Iout5		max. 25 mA

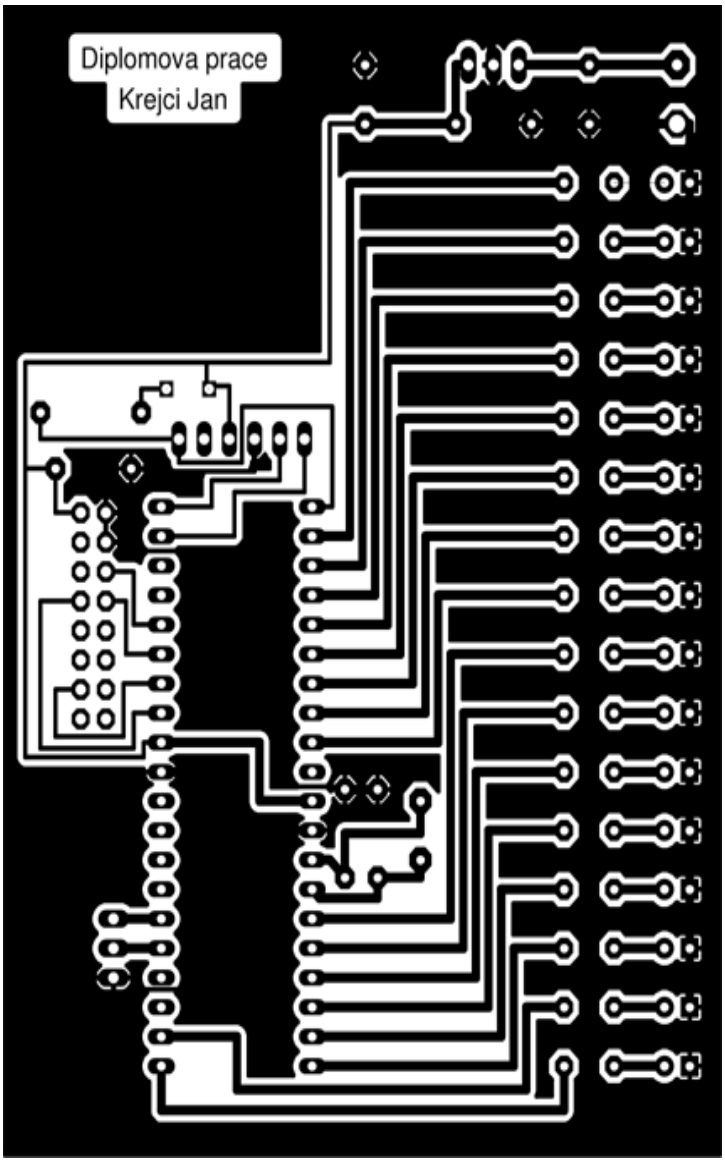
4.6. Technická specifikace (viz [2])

4.2 Schéma modulu

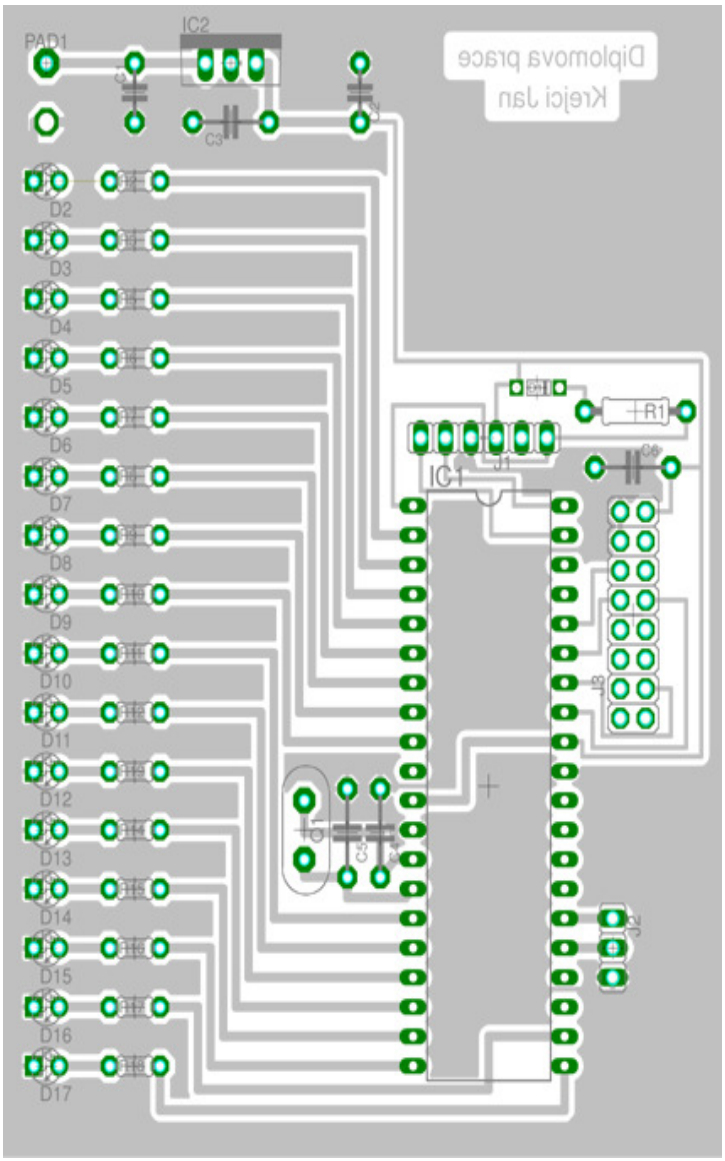


4.7. Schéma modulu

Základní část modulu tvoří naprogramovaný mikroprocesor IC1:PIC18F458, Mikroprocesor řeší protokol TCP/IP a zároveň ovládá vstupy a výstupy připojeného zařízení. V tomto případě se bude rozsvěcováním a zhasínáním LED diod demonstrovat funkčnost ovládání vstupů a výstupů. V praxi se místo LED diod mohou využít vstupy a výstupy na jakékoli elektronické zařízení a tím umožnit jeho ovládání. Mikroprocesor je napájen stabilizátorem IC2, který má u sebe zapojeny kondenzátory C1 a C2 proti rozkmitání. Kvůli rozkmitání napájecího napětí mikroprocesoru jsou použity kondenzátory C3 a C6. K mikroprocesoru je připojen krystal Q1 s frekvencí 10 MHz. Procesor si tuto frekvenci násobí, běží tedy na 40 MHz. Odpor R1 a dioda D1 slouží pouze k oddělení programovacího napájení od napájení mikroprocesoru. Konektorem J3 je k zapojení připojen ethernetový modul firmy ASIX, který má na starosti fyzickou vrstvu komunikace s počítačem. Na vstupy a výstupy jsou připojeny LED diody, jak je vidět na schématu. Vstupy a výstupy jsou pěti-voltové to znamená, že se musí vhodně zvolit odpory R2 – R19.



4.8 Deska plošných spojů



4.9 Rozmístění součástek

Seznam součástek:

R1	10K
R2-R17	250 Ω
C1-C3,C6	100nF
C4,C5	22pF
D1	1N4148
IC1	PIC18F458
IC2	7805
Q1	10MHz
D2-D17	LED dioda,průměr 4mm

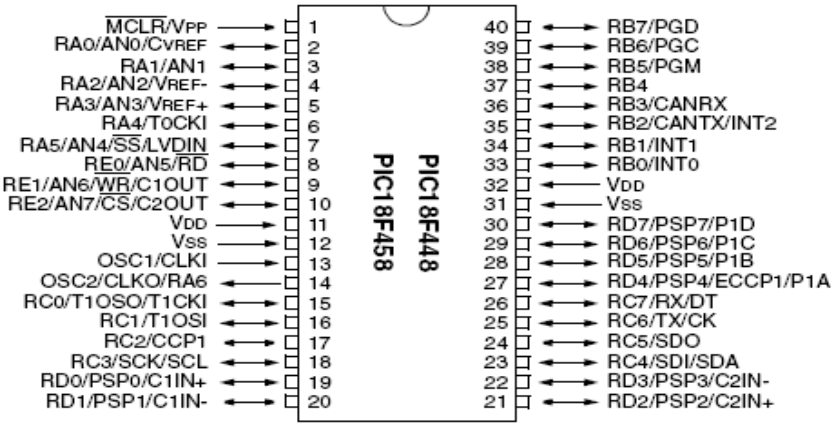
4.3 Mikroprocesor PIC18F458 – I/P

Mikroprocesor PIC18F458 patří do rodiny PIC18F. Jsou to univerzální 8-mi bitové jednočipové mikrokontrolery. Všechny tyto řadiče jsou vyrobeny technologií CMOS a jsou založeny na rozšířené architektuře RISC (Reduced Instruction Set). Mají oddělenou programovou a datovou paměť (Harvardská architektura). Vnitřní systém redukuje nutnost připojení externích obvodů na minimum, čímž zlevňuje konečné aplikace. (viz [3])

4.3.1 Popis

- Vysoký výkon RISC procesoru
- Provozní frekvence 40Mhz
- 32KB paměť programu – flash
- 1536Byte datová paměť (RAM)
- 256Byte datová paměť EEPROM
- Schopnost přerušení (až 21 zdrojů)
- Přímý, nepřímý a poměrný adresový režim
- Po zapnutí RESET (POR)
- Časovač zapnutí (PWRT) a časovač zapnutí oscilátoru (OST)
- Watchdog (WDT)
- Programovatelná ochrana kódu
- Spící mód
- Výběr typu oscilátoru
- Nízký odběr, vysoká rychlost CMOS FLASH / EEPROM technologie
- Úplně statický design
- Sériové programování v zapojení (ICSP) pomocí dvou vývodů
- Interface typ:AUSART, CAN, I2C, PSP, SPI
- Napájení Min:4.2V
- Napájení Max:5.5V
- Pouzdro DIP
- Provozní teplota Max:85°C
- Provozní teplota Min:-40°C

