WWW a jednočipový mikroprocesor ?

Nic jednoduššího, stačí vzít procesor, v tomto případě klon řady 51 a LAN kartu, původně určenou pro počítače PC, pár drobných součástek jako je konektor pro ISA kartu, kousek bastldesky, krystal, jeden až dva elektrolytické kondenzátory, dva keramické a pokud chcete i nějaká blikátka tak pár LEDek s příslušnými odpory. Vše se řádně propojí a co vznikne? WWW server, v tomto případě jeden z nejmenších na světě.

Podnětem k tomuto projektu byl článek z červencového čísla časopisu Circuit Cellar popisující $25 WEB SERVER postavený na RISC procesoru AT90S8515. Na webu lze najít i další více, či méně zdařilé konstrukce, např. Desrosiersův Embedded 10BaseT Ethernet, postavený na LAN čipu CS8900 od firmy Crystal (Cirrus Logic).

Pro tuto konstrukci jsem použil LAN kartu SP2020B od firmy Micronet, používající NE2000 kompatibilní řadič RTL8019AS od firmy Realtek. Tato karta je propojena s jednočipovým procesorem 8052, resp. jeho novějším klonem AT89C52 od firmy Atmel. Tento procesor obsahuje 8 kByte programové paměti a 256 byte paměti RAM, obsahující jak registry procesoru a stack, tak datovou paměť. Je vidět, že použitý procesor nedává možnost velkého programátorského rozmachu, nicméně praktická realizace beta verze firmware ukazuje, že prostředky tohoto procesoru jsou pro zvolený účel postačující. Beta verze zabírá necelé 3 kByte a obsahuje obsluhu LAN karty, založenou na paketovém driveru a podporu protokolů ARP, ICMP, IP a TCP. Software podporuje jednoduchý WWW server a umožňuje prostřednictvím WWW řídit či sledovat zbývající I/O piny, nepoužité pro komunikaci s LAN kartou. Rovněž umožňuje pomocí telnetu připojit se k sériovému portu procesoru a tím jej používat jako převodník LAN/sériová linka.

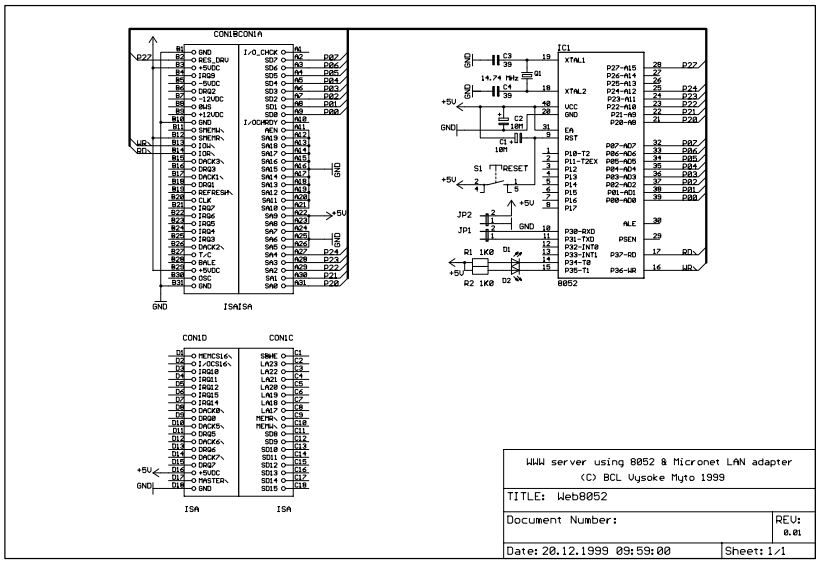
Hardware WWW8052 - prototyp

Použitá LAN karta je jednou z NE2000 kompatibilních karet. Její výhodou jsou roky používáním ověřené drivery dostupné v tomto případě i na zdrojové úrovni. Čip RTL8019AS použitý na této kartě je jednou z moderních implementací karty NE2000 v provedení "vše na jednom čipu". Čip v sobě integruje 16 kByte SRAM paměti, modulační a demodulační obvody fyzického rozhraní, vlastní řadič ethernetovského protokolu, obvody ovládání paměti a řadu dalších funkcí.

Neobsahuje však na rozdíl od čipu CS8900 filtry na vstupu/výstupu etemetovského rozhraní a proto je nutno použít vstupně/výstupní transformátor obsahující dolní propust. Hlavní výhodou tohoto čipu je však možnost provozovat jej v osmibitovém režimu. Toho je využito i v následujícím zapojení, zobrazeném na schematu 1. Karta je s procesorem propojena takto:

P0 je použit pro vytvoření datové "ISA" sběrnice
P2 je použit zčásti (P2.0 až P2.4) jako adresová "ISA" sběrnice, na druhou část je připojen reset "ISA" sběrnice (P2.7)
P3.6 a P3.7 je použit ve funkci řízení čtení a zápisu, kdy je použita alternativní HW funkce tohoto procesoru

Procesor je v prototypu osazen krystalem 14,74 MHz, umožňujícím snadné generování frekvencí nutných pro dodržení přesných rychlostí sériového portu. Není-li tato sériová komunikace požadována, je možno osadit krystal s frekvencí jinou, nejlépe však maximální procesorem povolenou, zvýší se tak rychlost odezvy. Prototyp odebírá cca 1 W.