4.3.2 Rozložení vývodů

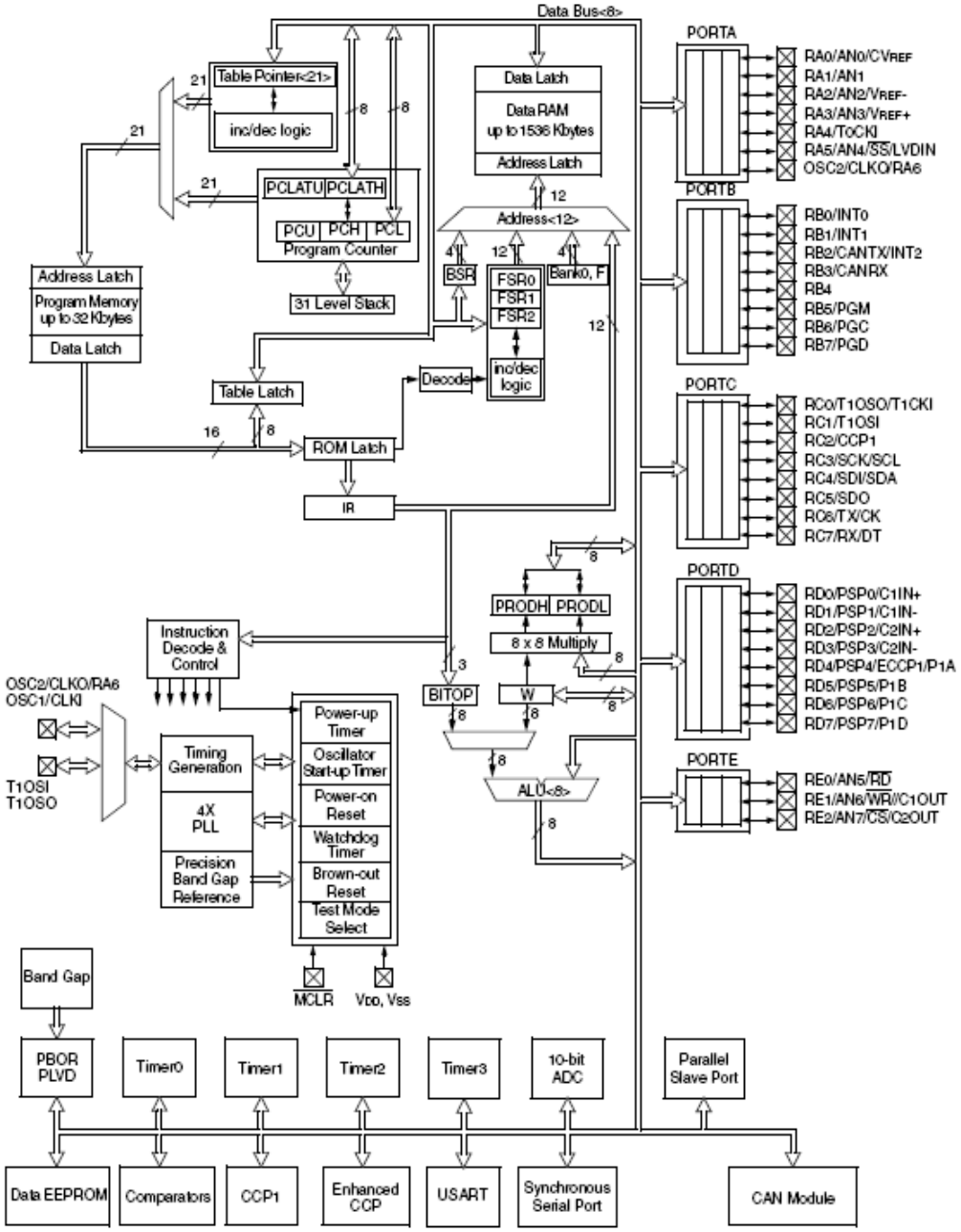


4.10 Rozložení vývodů (viz [3])

vývod	pin	typ I/O/P	provedení	popis
				PORT A je obousměrný vstupně/výstupní port
RA0/AN0	2	I/O	TTL	analogový vstup 0
RA1/AN1	3	I/O	TTL	analogový vstup 1
RA2/AN2/Vref-	4	I/O	TTL	analogový vstup 2/- reference
RA3/AN3/Vref+	5	I/O	TTL	analogový vstup 3/+ reference
RA4/TOCKI	6	I/O	ST	RA4 může být jako zdroj CLK signálu pro TMR1.
RA5/AN4/SS	7	I/O	TTL	Jako výstupní má otevřený kolektor!!! analogový vstup 4
MCLR/V_{pp}/THV	1	I/P	ST	RESET/vstup programovacího napětí. Tento vývod je aktivní v nule, kdy provádí RESET obvodu.
V_{ss}	8,19	P	-	zem
				PORT C je obousměrný vstupně/výstupní port
RC0 T1OSC T1CKI	11	I/O	ST	výstup oscilátoru nebo vstup hodin TIMER1
RC1 T1OSC CCP2	12	I/O	ST	vstup oscilátoru TIMER1
RC2/CCP1	13	I/O	ST	
RC3/SCK/SCL	14	I/O	ST	hodinový vstup/výstup pro SPI a I2C
RC4/SDI/SDA	15	I/O	ST	datový vstup SPI nebo datový vstup/výstup I2C
RC5/SDO	16	I/O	ST	datový výstup SPI
RC6/TX/CK	17	I/O	ST	synchronní hodiny nebo USART
RC7/RX/DT	18	I/O	ST	synchronní data nebo USART
				PORTB je obousměrný vstupně/výstupní port. PORTB může mít programově připojen slabý vnitřní pull-up odpor na všech vstupech. může být vybrán jako zdroj vnějšího přerušení
RB0/INT	21	I/O	TTL/ST	
RB1	22	I/O	TTL	
RB2	23	I/O	TTL	
RB3	24	I/O	TTL	přerušení při změně vstupu
RB4	25	I/O	TTL	přerušení při změně vstupu
RB5	26	I/O	TTL	přerušení při změně vstupu/CLK při
RB6	27	I/O	TTL/ST	programování
RB7	28	I/O	TTL/ST	přerušení při změně vstupu/DATA při programování
V_{dd}	20	P	-	napájení +5V
OSC2/CLKOUT	9	I	ST/CMOS	Vstup pro krystalový oscilátor/vstup ext. hodin
OSC1/CLKIN	10	O	-	Výstup krystalového oscilátoru. Připojení krystalu nebo keramického rezonátoru. V RC módu výstup CLK signálu, který je 1/4 kmitočtu na OSC1

4.11 Popis vývodů (viz [3])

4.3.3 Architektura



4.12 Architektura mikroprocesoru PIC18F458 (viz [3])

4.3.4 Sériové rozhraní SPI

Sériová komunikační rozhraní se v mikropočítačové technice používají ke dvěma základním účelům. Ke komunikaci mezi jednotlivými mikropočítačovými moduly. Typická délka vedení je zde v řádu jednotek až stovek metrů, takže fyzická vrstva v různé míře řeší i problémy odolnosti proti rušení atd. Kromě známých a rozšířených rozhraní (RS232, RS485) se používají i specializovaná rozhraní resp. sběrnice (CAN, TTP). Při rostoucích nárocích na přenosovou kapacitu se pro toto propojení často používá i Ethernet a ke komunikaci mezi integrovanými obvody, případně ke komunikaci mezi mikropočítačovými moduly na krátkou vzdálenost. Typická délka vedení zde nepřesahuje jednotky metrů. Často používaná rozhraní pro komunikaci mezi jednotlivými IO jsou rozhraní Microwire, SPI a I2C.

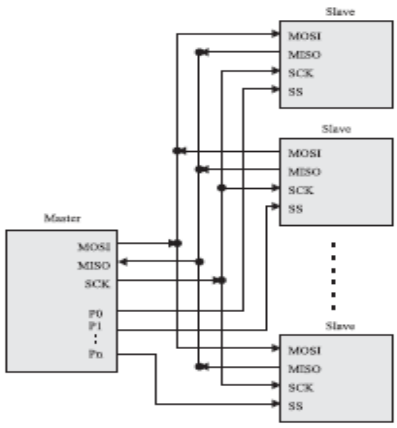
Důvodem používání sériové komunikace mezi jednotlivými obvody je především zmenšení počtu vývodů jejich pouzder. Při použití sériových pamětí se zredukuje množství adresních, datových a řídicích vývodů obvyklé (nikoliv sériové) paměti na tři až čtyři vývody. To umožňuje zmenšit rozměry pouzdra i spojové desky, protože odpadá prostorově náročné propojování velkým počtem vodičů. Další výhodou může v některých případech být možnost připojení obvodů se sériovým rozhraním i k mikrokontrolérům bez vyvedené vnitřní sběrnice, v krajním případě i bez příslušného řadiče sériového rozhraní. Funkce řadiče je potom realizována programově s využitím několika vývodů vhodného portu mikrokontroléru. Nelze tak většinou dosáhnout plné rychlosti daného rozhraní, ale tato skutečnost nemusí být v některých případech na závadu.

Rozhraní SPI je určeno především pro připojení vnějších pamětí, A/D převodníků a dalších obvodů k mikrokontroléru, případně pro vzájemnou komunikaci mezi mikrokontroléry. U některých mikrokontrolérů je SPI využíváno i pro programování jejich vnitřní paměti Flash. V našem případě bude rozhraní SPI použito pro připojení mikrokontroléru s ethernetovým modulem.

Základní koncepce systému využívajícího sběrnici SPI je následující. V systému mohou být zapojeny dva nebo více obvodů. Jeden z obvodů, obvykle procesor, je typu Master, ostatní jsou typu Slave. Jednotlivé obvody jsou propojeny čtyřmi vodiči:

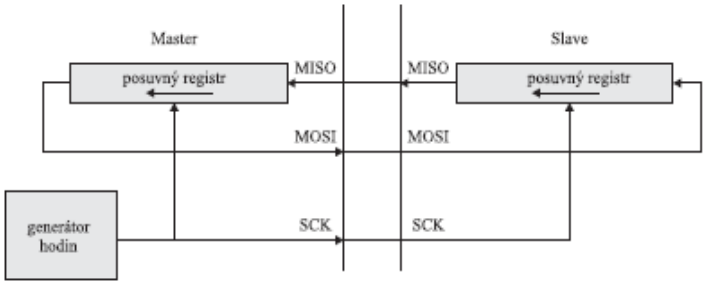
- Datový výstup MOSI (Master Out, Slave In) obvodu Master je připojen na vstupy MOSI všech obvodů Slave.

- Datový vstup MISO (Master In, Slave Out) obvodu Master je propojen s výstupy MISO všech obvodů Slave.
- Výstup hodinového signálu SCK je připojen na vstupy SCK všech obvodů Slave.
- Každý obvod Slave má vstup SS (Slave Select) pro výběr obvodu. Je-li SS v neaktivní úrovni, je rozhraní SPI daného obvodu neaktivní a jeho výstup MISO je ve vysokoimpedančním stavu. Vstupy SS jednotlivých obvodů jsou samostatnými vodiči propojeny s obvodem Master. Je-li obvodem Master mikrokontrolér, bývají tyto vodiče připojeny k některému z jeho portů. Tak lze snadno vybírat obvod, se kterým má být v daném okamžiku vedena komunikace. (viz [4])



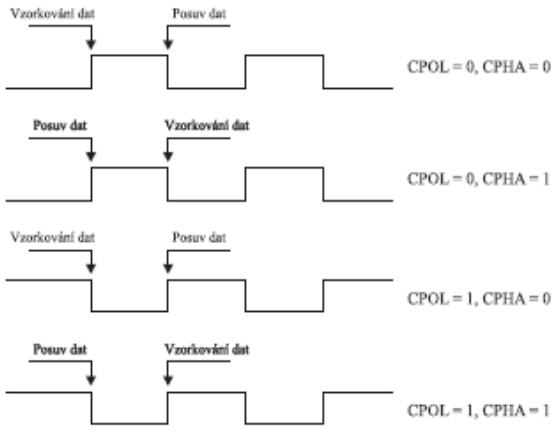
4.13 Zapojení zařízení rozhraním SPI (viz [4])

Přenosy na sběrnici SPI probíhají vždy mezi obvodem Master a některým z obvodů Slave. Oba obvody obsahují posuvné registry, které jsou v okamžiku komunikace propojeny tak, jak je schematicky naznačeno na Obr. 4.14.



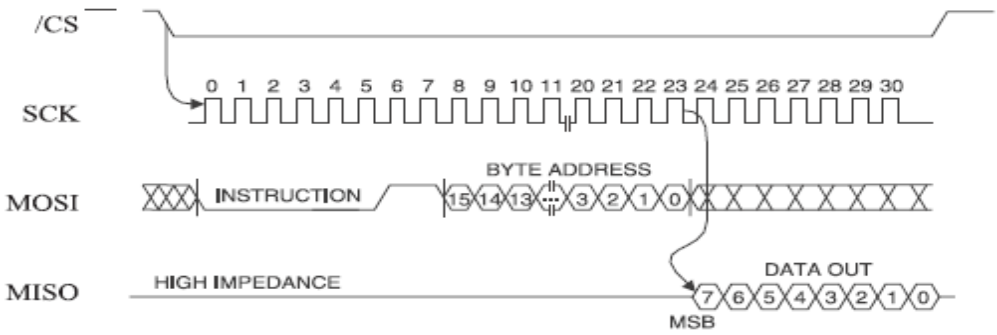
4.14 Propojení stanic Master a Slave (viz [4])

Obvod Master generuje hodinový signál, který řídí posouvání obou posuvných registrů. Klidová úroveň signálu SCK a vztah mezi datovým a hodinovým signálem je dán parametry CPOL a CPHA (viz obr. 4.15). Pokud je rozhraní SPI realizováno specializovaným řadičem, je obvykle možné tyto parametry v řadiči nastavit. Je-li rozhraní SPI realizováno programově, musí být okamžiky změny úrovně datových a hodinových signálů zvoleny tak, aby přijímající obvod vzorkoval ustálená data.

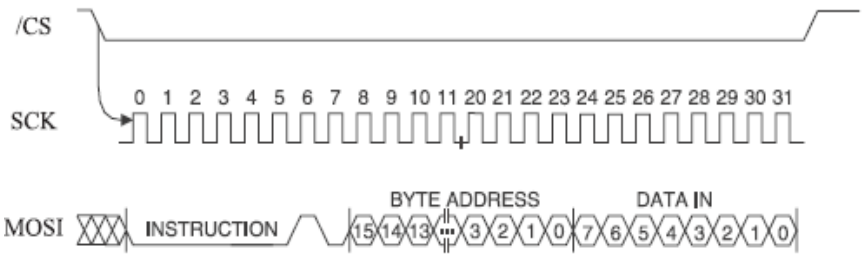


4.15 Význam parametrů CPOL a CPHA na SPI (viz [4])

Napěťové úrovně jednotlivých signálů rozhraní SPI jsou dané použitou technologií. Maximální frekvence hodinového signálu je 2 MHz. Na Obr. 4.16 a Obr. 4.17 je příklad komunikace se sériovou pamětí při čtení dat. Mikrokontrolér musí nejprve do paměti zapsat povel (čtení) a adresu dat. Potom jsou z paměti přečtena příslušná data. (viz [4])



4.16 Čtení dat z paměti s rozhraním SPI (viz [4])



4.17 Zápis dat do paměti s rozhraním SPI (viz [4])

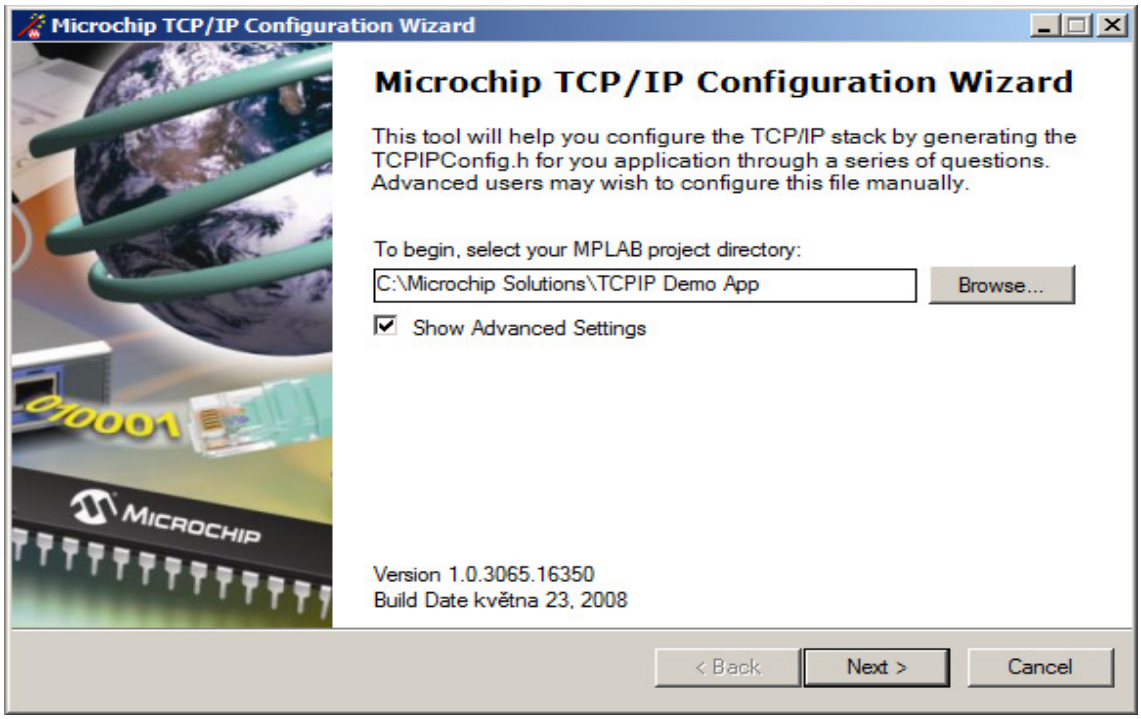
4.4 TCP/IP Stack

TCP/IP stack je balík softwaru a zdrojových kódů od firmy Microchip určený právě pro usnadnění vývoje ethernetových aplikací s mikrokontroléry PIC a Ethernetovým modulem ENC28J60, který obsahuje náš ethernetový modul. Tento software je zcela zdarma a volně k dispozici na stránkách výrobce. Přiložené příklady umožňují jednoduše vytvořit například HTTP server, e-mailového klienta a mnoho dalších zajímavých aplikací. TCP/IP stack je realizován jako modulární systém, kde každý modul reprezentuje funkce příslušné komunikační vrstvy. Potencionální uživatel nemusí přesně vědět, jak jednotlivé moduly fungují, prostě využívá jejich služeb prostřednictvím funkcí (jakýchsi API), které byly vývojáři Microchipu připraveny.