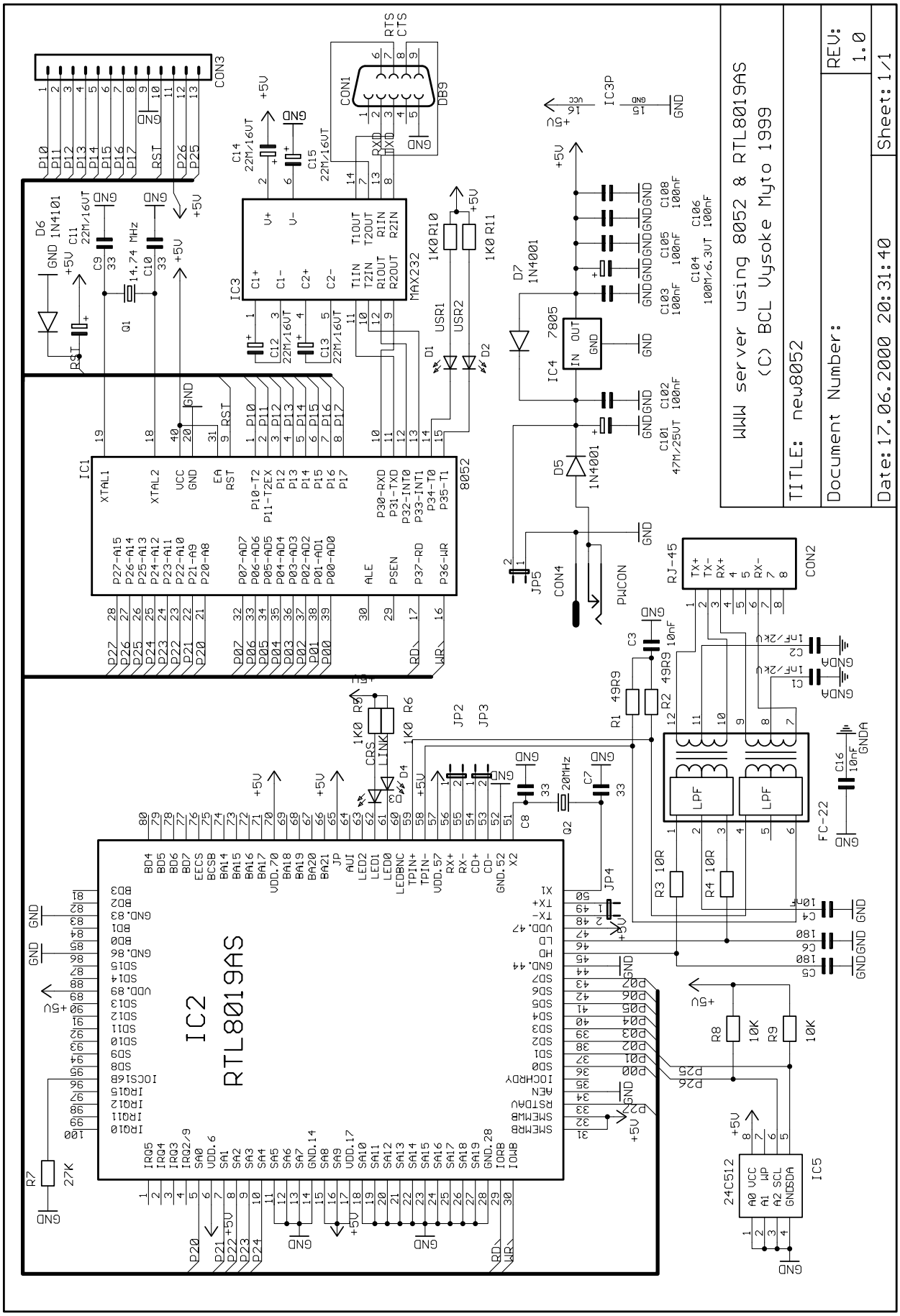
Schema 1

Vzhledem k tomu, že karta není konfigurována v tomto zapojení pomocí PnP ale má uloženu konfiguraci v EEPROM na kartě, je nutno ji před prvním použitím v této aplikaci nastavit pomocí PC a utility z driver diskety. V tomto zapojení je použita jako bázová adresa 300h, napevno nastavená propojením vodičů SA5 až SA19 ISA sběrnice na GND popř. +5V. Na nastavení Boot ROM a přerušení nezáleží, nejsou použity.

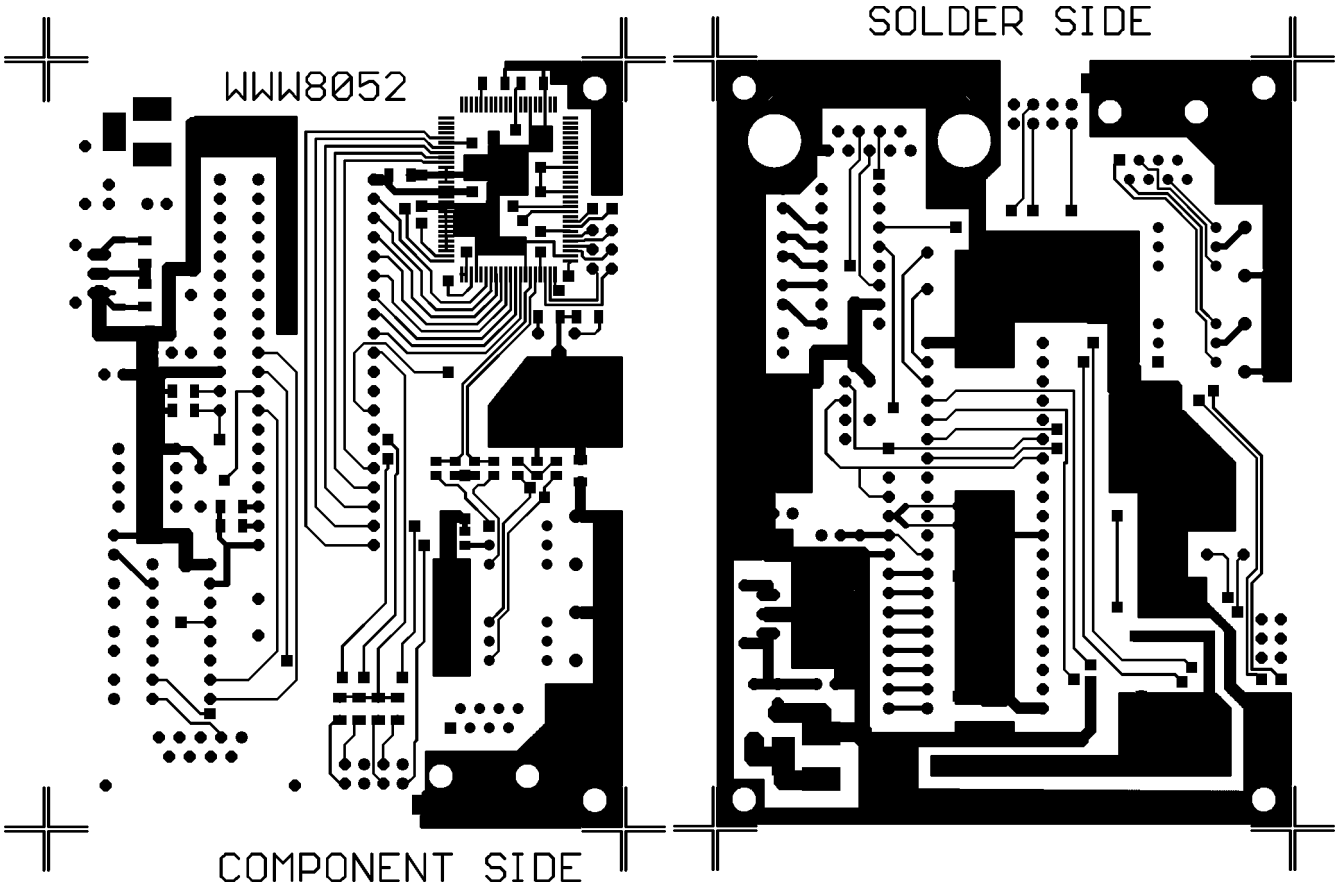
Hardware WWW8052 - kompaktní verze

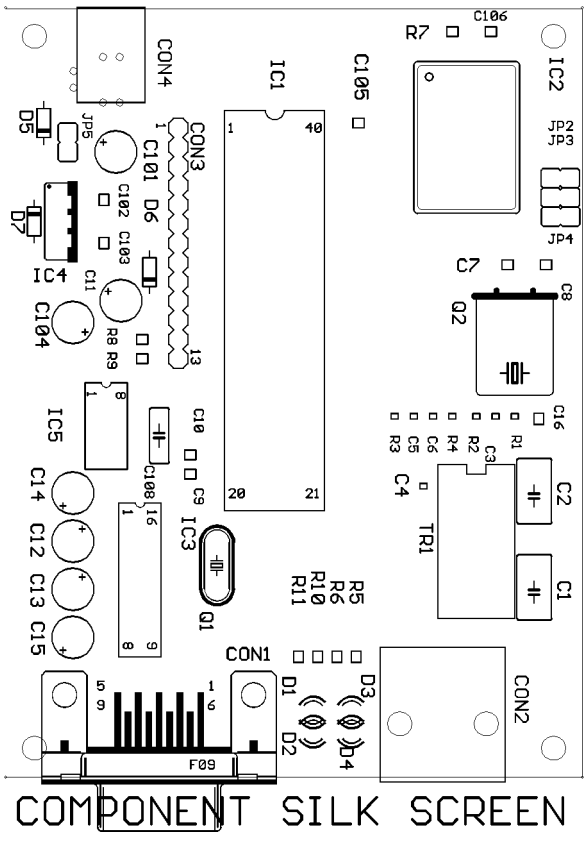
Kompaktní verze vychází z prototypu a pro první sérii i z řešení karty Micronet, karta poslouží jako jednoduchý zdroj součástek. Pohledem na schema 2 je vidět, že byl doplněn jednoduchý zdroj, obvody převodníku TTL a RS232 a paměť EEPROM. Zdroj přímo na desce umožňuje pro napájení použít jednoduchý adaptér "do zdi". Převodník úrovní umožňuje přímo připojit zařízení ovládané po sériovém portu. Použité řešení realizuje i jednoduchý, standartní HW handshake pomocí signálů RTS/CTS. Paměť EEPROM je určena jako paměť pro vlastní WWW stránky a v připravované verzi software i pro uložení CGI skriptů. Finální verze předpokládá použití obvodu 89C8252, vzhledem k jeho možnosti ISP programování. Tomuto programování je přizpůsoben i konektor CON3, sloužící k připojení celého serveru k vnějšímu prostředí. Tento konektor umožňuje připojit i externí I2C periferie.

Obvod RTL8019AS je zapojen v minimalizované verzi, tzn. bez paměti EEPROM, sloužící pro uložení konfigurace, MAC adresy, ... Tyto údaje jsou zčásti uloženy přímo ve firmware (MAC adresa), zčásti jsou realizovány přepnutím karty do "jumper" režimu. Těchto přepínačů je celkem 16 a většina je jich realizována jako sdílené s jiným signálem. Princip sdílení spočívá v tom, že ve stavu reset, kdy se vývody sloužící při normální funkci jako výstupy přepnou do třetího stavu, je možno přes jednoduché odporové děliče nastavit konfiguraci. Tyto děliče jsou realizovány odpory 100K proti zemi, realizovanými ve struktuře obvodu, a přes "jumpery" externími odpory 10K proti +5V. Těmito přepínači je nastavována adresa boot ROM, bázová I/O adresa, přerušení a typ media.


WW server using 8052 & RTL8019AS
(C) BCL Vysoke Myto 1999
TITLE: new8052
Document Number:
REV: 1.0
Date: 17.06.2000 20:31:40
Sheet: 1/1
IC1
8052
IC2
RTL8019AS
IC3
MAX232
IC4
7805
IC5
24C512
CON1
DB9
CON2
RJ-45
CON3
CON4
PWCON
JP1
JP2
JP3
JP4
JP5
Q1
14.74 MHz
Q2
20MHz
D1
D2
D3
D4
D5
D6
D7
1N4001
1N4148
LPF
FC-22
LINK
CRS

Schema 2


WWW8052
SOLDER SIDE
COMPONENT SIDE

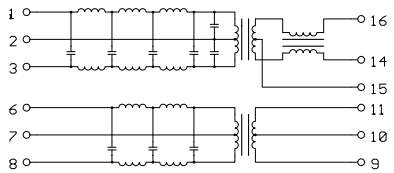

CON4
IC1
C105
R7
C106
IC2
JP2
JP3
JP4
D5
JP5
CON3
C101
D6
D7
C102
C103
IC4
C11
C104
R8
R9
C7
C8
Q2
C16
C5
C6
R4
R2
C3
R1
R3
C4
TR1
C2
C1
IC5
C14
C12
C13
C15
C10
C9
C108
IC3
Q1
R5
R6
R10
R11
CON1
D1
D2
D3
D4
CON2
F09
COMPONENT SILK SCREEN


WWW8052
CON1

Neosadíme-li tyto přepínače dostáváme tyto implicitní hodnoty

boot ROM *(BS4..BS0 = 00000)* - disabled
bázová I/O adresa *(IOS3..IOS0 = 0)* - 300h
přerušení *(IRQ2..IRQ0 = 0)* - int 2/9
medium *(PL1..PL0 = 0)* - TP/CX auto detekce se zapnutým 10BaseT linkovým testem
Stav přepínač PnP je nezajímavý, vzhledem k přepnutí čipu pomocí JP=1 do jumper modu. Přepínač AUI přepína při autodetekci režim AUI/BNC a je rovněž nezajímavý. Těmto implicitním hodnotám je podřízeno i celé zapojení a pohledem zpět je vidět, že jsou dodrženy i na prototypu.

Relativně nejsložitejší je na celém zapojení obvod 10BaseT tranceiveru. Obvod RTL8019AS v sobě obsahuje veškeré digitální a analogové obvody mimo vstupních/výstupních filtrů a oddělovacích transformátorů. Obvod proto vyžaduje připojení externího hybridního filtru/transformátoru. Doporučovaným obvodem je obvod 20F001N.



Jak je vidět, obvod obsahuje dvojici dolních propustí s frekvencí 17 MHz a dva oddělovací transformátory. Toto zapojení slouží k odstranění vyšších harmonických, vznikajících vlivem toho, že výstup z obvodu má digitální charakter a je potřeba mu "zaoblit" hrany tak, aby co nejvíce připomínal sinusovku a ne obdélník. Odfiltrováním harmonických signálů se přes UTP kabeláž přenáší pouze užitečný signál a nikoliv nežádoucí rušení.

Karta Micronet obsahuje obdobný hybrid FC-22 (od firmy GTS). Oproti zapojení demo board od Realteku bylo připojení tohoto hybridu doplněno na výstupu přizpůsobovacím impedančním a filtračním obvodem složeným z R3/C5 a R4/C6. Řešení vstupních obvodů bylo důsledně symetrizováno. Kondenzátory C1 až C4 uzemňují pro střídavý signál středy oddělovacích transformátorů a slouží k potlačení nesymetrických rušicích signálů.

Stavíme WWW8052

Celou desku www serveru, zobrazenou na schematu 2 a deskách plošných spojů je možno rozdělit do několika základních částí.