TCP/IP Stack je stavěn na základě upraveného referenčního modelu TCP/IP. Software založený na tomto modelu je rozdělován do několika základních vrstev, kde vyšší vrstvy využívají služeb vrstev bezprostředně nižších. (viz [5])

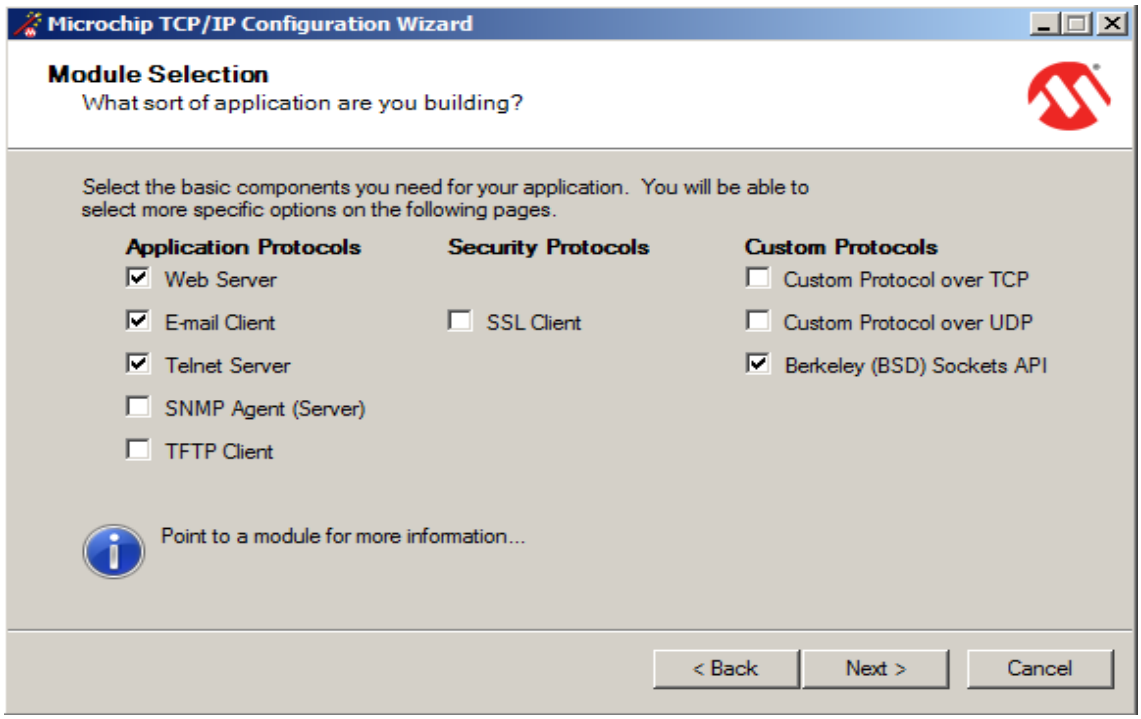
Z adresy výrobce lze stáhnout zabalený instalační soubor (viz [6]), kterým se TCP/IP Stach nainstaluje. S instalací by neměly být žádné problémy, program se nainstaluje do zvoleného adresáře a vytvoří svoji skupinu v nabídce start – programy. Skupina obsahuje pro nás nejdůležitější TCPIP Configuration Wizard, kterým vytvoříme kód pro obsluhu TCP/IP komunikace. Program umožňuje přímo zavedení kódu do mikroprocesoru a MPFS Generátor, kterým lze převést internetové stránky do kódu a nahrát do mikroprocesoru.

Nyní budou popsány jednotlivé kroky pro vytvoření zdrojového kódu pro mikroprocesor a transfer dat do mikroprocesoru včetně internetových stránek.



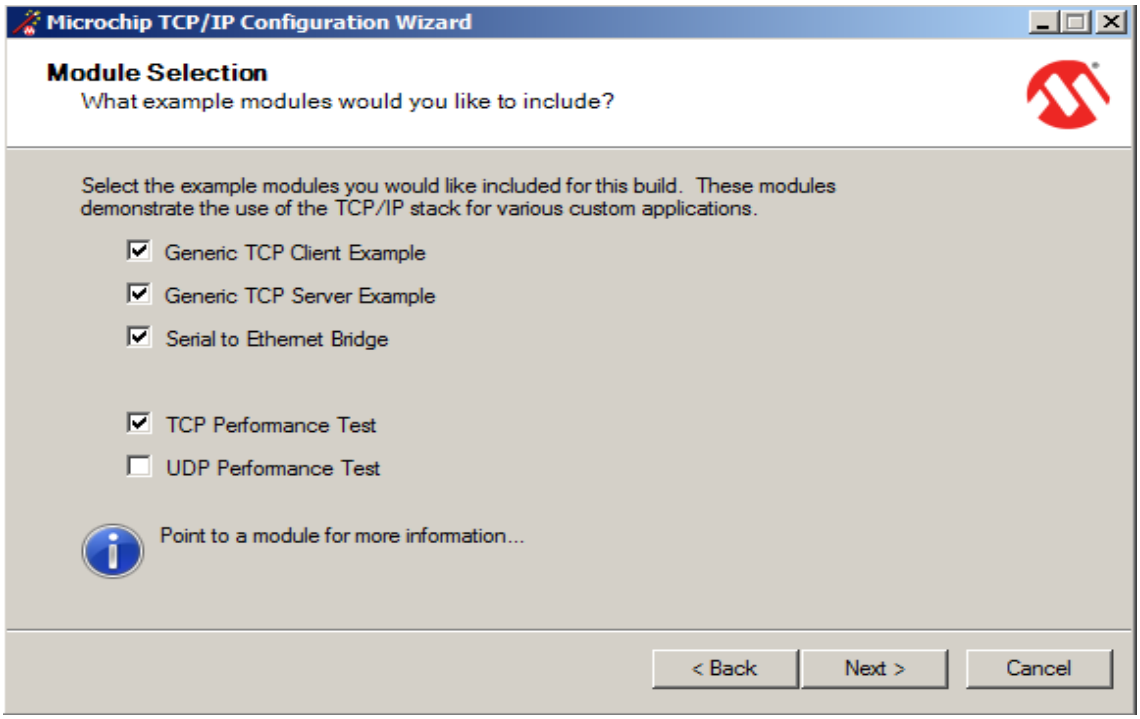
4.18 Úvodní okno programu TCP/IP Configuration Wizard

Po spuštění programu TCP/IP Configuration Wizard z nabídky start-programy-microchip se zobrazí úvodní stránka s dotazem kam uložit po vytvoření zdrojové kódy.



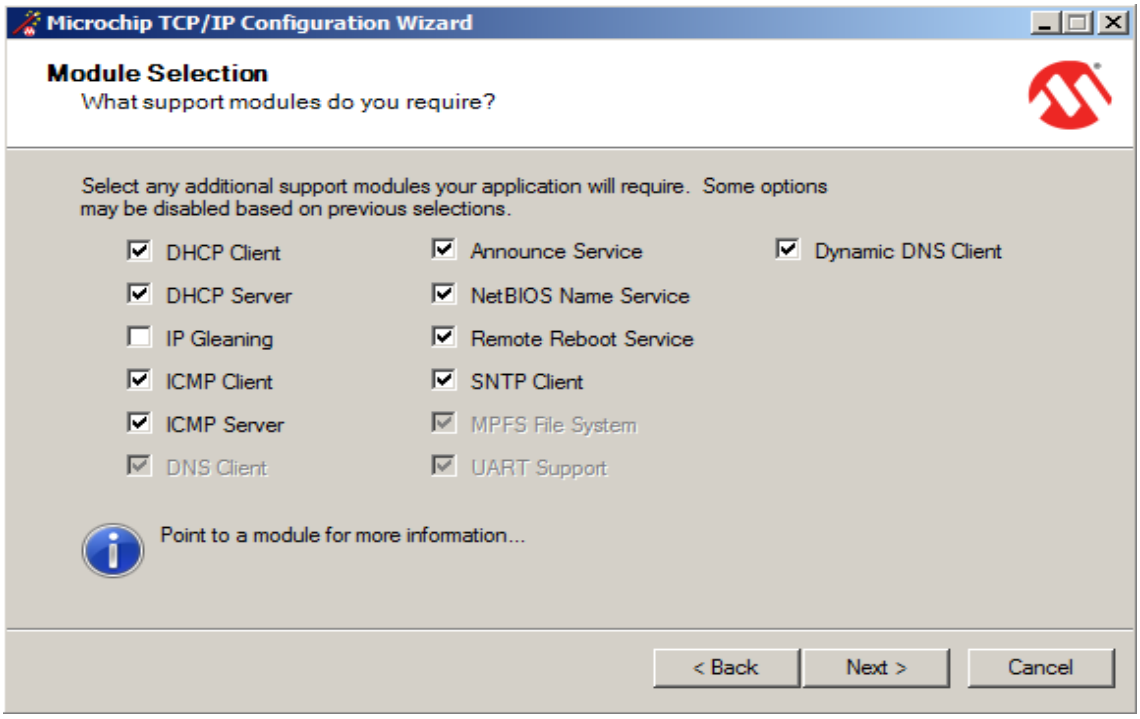
4.19 Výběr protokolů

Na další stránce je možnost si vybrat mezi protokoly, kterýma má být mikroprocesor vybaven.