Zdroj

Zdroj integrovaný na desce vychází z klasického zapojení s obvodem 7805. Napájecí napětí z"adaptéru do zdi" je vedeno přes konektor CON4 na diodu D5 (libovolná malá usměrňovací) chránící celou desku před přepólováním na filtrační elektrolytický kondenzátor C101. Obvod 7805 je v klasickém zapojení, C102 a C103 blokují celé zapojení před zákmity, dioda D7 chrání stabilizátor před poškozením při poklesu vstupního napájecího napětí pod úroveň napětí na výstupu. Kondenzátor C104 slouží jako hlavní výstupní filtrační kondenzátor. SMD keramické kondenzátory C105 a C106 blokují napájení procesoru a obvodu RTL proti zákmitům. Kondenzátor C108 je v blokující v klasickém provedení, buď keramickém nebo lépe speciálním odrušovacím.

Procesor

Reset procesoru je tvořen v nejjednodušší verzi pouze DC obvodem D6 a C11. Kondenzátor C11 resetuje procesor při zapnutí napájení. Dioda D6 slouží k vybití C11 při resetu zařízení krátkým vypnutím napájecího napětí. Při stavbě se ukázalo vhodné nespoléhat na integrovaný odpor v procesoru a doplnit tuto kombinaci ještě doplněním odporu cca 10K k paralelně k diodě D6. Pro profi spolehlivost se vyplácí celé toto zapojení nahradit integrovaným resetovacím obvodem, např. DS1833 v pouzdru TO92.

Oscilátor je klasický, tvořený krystalem Q1 a kondenzátory C9 a C10. Použitý krystal je vminiaturním provedení HC49U/S, na desku je však možno osadit i krystal v klasickém pouzdru HC49/U či HC18.

K procesoru je připojena I²C paměť s obsahem www stránek či skripty, tvořená obvodem IC5. I²C sběrnice je zakončena pull up odpory R8 a R9. Jejich hodnota závisí na kapacitě a zatížení celé sběrnice. Je-li sběrnice použita externě je vhodné ji zakončit i na druhém konci, další dvojicí pull up odporů.

Komunikaci procesoru s okolím zabezpečuje sériová linka řízená interním UARTem. Logické úrovně TTL a RS232 jsou mezi sebou převáděny obvodem IC3. Obvod MAX232 či jeho ekvivalent vsobě obsahuje dvojici převodníků TTL/RS232, dvojici převodníků RS232/TTL a měnič 5V/+-10V. Zabudovaný měnič napětí na principu nábojové pumpy potřebuje ke své činnosti pouze čtyři externí kondenzátory C12 až C15. Zapojení umožňuje, mimo holé komunikace přes TxD/RxD ijednoduché řízení toku dat pomocí signálů RTS/CTS. Výstupní konektor CON1, je Cannon 9 v provedení FEMALE (dutinky) do desky plošného spoje.

Další rozšiřování a programování procesoru přes ISP rozhraní zabezpečuje konektor CON3, tvořený modulární lámací konektorovou lištou (13 pinů, rozteči 2.54 mm). Na tento konektor je vyveden port P1, zem, reset, napájení a sběrnice I²C.

Vizuální komunikaci mezi uživatelem a programem v procesoru zabezpečují LED diody D1 a D2. Proud diodami je omezen pomocí sériových odporů R10 a R11.

Ethernet řadič

Je realizován obvodem IC2 v minimalizovaném zapojení, s konfigurací nastavenou natvrdo jumper režimem. Odpor R7 nastavuje při resetu obvod do osmibitového režimu.

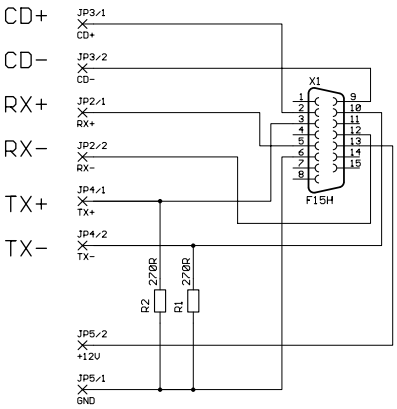
Vizuální sledování stavu komunikace na ethernetovském rozhraní umožňují LED diody D3 a D4. Proud diodami je omezen pomocí sériových odporů R5 a R6. Dioda D3 indikuje příjem paketu, dioda D4 indikuje kolizi na ethernetu. Duální funkci tj. indikaci LINK a CRS se autorovi bez použití konfigurační EEPROM nepodařilo zatím dosáhnout.

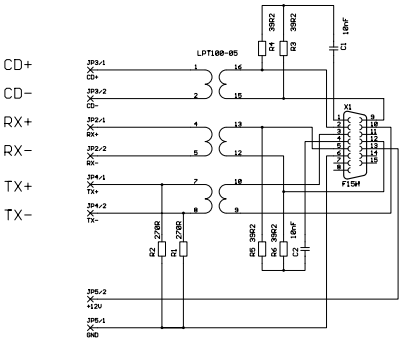
Obvod má samostatný krystalový oscilátor tvořený krystalem Q2 a kondenzátory C7 a C8.

AUI rozhraní

Rozhraní, zobrazené na následujícím obrázku, je zjednodušeno a předpokládá použití převodníku se zabudovaným oddělovacím transformátorem na signálech TX/RX/CD.

Pokud by mělo být plnohodnotné, je potřeba tento transformátor, např. 16PT-005A, zařadit mezi výstupní AUI konektor a desku WWW8052.





Deska neobsahuje konektorové pole pro přímé osazení konektoru AUI, připojení konektoru je provedeno kabelem, pomocí propojovacích bodů JP2 (RX), JP3 (CD) a JP4 (TX), vedoucích na příslušné vývody na řadiči. Na výstup TX+ a TX- je potřeba doplnit zatěžovací odpory 270R(1%), zapojené proti zemi. Připojení napájení na AUI je pomocí JP5, paralelně připojeného na vstupní filtrační kondenzátor C101. Desku WWW8052 je proto nutné napájet napětím požadovaným AUI rozhraním a příslušně dimenzovat i co se týká odběru. Úprava je zhruba vidět na této fotografii.

TP rozhraní

Je integrováno na desce WWW8052. Celé zapojení je důsledně symetrizováno.

Na straně vysílače signál prochází dvojicí RC článků R3/C5 a R4/C6 sloužících k impedančnímu přizpůsobení a základní filtraci vysílaného signálu. Signál dále prochází hybridním filtrem, popsaným v předchozí části, na výstup, tvořený konektorem CON2.

Přijímaný signál prochází z konektoru CON2 přes hybridní filtr přímo do obvodů řadiče. Přijímač je

na straně řadiče impedančně zakončen pomocí odporů R1 a R2. Tyto odpory by měly mít impedanci rovnu polovině impedance vedení, tj. 100R/2=50R.

Kondenzátory C1 až C4 slouží k potlačení nesymetrických rušivých signálů a uzemňují pro střídavý signál středy oddělovacích transformátorů. Kondenzátor C16 pro střídavý signál spojuje země WWW8052 a UTP interface.

Stavíme WWW8052 MAC adresy

Kde vzít MAC adresu pro WWW8052?

První možností, ale spíše jen z říše snů, je zařadit se po bok takových výrobců, jako je Intel či 3Com a nechat si přidělit příslušný MAC segment.
Druhou možností je vymyslet si své, náhodně vygenerované číslo. Vzhledem k minimální pravděpodobnosti, že se potkají dvě stejné 48-osmibitové adresy na jedné síti, je tato metoda, obzvláště v amatérské praxi často používána.
Třetí, autorem použitá možnost je, použít adresu z jiného zařízení, o kterém je stoprocentní jistota, že neexistuje. V tomto případě je takovýmto zařízením vždy odpovídající síťová karta, která posloužila jako zdroj materiálu.

Jak je možno tuto MAC adresu ze síťové karty získat? Opět jsou tři možnosti.

Kartu vloži do PC a pomocí utilit z ní MAC adresu přečíst.
Použít pokusnou verzi WEB8051 a pomocí packetových driverů MAC adresu přečíst. Nicméně, vzhledem k tomu, že jde o PnP adaptér a WEB8051 umí pracovat jen s kartou příslušně seřízenou, tak se občas neobejdeme bez seřízení karty v PC.
Třetí možností, je přečíst MAC adresu z EEPROM na kartě

Obsah konfigurační paměti RTL8019

Význam jednotlivých hodnot je uveden v dokumentaci čipu RTL8019AS, nicméně není nad názorný příklad, přečtený z karty Micronet SN. 93700936:

```
(00H): 10 00 90 00 00 00 E8 EE  ..É...##
(08H): 10 33 20 20 20 20 20 20  .3
(10H): 20 20 41 D0 00 60 33 10    A#.`3.
(18H): EE 00 FB 0A 10 10 82 21  #.#...é!
(20H): 00 4E 45 32 30 30 30 20  .NE2000
(28H): 50 4C 55 47 20 26 20 50  PLUG P
(30H): 4C 41 59 20 45 54 48 45  LAY ETHE
(38H): 52 4E 45 54 20 43 41 52  RNET CAR
(40H): 44 00 16 00 00 95 19 02  D....Ó..
(48H): 00 1C 41 D0 80 D6 47 00  ..A#Č#G.
(50H): 00 02 E0 03 20 20 23 38  ..#.  #8
(58H): 9E 01 79 09 FF FF FF FF  Ř.y.
(60H): FF FF FF FF FF FF FF FF
(68H): FF FF FF FF FF FF FF FF
(70H): FF FF FF FF FF FF FF FF
(78H): FF FF FF FF FF FF FF FF
```

Nyní tedy podrobněji:

```
(00H): 10 00 90 00
```

První čtyři byte obsahují konfigurační data pro registry CONFIG1 až CONFIG4

```
(04H): 00 00 E8 EE 10 33
```

Dalších 6 byte obsahuje požadovanou MAC adresu v tomto případě tedy 00-00-E8-EE-10-33

```
(0AH): 20 20 20 20 20 20 20 20
```

Produkt ID je jak je vidět v případě Micronetu nevyužito.

Za touto částí jsou v EEPROM PnP data: Nejprve je PnP identifikátor, sestávající se ze dvou částí:

```
(12H): 41 D0 00 60
```

Vendor ID

```
(16H): 33 10 EE 00
```

Serial Number

```
(1AH): FB
```

Suma PnP identifikátoru tj. Vendor ID a Serial Number
PnP identifikátor je následován PnP Resource Daty:

```
(1BH): 0A 10 10 82 21 00
(21H): "NE2000 PLUG PLAY ETHERNET CARD"
(41H): 00 16 00 00 95 19 02
(48H): 00 1C 41 D0 80 D6 47 00
(50H): 00 02 E0 03 20 20 23 38
(58H): 9E 01 79 09
```

Stavíme WWW8052 - Rozpis materiálu

Součástku označené (*) jsou "recyklovány" ze síťové karty Micronet. V kusovém množství jsou obvody RTL a10BaseT transformátory v běžných obchodních kanálech nedostupné.

WWW8052 - rozpis materiálu dle označení součástek

Part	Value	Package
C1(*)	1nF/2kV	
C2(*)	1nF/2kV	
C3	10nF	SMD 0805
C4	10nF	SMD 0805
C5	180pF	SMD 0805
C6	180pF	SMD 0805
C7	33pF	SMD 1206
C8	33pF	SMD 1206
C9	33pF	SMD 1206
C10	33pF	SMD 1206
C11	22M/16VT	
C12	22M/16VT	
C13	22M/16VT	
C14	22M/16VT	
C15	22M/16VT	
C16	10nF	SMD 1206
C101	47M/25VT	
C102	100nF	SMD 1206
C103	100nF	SMD 1206
C104	100M/6.3VT	
C105	100nF	SMD 1206
C106	100nF	SMD 1206
C108	100nF	ceramic capacitor
CON1	DB9	Cannon 9 pin FEMALE
CON2(*)	RJ-45	UTP RJ-45
CON3	lišta RM 2,54 13 pinů	
CON4	PWCON	power socket
D1		LED diode
D2		LED diode
D3		LED diode
D4		LED diode
D5	1N4001	diode
D6	1N4101	diode
D7	1N4001	diode
IC1	8052	8052 CPU / 89S8252
IC2(*)	RTL8019AS	network controller
IC3	MAX232	TTL/RS-232 convertor
IC4	7805	V-REG
IC5	AT24C512	serial EEPROM
Q1	14.74 MHz	QUARTZ
Q2(*)	20MHz	QUARTZ
R1	49R9	SMD 0805
R2	49R9	SMD 0805
R3	10R	SMD 0805
R4	10R	SMD 0805
R5	1K0	SMD 1206
R6	1K0	SMD 1206
R7	27K	SMD 1206
R8	10K	SMD 1206
R9	10K	SMD 1206
R10	1K0	SMD 1206
R11	1K0	SMD 1206
TR1(*)	FC-22	10BaseT isolation transformer / filter

WWW8052 - rozpis materiálu dle hodnot

Qty	Value	Parts	
2	10R	SMD 0805	R3, R4
2	49R9	SMD 0805	R1, R2
4	1K0	SMD 1206	R5,R6,R10,R11
2	10K	SMD 1206	R8, R9
1	27K	SMD 1206	R7
2	33pF	SMD 1206	C7,C8,C9,C10
2	180pF	SMD 0805	C5, C6
2(*)	1nF/2kV		C1, C2
2	10nF	SMD 0805	C3, C4
1	10nF	SMD 1206	C16
4	100nF	SMD 1206	C102,C103, C105,C106
1	100nF	ceramic cap.	C108
5	22M/16VT		C11, C12, C13, C14, C15
1	47M/25VT		C101
1	100M/6.3VT		C104
4		LED diode	D1, D2, D3, D4
2	1N4001	diode	D5, D7
1	1N4101	diode	D6
1	14.74 MHz	QUARTZ	Q1
1(*)	20MHz	QUARTZ	Q2
1	8052	CPU 8052...	IC1
1(*)	RTL8019AS	ethernet controller	IC2
1	MAX232	TTL/RS232 convertor	IC3
1	7805	V-REG	IC4
1	AT24C512	serial EEPROM	IC5
1(*)	FC-22	transformer filter	TR1
1	DB9	connector Cannon9 Female	CON1
1(*)	RJ-45	RJ45	CON2
1	lámací lišta	RM 2,54 13 pinů	CON3
1	PWCON	power connector	CON4
1	PCB	WWW8052	