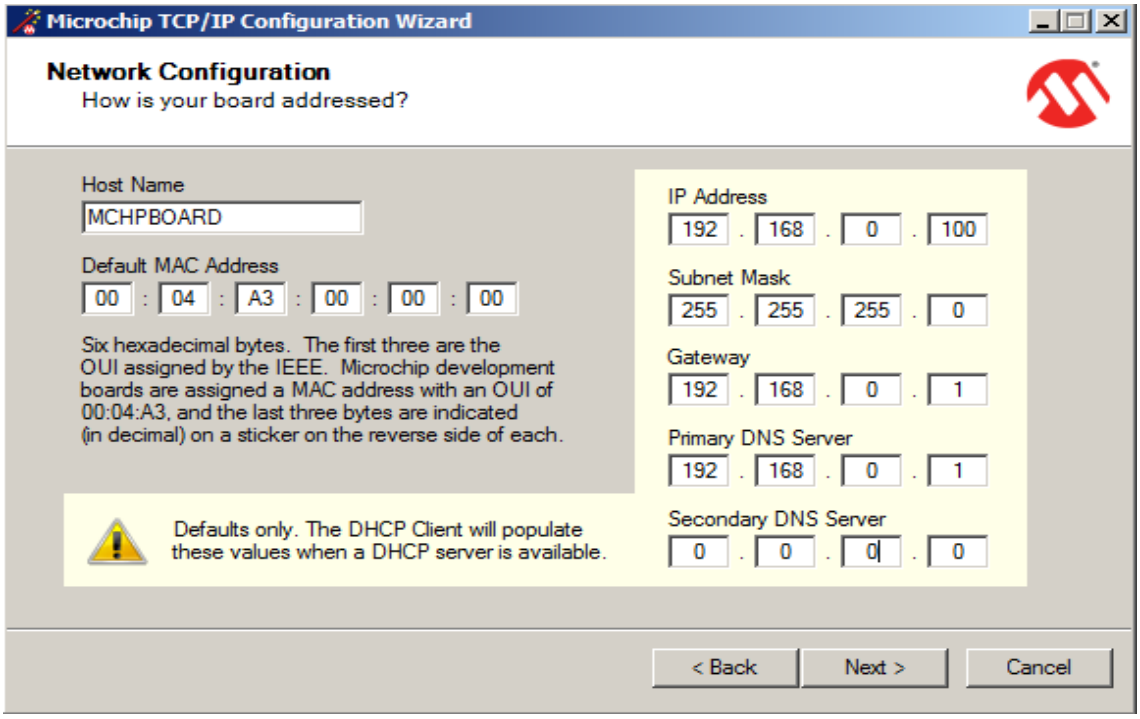
4.20 Nastavení modulů

Na dalším obrázku se vybírají moduly, které mají být zahrnuty do kódu



4.21 Povolení služeb

Na obrázku 4.21 se buď povolují nebo zakazují jednotlivé služby, kterýma má mikroprocesor disponovat.



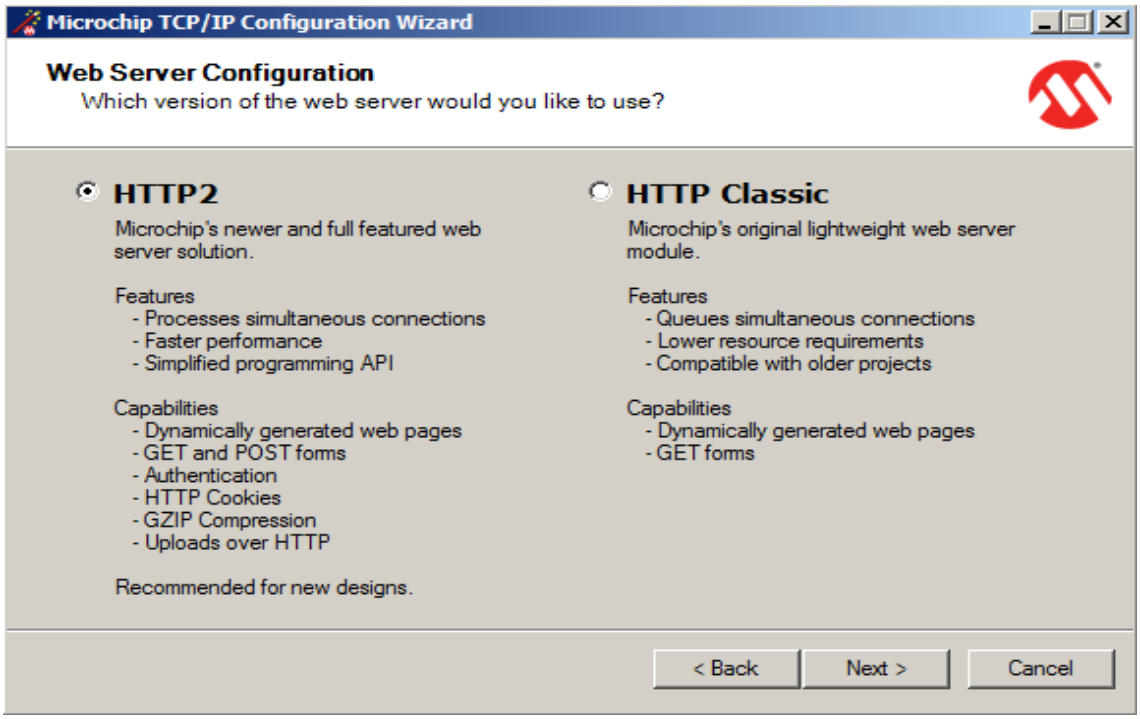
4.22 Nastavení síťových parametrů

Na obrázku 4.22 se nastavují parametry sítě tzn. IP adresa, maska sítě, výchozí brána, primární a sekundární DNS a MAC adresa mikroprocesoru. Je velice důležité aby IP adresa připojeného počítače na toto zařízení byla ve stejném rozsahu. Tzn. pokud bude mít zařízení adresu 192.168.0.1 a masku sítě 255.255.255.0 tak připojený počítač na zařízení by měl mít adresu od 192.168.0.2 do 192.168.0.254 a masku sítě 255.255.255.0 čili stejnou masku sítě. V opačném případě, nebude možné se na toto zařízení dostat. Pokud bude zařízení přidělena veřejná IP adresa, bude zařízení přístupné odkudkoliv z internetu.

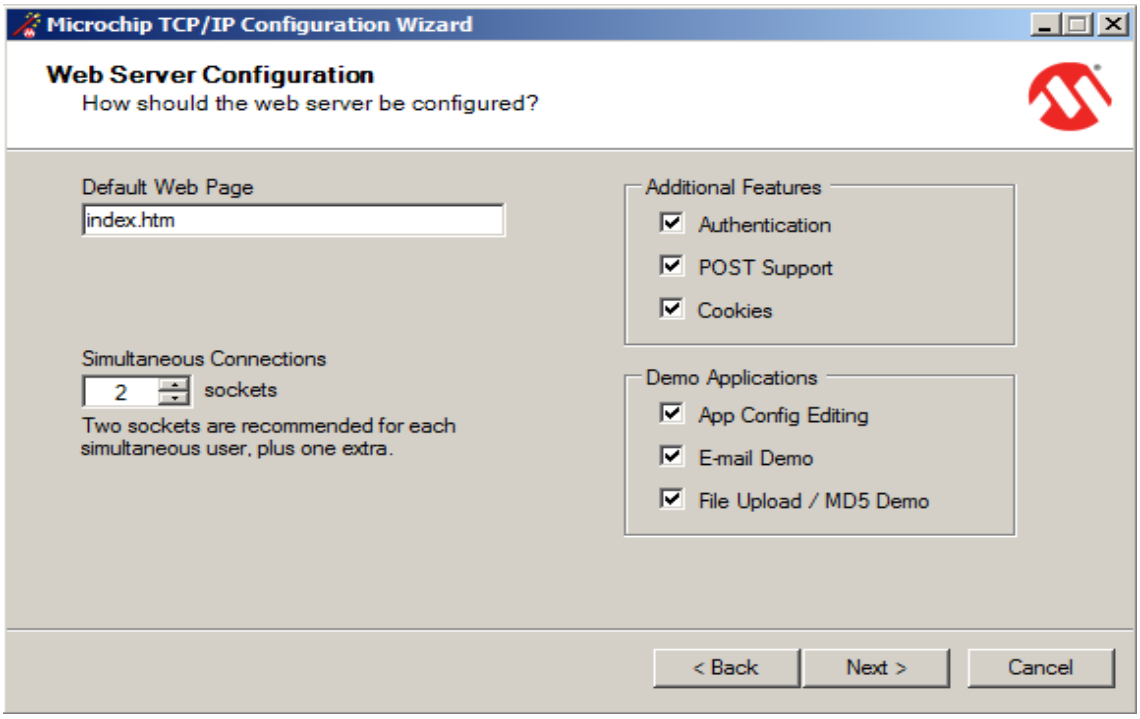
Na obrázku 4.23 se volí verze typu http protokolu, výhodnější a bezpečnější je verze http2 proto volím tuto verze.

Na obrázku 4.24 se nastavuje webové rozhraní, především jestli výchozí stránka webových stránek se jmenuje index.htm

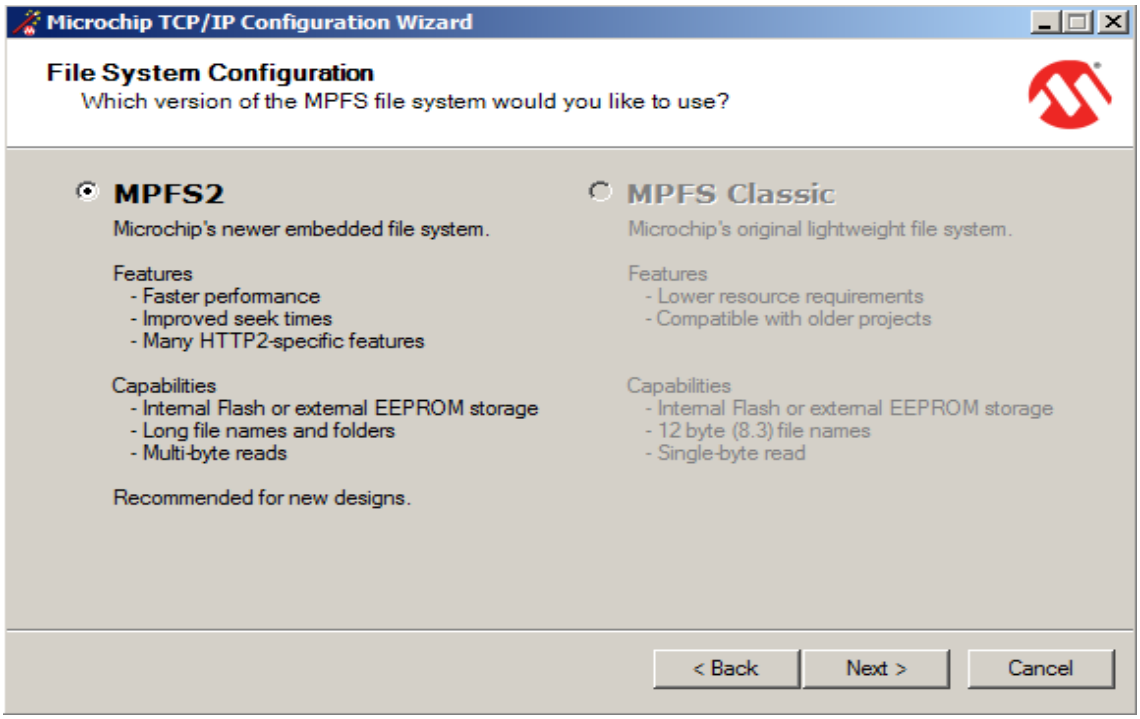
Na obrázku 4.26 se nastavuje úložiště, tzn. paměť pro kterou jsou data určeny. Na výběr jsou tři možnosti. Externí EEPROM, Externi SPI Flash a interní programová paměť.