Filozofie WWW8052

Základní principy WWW8052 demonstrujeme na příkladu verze umožňující jednoduchou konverzi komunikace po sériové lince na komunikaci pomocí protokolu TCP/IP přes Ethernet a zpět. Co tedy tato verze obsahuje:

Jednoduchý WWW server, naslouchající na portu 80, umožňující konfiguraci WWW8052

Základní "Telnet" server, naslouchající na portu 23, sloužící k přenosu dat z otevřeného TCP/IP spojení do sériové linky a zpět.

Základní "Telnet" klient umožňující, není-li k dispozici otevřené TCP/IP spojení uvedené v předchozím bodě, navázat TCP/IP spojení.

WWW8052 v sobě obsahuje pro každý port jeden samostatný TCP stack. Současně je proto možné mít dvě otevřená spojení, po jednom pro každý port.

WWW server po pasivním navázání spojení počká na zaslání potřebné minimální části http požadavku (požadavek je možno posílat i telnetem, tj. znak po znaku). Po vyhodnocení jména požadované stránky je požadovaná stránka doplněna http hlavičkou a po případném rozdělení na části, odeslatelné přes Ethernet, vyslán klientovi. Poslední odeslaný paket nese příznak ukončení spojení a na straně serveru okamžitě uvolňuje prostředky pro další spojení. To znamená, že server neumožňuje tzv. ersistent connection, či KeepAlive. Server rovněž neumožňuje, vzhledem k nemožnosti uložení

dynamicky generovaných stránek do paměti (kterou nemá k dispozici), zopakování špatně odeslaného paketu. V případě chyby je nutno zopakovat celý požadavek.

Telnet server umožňuje jak pasivní spojení tak, pokud to má povoleno, aktivní spojení. Na rozdíl od WWW serveru, plně podporuje opakování špatně odeslaných paketů. Při vyčerpání paměti protistranou nepotvrzených a tím pádem v paměti držených odeslaných dat však dochází k zpomalení komunikace. Spojení je ukončeno automaticky po vypršení cca 50 sekund od přijetí posledního potvzení.

WWW8052 využívá pro urychlení některých síťových operací, místo pomalého sestavování po částech, předem připravených paketů, uložených v paměti EEPROM procesoru. Při změně IP konfigurace je nutno tyto pakety rekonfigurovat.

WWW8052
Vývojový systém

WWW8051.HEX - výsledný soubor, generováno pomocí makefile
WWW8051.M51 - mapa z linkeru, generováno pomocí makefile
makefile - hlavní soubor pro MAKE
ether - soubor se zadáním parametrů MAC adresy
ETHER.INC automaticky generováno
ip soubor se zadáním parametrů IP adresy, masky gateway, ...
IP.INC automaticky generováno
http.a51 zdrojový kód, primitivní http server, popsán v části http server
HTTP.OBJ přeložený soubor
htmlpage.a51 zdrojový kód, html stránky a cgi procedury, příklady popsány v části WWW setup
HTMLPAGE.OBJ přeložený soubor
www8051.a51 zdrojový kód, hlavní modul
WWW8051.OBJ přeložený soubor

Directory: ***bin***

ip2hex.pl Perl skript pro generování IP.INC
mac2hex.pl Perl skript pro generování ETHER.INC
perl.exe GNU Perl 4.0.1.8 pro Windows NT (resp. Win32), používaný pro běh .pl skriptů
read.bat Batch soubor pro čtení procesoru AT89S8252 pomocí ISP programátoru
prog.bat Batch soubor pro programování procesoru AT89S8252 pomocí ISP programátoru
PROGE.EXE DOS verze obslužného programu ISP programátoru (*) pro programování EEPROM v AT89C8252,..
PROGF.EXE DOS verze obslužného programu ISP programátoru (*) pro programování FLASH v AT89C8252,..
PROGFUSE.EXE DOS verze obslužného programu ISP programátoru (*) pro programování pojistek v AT89C8252,..
PROGLOCK.EXE DOS verze obslužného programu ISP programátoru (*) pro programování zámků v AT89C8252,..
READE.EXE DOS verze obslužného programu ISP programátoru (*) pro čtení EEPROM v AT89C8252,..
READF.EXE DOS verze obslužného programu ISP programátoru (*) pro čtení FLASH v AT89C8252,..
bin2hex.exe Konvertor BIN na HEX (*)
hex2bin.exe Konvertor HEX na BIN (*)

Directory: ***doc***

GPLCZ.HTM GNU General Public Licence - český překlad
GNU.JPG GNU logo

Directory: ***lib***

8019.inc definice Ethernet, IP, TCP,... konstant, registrů RTL8019, ...
param.inc základní definice HW a SW prostředí pro překlad
pmacro.inc makra pro generování Pkódu, popsáno v Struktura P-kódu a jeho interpretr
KERNEL.LIB TCP/IP knihovna, část popsána v P-instrukce pro základní síťové operace
WWW8051.LIB knihovna, sestavena pomocí MAKE z následujících částí:

makefile hlavní soubor pro MAKE knihovny www8051.lib
clone.a51 zdrojový kód, klonování Ethernet & IP adres z EEPROM do připravených EEPROM packetů
CLONE.OBJ přeložený soubor
clonep.a51 zdrojový kód, klonování IP point adresy z EEPROM do připravených EEPROM packetů
CLONEP.OBJ přeložený soubor
drams.a51 zdrojový kód, Dump RAM to Serial
DRAMS.OBJ přeložený soubor
i2c.a51 zdrojový kód, I²C komunikace
I2C.OBJ přeložený soubor
outhexb.a51 zdrojový kód, princip činnosti je popsán v PREVOD 8mi bitovych cisel do ASCII
OUTHEXB.OBJ přeložený soubor
outstrg.a51 zdrojový kód
OUTSTRG.OBJ přeložený soubor
packet.a51 zdrojový kód, modifikovaná verze bez callback(ProcessEthPacket)
popsán v části Inicializace driveru, Vysílání paketů, Příjem paketů
PACKET.OBJ přeložený soubor
paritm.a51 zdrojový kód, popsán v části P-instrukce pro základní matematické operace a větvení programu
PARITM.OBJ přeložený soubor
pcode.a51 zdrojový kód, popsán v části Struktura P-kódu a jeho interpretr
PCODE.OBJ přeložený soubor
pdebug.a51 zdrojový kód, debug výpis R6R7
PDEBUG.OBJ přeložený soubor
serial.a51 zdrojový kód, popsán v části Obsluha sériového kanálu
SERIAL.OBJ přeložený soubor

(*) Zde použitá část programů pro ISP programování procesoru AT89S8252 a dalších pochází z WWW firmy Atmel (APROGDOS HEX2BIN BIN2HEX). Na tomto www (především v jeho části AVR) je možno najít i specifikace použitého protokolu a zapojení programátoru. Konstrukci programátoru se věnuje v českém jazyce server www.hw.cz, kde je jej možno koupit i jako stavebnici.
Základní GNU vývojový systém neobsahuje překladač assembleru pro procesor I8051. Předpokládá použití překladače kompatibilního s překladačem Intel, popř. KEIL. MAKE soubory předpokládájí následující programy:

Directory: ***bin***

A51.EXE MCS-51 MACRO ASSEMBLER A51
L51.EXE L51 LINKER/LOCATOR
LIB51.EXE MCS-51 LIBRARY MANAGER LIB51
OH51.HEX MCS-51 OBJECT TO HEX FILE CONVERTER

U některých verzí překladače dochází při překladu k dost zajímavým chybám, vzhledem k nim je v sekci download opravená verze vývojového systému. Původní byla přeložena špatnou verzí překladače. Chyba je demonstrována na následujícím fragmentu výpisu:

```
C014 fail_ip_mask  equ  $ - 2000h + 4
C014 fail2_ip_mask equ ($ - 2000h) + 4
E00A flash_ip_mask equ  $ - (2000h - 4)
```

Proměnná $ je v době výpočtu rovna 0006. Některé verze překladačů pro všechny 3 možnosti zápisu správně spočítají E00A, některé však spočítají výše uvedené hodnoty. Bylo proto nutno zdrojové texty přepsat do tvaru na třetím řádku.

Potřebný je i program MAKE, ať již GNU či autorem používaná MS verze, dodávaná s překladači jazyka C (MSC ver. 7.0):

Directory: ***bin***

NMAKE.EXE Microsoft (R) Program Maintenance Utility

GNU vývojový systém je možno stáhnout z domovské stránky celého projektu **http://8052.lphard.cz** ve formě RAR souboru. Všechny uvedené zdrojové texty podléhají licenci GNU. V případě požadavku na šíření pod jinou licencí, např. pro použití v komerčním produktu, kontaktujte autora.

Packet Driver RTL8019AS
část 1 - inicializace driveru

Obvod RTL8019AS v sobě obsahuje vše potřebné pro příjem a vysílání paketů dle standartu Ethernet. Přijaté pakety, či paket(y) připravené k vysílání jsou uloženy v paměti čipu. Tato paměť je navenek přístupná pouze pomocí DMA kanálu. Celý čip je ovládán přes čtyři banky registrů. Číslo registru je možno zadávat přímo, pomocí adresových vývodů A0..A4. Pro řízení a čtení stavu čipu je použito prvních 16 registrů 0..15, další osmice 16..23 reprezentuje ve skutečnosti pouze jediný datový registr sloužící ke komunikaci s DMA. Zbývající osmice 24..31 je rovněž ve skutečnosti jediným registrem a slouží k programovému resetu čipu. Jednotlivé banky registrů není možno adresovat přímo, ale pouze pomocí 2 bankovacích bitů v registru 0, který je přístupný ve všech čtyřech bankách registrů.
Jak vlastně vypadá ethernet paket vysílaný, či přijímaný obvodem RTL8019? Každý paket se skládá celkem ze 60 až 1514 bytů.

Destination Address	MAC Adresa příjemce	6 bytů
Source Address	MAC Adresa odesílatele	6 bytů
Length	Délka paketu	2 byty
DATA	Data	min. 46 max.1500 bytů

Této struktuře paketu je podřízena i struktura registrů celého obvodu. V jednotlivých registrech obvodu, je možno nastavit, jak adresu příjemce (registry EN1_PHYS), tak adresu odesílatele (registry EN1_MULT). Toto nastavení je určeno k filtraci přijímaných paketů ukládaných do paměti obvodu. Obvyklé nastavení je takové, že obvod přijímá pakety s adresou příjemce rovnou MAC adrese čipu a s libovolnou adresou odesílatele. Pro vysílání je nutno nastavit délku vysílaného paketu do registrů EN0_RCNTLO (méně významný byte) a EN0_RCNTHI (více významný byte).

Přístup do paměti adaptéru je, jak již bylo uvedeno, možný pouze přes DMA kanály. Obvyklá organizace paměti a z ní vyplývající počáteční nastavení DMA registrů je následující:

00h..FFh WORD
konfigurační data z EEPROM 9346
4000h..45FFh BYTE
buffer pro vysílání
4600h..5FFFh BYTE
kruhový přijímací buffer

Celá paměť je z hlediska přístupu rozdělana na jednotlivé stránky o velikosti 256 byte. Je-li paket menší než je násobek 256, je zarovnán na celou stránku. Přijímací část paměti je organizována

jako kruhový buffer. Činnost kruhového bufferu řídí celkem 4 registry. První dvojice registrů, EN0_STARTPG a EN0_STOPPG, reprezentuje hranice tohoto kruhového bufferu. Vyjdeme-li z obvyklé organizace paměti, bude jejich nastavení následující:

EN0_STARTPG 46h (4600h)
adresa první stránky
EN0_STOPPG 60h (6000h)
adresa poslední stránky + 1

Do tohoto bufferu ukazuje dvojice ukazatelů, reprezentovaných registry EN0_BOUNDARY a EN1_CURPAG. První z nich, EN0_BOUNDARY, je registrem chránícím data nepřečtená z kruhového bufferu. Přijímač nesmí ukládat data do paměti nad tuto hranici. Při přečtení dat z obvodu je proto nutno jeho obsah nastavit vždy na adresu stránky o jednu menší než je adresa nepřečtených dat. Druhý z nich, EN1_CURPAG je registrem, určujícím kam se budou zapisovat přijímaná data. Označíme-li obsazené místo v paměti "číslem" paketu a volné např. znakem -, může situace vypadat například takto:

```
46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52
 .  .  .  .  .  .  .  .  .  .  .  .  .

53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F
 1  1  1  2  2  2  3  3  4  4  4  .  .
EN0_BOUNDARY=52h  EN1_CURPAG=5Eh
```

Zpracujeme-li např. tři přijaté pakety a mezitím dva další přijmeme bude situace následující:

```
46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52
 5  6  .  .  .  .  .  .  .  .  .  .  .

53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F
 .  .  .  .  .  .  .  .  .  .  .  5  5
EN0_BOUNDARY=5Ah  EN1_CURPAG=48h
```

A jak vypadá počáteční nastavení ? Obvykle používané je:

EN0_BOUNDARY 46h (4600h)
adresa první stránky
EN1_CURPAG 47h (4700h)
adresa druhé stránky