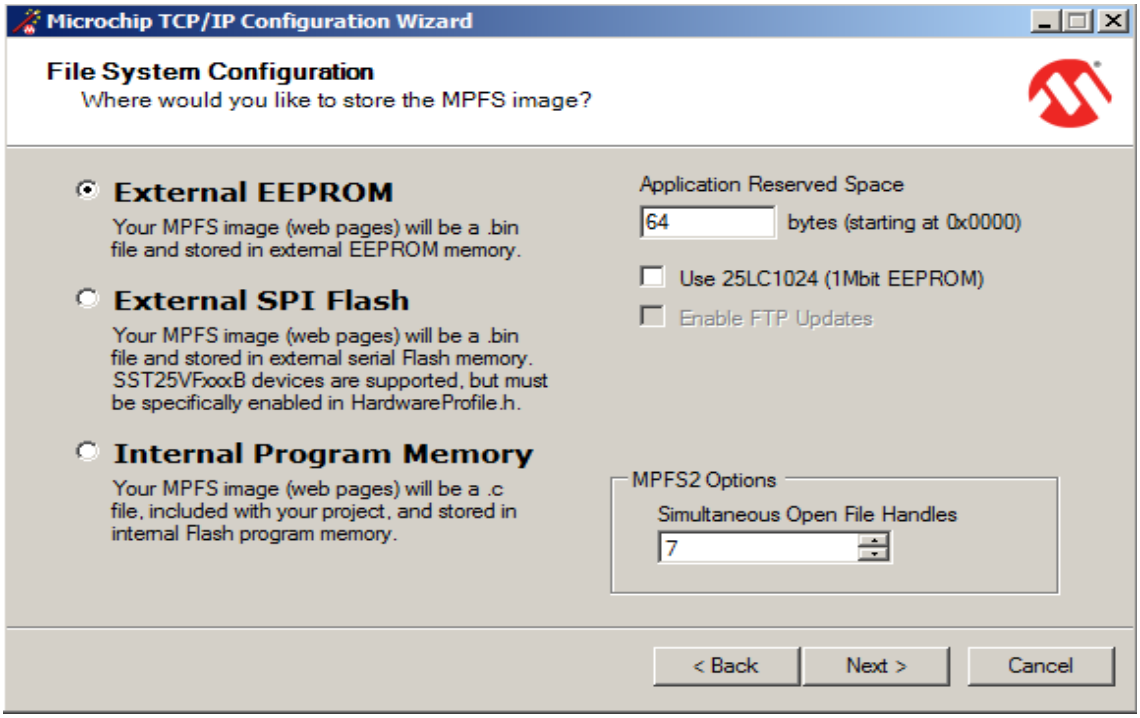
4.23 Nastavení typu http protokolu



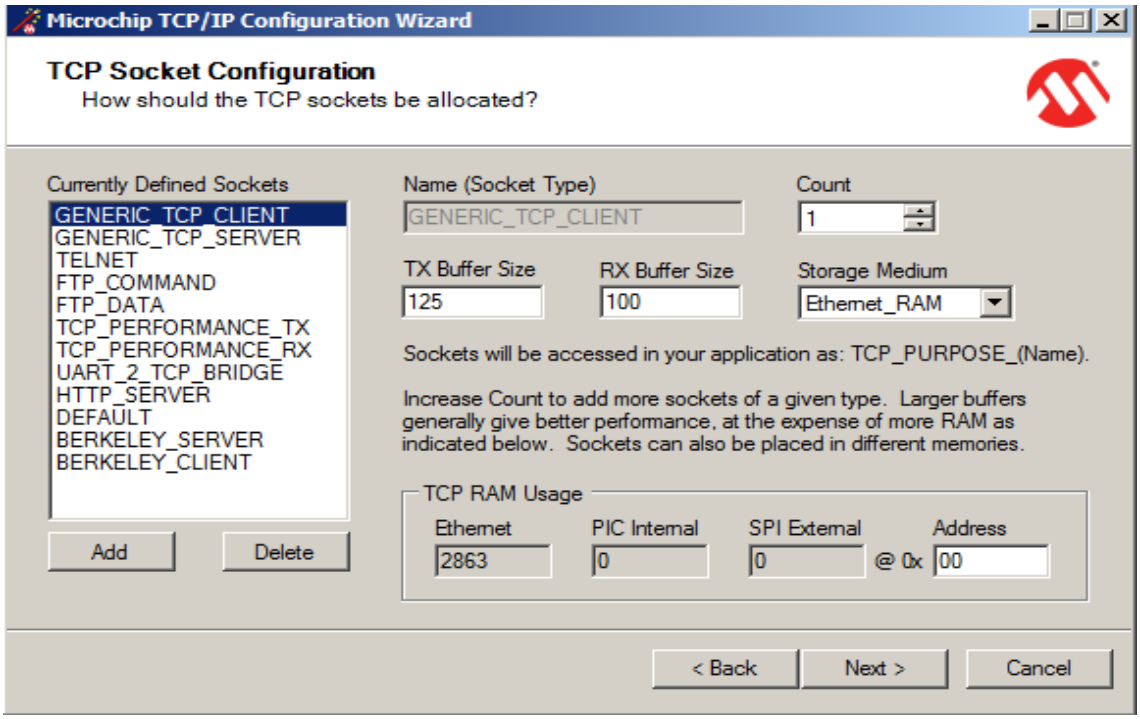
4.24 Nastavení webového rozhraní



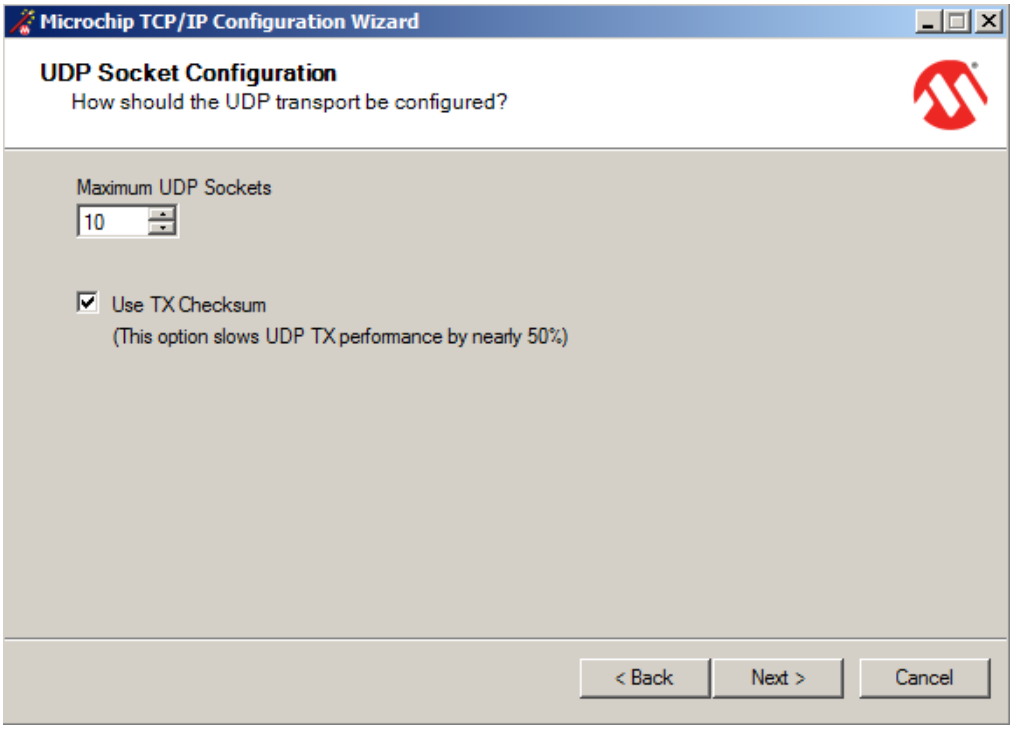
4.25 Volba souborového systému



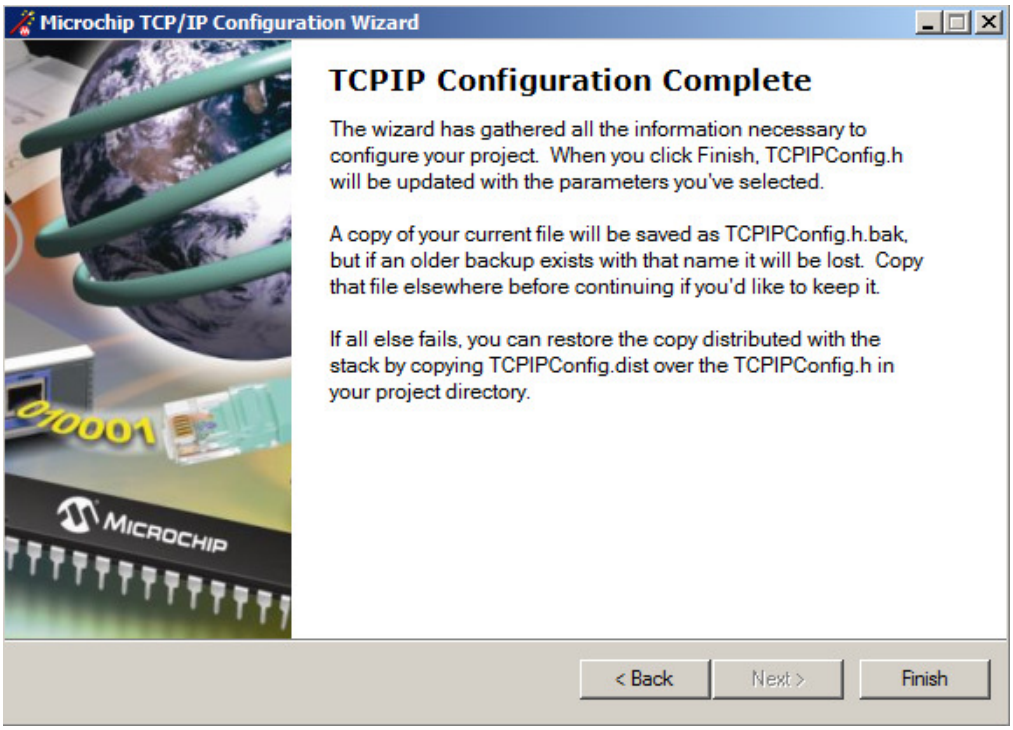
4.26 Nastavení úložiště



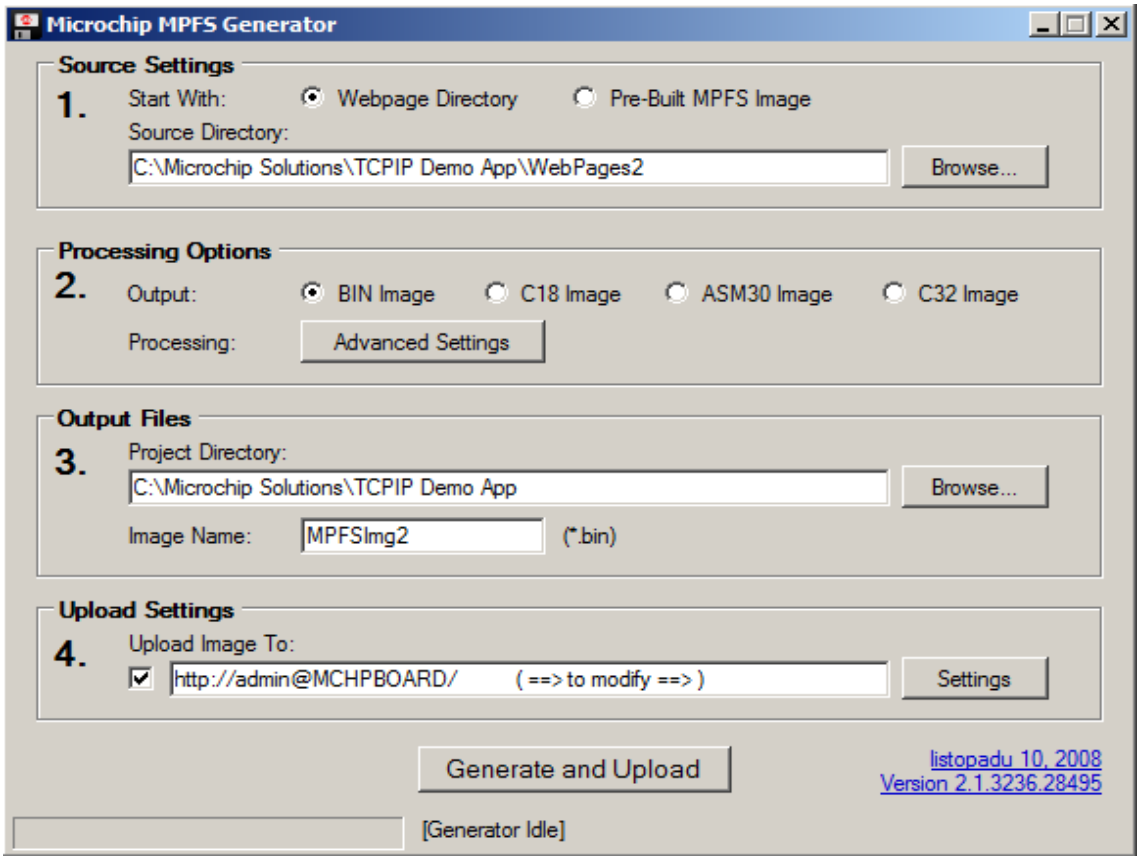
4.27 Nastavení TCP



4.28 Nastavení UDP



4.29 Ukončení průvodce



4.30 MPFS Generator

Na obrázku 4.29 je průvodce ukončen, data jsou nahrána jak v pracovním adresáři tak v mikroprocesoru. Poslední úkol, který je potřeba udělat, je transformovat vytvořené webové stránky do kódu a nahrát je do mikroprocesoru. Tuto funkci provede program MPFS Generátor. Na obrázku 4.30. je potřeba upřesnit cesty tzn. první položka je cesta k vytvořeným webovým stránkám, druhá položka nám ukazuje kam budou transformovaná data uloženy a poslední položka nahraje webovou prezentaci do mikroprocesoru.

Tímto jednoduchým způsobem byl vytvořen webový server, který běží na mikroprocesoru. Není k tomu potřeba žádný výkonný server nebo aplikace, která by tento proces simulovala.

Závěr

Cílem této diplomové práce byla konstrukce zařízení pro ovládání jiných zařízení přes webové rozhraní. Díky mikroprocesoru a softwaru firmy Microchip jsem naprogramoval mikroprocesor, který zvládá komunikaci přes rodinu protokolů TCP/IP a zároveň čte vstupy a ovládá výstupy mikroprocesoru. Pro mikroprocesor jsem vytvořil, zkonstruoval a odladil modul, ve kterém mikroprocesor pracuje. Toto zařízení s ethernetovým modulem ASIX je velice jednoduché a relativně levné zapojení, které má mnoho využití, díky tomu, že internet je v dnešní době dostupný skoro úplně všude a to formou bezdrátového internetu, pevných linek nebo mobilních operátorů, lze zařízení ovládat opravdu odkudkoliv.

Protože se zařízení chová jako webový server, nepotřebujeme žádnou aplikaci typu klient – server. Stačí pouze obyčejný webový prohlížeč, kterým lze číst vstupy a ovládat výstupy mikroprocesoru.

Pro názornost na výstupy mikroprocesoru byly připojeny LED diody, aby signalizovali stav výstupů. V praxi, ale na tyto výstupy můžeme po úpravě připojit např. sepínání osvětlení, elektrické spotřebiče. Díky A/D převodníku, který obsahuje mikroprocesor můžeme i regulovat.

Seznam použité literatury

[1] Libor Dostálek, Alena Kabelová: Velký průvodce protokoly TCP/IP a systém DNS, Computer Press, Praha, 2000, ISBN 80-7226-323-4

[2] http://www.asix.cz

[3] http://www.microchip.cz

[4] http://www.zcu.cz

[5] http://www.hw.cz

[6] http://www1.microchip.com/downloads/en/DeviceDoc/Microchip%20TCPIP%20Stack%20v4.55%20Installer.zip

Přílohy

Příloha1:

Zdrojový kód www stránek:

```
const char *html_text[]={
"<html>" "\x0D\x0A"                          // Hlavička pro všechny stránky
"<head>" "\x0D\x0A"
"\x00"
,
"<html><head>" "\x0D\x0A"                    // 404
"<title>404 Not Found</title>" "\x0D\x0A"
"</head><body>" "\x0D\x0A"
"<h1>Not Found</h1>" "\x0D\x0A"
"<p>The requested URL \xFF\x01 was not found on this server.</p>" "\x0D\x0A"
"</body></html>" "\x0D\x0A"
"\x0D\x0A"
"\x00"
,
"<title>Web-Server</title>" "\x0D\x0A"
"<style type='text/css'>" "\x0D\x0A"
"body {font-size:1em;font-family:tahoma,sans-serif;}" "\x0D\x0A"
"div#nadpis{color:#000080;font-size:1.5em;height:66px;}" "\x0D\x0A"
"p#hlavni{color='#0000FF'}" "\x0D\x0A"
"a#panda{text-decoration:none;color=black}" "\x0D\x0A"
"a#panda:hover{text-decoration:underline;color:red}" "\x0D\x0A"
"a#menu{text-decoration:none;color='#0000FF'}" "\x0D\x0A"
"a#menu:hover{text-decoration:underline;color:red}" "\x0D\x0A"
"img.right{float:right}" "\x0D\x0A"
"</style>" "\x0D\x0A"
"</head>" "\x0D\x0A"
"<body LEFTMARGIN='1' TOPMARGIN='0'>" "\x0D\x0A"
"<div id='nadpis'>" "\x0D\x0A"
"<p style='position:relative; top:20%' align='center'>" "\x0D\x0A"
"Web-Server" "\x0D\x0A"
"</p>" "\x0D\x0A"
"</div>" "\x0D\x0A"
"<p id='hlavni' align='center'>" "\x0D\x0A"
"<a href='form.htm' id='menu'>formulář</a>" "\x0D\x0A"
"</p>" "\x0D\x0A"
"<p id='hlavni' align='center'>" "\x0D\x0A"
"<a href='info.htm' id='menu'>status</a>" "\x0D\x0A"
"</p>" "\x0D\x0A"
"</body>" "\x0D\x0A"
"</html>" "\x0D\x0A"
"\x00"
,                                                                  // form.htm
"<html>" "\x0D\x0A"
"<head>" "\x0D\x0A"
"<title>In/Out</title>" "\x0D\x0A"
"<style type='text/css'>" "\x0D\x0A"
```

```
"body {font-size:1em;font-family:tahoma,sans-serif;}" "\x0D\x0A"
"div#nadpis{color:#000080;font-size:1.5em;height:66px; }" "\x0D\x0A"
"p#hlavni{color='#0000FF'}" "\x0D\x0A"
"a#panda:link{text-decoration:none;color=black}" "\x0D\x0A"
"a#panda:visited{text-decoration:none;color=black}" "\x0D\x0A"
"a#panda:hover{text-decoration:underline;color:red}" "\x0D\x0A"
"img.right{float:right}" "\x0D\x0A"
"</style>" "\x0D\x0A"
"</head>" "\x0D\x0A"
"<body>" "\x0D\x0A"
"<div id='nadpis'>" "\x0D\x0A"
"<p style='position:relative; top:20%' align='center'>" "\x0D\x0A"
"In/Out" "\x0D\x0A"
"</p>" "\x0D\x0A"
"</div>" "\x0D\x0A"
"<p align='center'> </p>" "\x0D\x0A"
"<form action='form.htm' method='GET'>" "\x0D\x0A"
"<div align='center'><center><table border='0'>" "\x0D\x0A"
"<tr>" "\x0D\x0A"
"<td><div align='center'><center><table border='0'>" "\x0D\x0A"
"<tr>" "\x0D\x0A"
"<td><font size='2' face='Arial'>IN 7</font></td>" "\x0D\x0A"
"<td> </td>" "\x0D\x0A"
"<td><input type='checkbox' " "\xFF\x07" " name='I7'" "\x0D\x0A"
"value='ON' disabled></td>" "\x0D\x0A"
"<td> </td>" "\x0D\x0A"
"<td><input type='checkbox' " "\xFF\x06" " name='I6'" "\x0D\x0A"
"value='ON' disabled></td>" "\x0D\x0A"
"<td> </td>" "\x0D\x0A"
"<td><input type='checkbox' " "\xFF\x05" " name='I5'" "\x0D\x0A"
"value='ON' disabled></td>" "\x0D\x0A"
"<td> </td>" "\x0D\x0A"
"<td><input type='checkbox' " "\xFF\x04" " name='I4'" "\x0D\x0A"
"value='ON' disabled></td>" "\x0D\x0A"
"<td> </td>" "\x0D\x0A"
"<td><input type='checkbox' " "\xFF\x03" " name='I3'" "\x0D\x0A"
"value='ON' disabled></td>" "\x0D\x0A"
"<td> </td>" "\x0D\x0A"
"<td><input type='checkbox' " "\xFF\x02" " name='I2'" "\x0D\x0A"
"value='ON' disabled></td>" "\x0D\x0A"
"<td> </td>" "\x0D\x0A"
"<td><input type='checkbox' " "\xFF\x01" " name='I1'" "\x0D\x0A"
"value='ON' disabled></td>" "\x0D\x0A"
"<td> </td>" "\x0D\x0A"
"<td><input type='checkbox' " "\xFF\x00" " name='I0'" "\x0D\x0A"
"value='ON' disabled></td>" "\x0D\x0A"
"<td> </td>" "\x0D\x0A"
"<td><font size='2' face='Arial'>IN 0</font></td>" "\x0D\x0A"
"</tr>" "\x0D\x0A"
"</table>" "\x0D\x0A"
"</center></div></td>" "\x0D\x0A"
```

```
"</tr>" "\x0D\x0A"
"<tr>" "\x0D\x0A"
"<td><div align='center'><center><table border='0'>" "\x0D\x0A"
"<tr>" "\x0D\x0A"
"<td><font size='2' face='Arial'>OUT 7</font></td>" "\x0D\x0A"
"<td> </td>" "\x0D\x0A"
"<td><input type='checkbox' " "\xFF\x17" " name='O7'" "\x0D\x0A"
"value='1'></td>" "\x0D\x0A"
"<td> </td>" "\x0D\x0A"
"<td><input type='checkbox' " "\xFF\x16" " name='O6'" "\x0D\x0A"
"value='1'></td>" "\x0D\x0A"
"<td> </td>" "\x0D\x0A"
"<td><input type='checkbox' " "\xFF\x15" " name='O5'" "\x0D\x0A"
"value='1'></td>" "\x0D\x0A"
"<td> </td>" "\x0D\x0A"
"<td><input type='checkbox' " "\xFF\x14" " name='O4'" "\x0D\x0A"
"value='1'></td>" "\x0D\x0A"
"<td> </td>" "\x0D\x0A"
"<td><input type='checkbox' " "\xFF\x13" " name='O3'" "\x0D\x0A"
"value='1'></td>" "\x0D\x0A"
"<td> </td>" "\x0D\x0A"
"<td><input type='checkbox' " "\xFF\x12" " name='O2'" "\x0D\x0A"
"value='1'></td>" "\x0D\x0A"
"<td> </td>" "\x0D\x0A"
"<td><input type='checkbox' " "\xFF\x11" " name='O1'" "\x0D\x0A"
"value='1'></td>" "\x0D\x0A"
"<td> </td>" "\x0D\x0A"
"<td><input type='checkbox' " "\xFF\x10" " name='O0'" "\x0D\x0A"
"value='1'></td>" "\x0D\x0A"
"<td> </td>" "\x0D\x0A"
"<td><font size='2' face='Arial'>OUT 0</font></td>" "\x0D\x0A"
"</tr>" "\x0D\x0A"
"</table>" "\x0D\x0A"
"</center></div></td>" "\x0D\x0A"
"</tr>" "\x0D\x0A"
"</table>" "\x0D\x0A"
"</center></div><p align='center'><input type='submit'" "\x0D\x0A"
"name='B1' value='Refresh'></p>" "\x0D\x0A"
"</form>" "\x0D\x0A"
"<p> </p>" "\x0D\x0A"
"<p><font color='#0000FF' size='3' face='Tahoma'><em><a
href=\"index.htm\">zpět</a></em></font></p>" "\x0D\x0A"
"<div class='end'>" "\x0D\x0A"
"</div>" "\x0D\x0A"
"</body>" "\x0D\x0A"
"</html>" "\x0D\x0A"
"\x00"
,
"\x00"  // nic
,
"<html>" "\x0D\x0A"                                  // info.htm
```

```
"<head>" "\x0D\x0A"
"<title>Status</title>" "\x0D\x0A"
"<style type='text/css'>" "\x0D\x0A"
"body {font-size:1em;font-family:tahoma,sans-serif;}" "\x0D\x0A"
"p#hlavni{color='#0000FF'}" "\x0D\x0A"
"a#panda{text-decoration:none;color=black}" "\x0D\x0A"
"a#panda:hover{text-decoration:underline;color:red}" "\x0D\x0A"
"a#menu{text-decoration:none;color='#0000FF'}" "\x0D\x0A"
"a#menu:hover{text-decoration:underline;color:red}" "\x0D\x0A"
"img.right{float:right}" "\x0D\x0A"
"</style>" "\x0D\x0A"
"</head>" "\x0D\x0A"
"<body LEFTMARGIN='1' TOPMARGIN='0'>" "\x0D\x0A"
"<div id='nadpis'>" "\x0D\x0A"
"<p style='position:relative; top:20%' align='center'>" "\x0D\x0A"
"Status" "\x0D\x0A"
"</p>" "\x0D\x0A"
"</div>" "\x0D\x0A"
"<p id='hlavni' align='center'>" "\x0D\x0A"
"<table border='1' cellpadding='6' cellspacing='0'" "\x0D\x0A"
"bgcolor='#FFFFFF' bordercolor='#000000'>" "\x0D\x0A"
"<tr>" "\x0D\x0A"
"<td><font size='2' face='Arial'><strong>My IP:</strong></font></td>" "\x0D\x0A"
"<td align='right'><font size='2' face='Arial'><em>\xFF\x30</em></font></td>" "\x0D\x0A"
"</tr>" "\x0D\x0A"
"<tr>" "\x0D\x0A"
"<td><font size='2' face='Arial'><strong>My MAC:</strong></font></td>" "\x0D\x0A"
"<td align='right'><font size='2' face='Arial'><em>\xFF\x31</em></font></td>" "\x0D\x0A"
"</tr>" "\x0D\x0A"
"<tr>" "\x0D\x0A"
"<td><font size='2' face='Arial'><strong>Your IP:</strong></font></td>" "\x0D\x0A"
"<td align='right'><font size='2' face='Arial'><em>\xFF\x32</em></font></td>" "\x0D\x0A"
"</tr>" "\x0D\x0A"
"<tr>" "\x0D\x0A"
"<td><font size='2' face='Arial'><strong>Your MAC:</strong></font></td>" "\x0D\x0A"
"<td align='right'><font size='2' face='Arial'><em>\xFF\x33</em></font></td>" "\x0D\x0A"
"</tr>" "\x0D\x0A"
"</table>" "\x0D\x0A"
"</p>" "\x0D\x0A"
"<p> </p>" "\x0D\x0A"
"<p><font color='#0000FF' size='3' face='Tahoma'><em><a
href=\"index.htm\">zpět</a></em></font></p>" "\x0D\x0A"
"<div class='end'>" "\x0D\x0A"
"</div>" "\x0D\x0A"
"</body>" "\x0D\x0A"
"</html>" "\x0D\x0A"
"\x00"};
```

Příloha2:

CD se zdrojovými soubory

Na přiloženém CD přikládám diplomovou práci, kompletní zdrojové kódy mikroprocesoru, překladač HCPIC18 a program Microchip TCPIP Stack