Registr EN1_CURPAG se musí inicializovat na hodnotu o jedničku vyšší, než je nastavení registru EN0_BOUNDARY z důvodů ochrany stránky. Stránka nastavená v EN0_BOUNDARY je stránkou mezní a při jejím dosažení obvod přerušuje přenos do paměti (chybné nastavení EN1_CURPAG = EN0_BOUNDARY je uvedeno např. i v AN-874 od NS). S těmito výchozími znalostmi lze dojít k následujícímu fragmentu kódu:

```
;*************************************************************
;** Reset and init card
;**
```

Vygenerujeme puls na vstupu RSTDRV (Reset) obvodu RTL8019AS.

```
network_init:
  SETB    RESETBIT  ;Reset ISA bus
  ACALL   waitbus
  CLR     RESETBIT  ;End of Reset ISAbus
  ACALL   waitbus
```

Za HW resetem následuje softwarový reset RTL8019AS, vyvolaný sekvencí čtení s následujícím zápisem na adresu resetovacího registru RTL čipu.

```
  ISAIN   EN_RESET
          ;Reset Port (for all pages)
  ACALL   longpause
  ISAOUTA EN_RESET
          ;Reset Port (for all pages)
```

Do čipu je následně pošleme příkaz stop a počkáme dobu nutnou k dokončení případného vysílání (i když má to po HW resetu smysl ?, nicméně takhle je to idioten sichr).

```
  ISAOUT  EN_CMD,#EN_PAGE0+
                 EN_NODMA+EN_STOP
          ;Remote DMA, Stop and reset the
                   chip
  ACALL   longpause
```

Nemáme požadavek na DMA, délku pro DMA přenos nastavíme na nulu.

```
  BEGINPORTDATA
  PORTDATA EN0_RCNTHI,0 ;MSB Remote byte
                   count reg
  PORTDATA EN0_RCNTLO,0 ;LSB Remote byte
                   count reg
```

Obvod přepneme do monitorovacího a smyčkového módu, kdy nepřijímá data z linky a umožňuje diagnostiku. Nastavíme přenos po bytech.

```
  PORTDATA EN0_RXCR, ENRXCR_MON
          ;RX configuration reg
          ;Monitor mode (no packets rcvd)
  PORTDATA EN0_TXCR, ENTXCR_LOOP
          ;TX configuration reg
          ;set internal loopback mode
  PORTDATA EN0_DCFG,ENDCFG_FT10+
                   ENDCFG_BMS+ENDCFG_WTS
          ;fifo treshold, Normal
          operation,
          ;word/byte transfer mode
          selection
  ENDPORTDATA
```

Pokud je k obvodu připojena EEPROM pamět 9346, tak z ní přečteme MAC adresu adaptéru. Tato adresa je uložena ve "stínové" RAM od adresy 0h a lze ji přečíst standartním postupem pomocí DMA. Nicméně se nebere v úvahu bytový režim a z této oblasti jsou datové byty zdvojovány a při čtení je nutno proto zpracovat pouze každý druhý byte.

```
  ACALL   get_board_data
```

Nastavíme výchozí parametry kruhového bufferu.

```
  BEGINPORTDATA
  PORTDATA EN_CMD, EN_PAGE0+EN_NODMA
                   ;Remote DMA
  PORTDATA EN0_STARTPG, RX_START_PG
          ;Starting page of ring buffer
          ;First page of RX Ring
  PORTDATA EN0_BOUNDARY, RX_START_PG
          ;Boundary page of ring buffer
  PORTDATA EN0_STOPPG,NE_STOP_PG
          ;Ending page +1 of ring buffer
          ;End Page = Last page + 1 of RX
          Ring
```

Vynulujeme případné přerušení, nastavení masky je spíše symbolické, použité zapojení nepoužívá přerušení a proto jej pro zviditelnění zakážeme.

```
  PORTDATA EN0_ISR, -1
          ;Interrupt status reg
  PORTDATA EN0_IMR, 0
          ;Interrupt mask reg =
          Disable All Interrupt
```

Nastavíme do registrů vlastní MAC adresu.

```
  PORTDATA EN_CMD, EN_PAGE1+EN_NODMA
          ;Page 1, Remote DMA
  PORTDATA EN1_PHYS+portind+0,
                   my_ether+0
          ;This board's physical enet
          addr
  PORTDATA EN1_PHYS+portind+1,
                   my_ether+1
  PORTDATA EN1_PHYS+portind+2,
                   my_ether+2
  PORTDATA EN1_PHYS+portind+3,
                   my_ether+3
  PORTDATA EN1_PHYS+portind+4,
                   my_ether+4
  PORTDATA EN1_PHYS+portind+5,
                   my_ether+5
```

Multicast registry v sobě obsahují 64 bitů, povolujících 64 skupin skupin multicast adres. Jednotlivé skupiny multicast adres jsou určeny pomocí CRC generátoru. Z vypočteného CRC MAC adresy příjemce je nejvyšších šest bitů použito jako index do bitové tabulky, uložené ve výše uvedených registrech. Jedničkový bit na dané pozici udává, že se případný multicast přijme. Nicméně příjem multicast paketů musí být povolen v kongfiguračním registru přijímače. Implicitně nastavíme povolení všech skupin multicast adres.

```
  PORTDATA EN_CMD,
          EN_PAGE1+EN_NODMA+EN_STOP
          ;Page 1, Remote DMA, Stop and
          reset the chip
  PORTDATA EN1_MULT+0,0FFh
          ;Multicast filter mask
          array (8 bytes)
  PORTDATA EN1_MULT+1,0FFh
  PORTDATA EN1_MULT+2,0FFh
  PORTDATA EN1_MULT+3,0FFh
  PORTDATA EN1_MULT+4,0FFh
  PORTDATA EN1_MULT+5,0FFh
  PORTDATA EN1_MULT+6,0FFh
  PORTDATA EN1_MULT+7,0FFh
  PORTDATA EN_CMD,EN_PAGE0+
                   EN_NODMA+EN_START
          ;Remote DMA, Start the chip,
          clear reset
```

Inicializujeme ukazatel, odkud se začnou ukládat přijaté pakety.

```
  PORTDATA EN_CMD, EN_PAGE1+
                   EN_NODMA+EN_STOP
          ;Page 1,
          ;Remote DMA, Stop and reset
          the chip
  PORTDATA EN1_CURPAG, RX_CURR_PG
;Current memory page = RX_CURR_PG
  PORTDATA EN_CMD, EN_PAGE0+
                   EN_NODMA+EN_START
          ;Remote DMA, Start the chip,
          clear reset
```

Nastavíme standartní režim vysílače.

```
  PORTDATA EN0_TXCR, 0
;TX configuration reg = Normal Operation
```

Nastavíme standartní režim přijímače, nepřijímají se chybné pakety, jsou povoleny broadcasty, nejsou povoleny multicasty.

```
  PORTDATA EN0_RXCR, ENRXCR_BCST
          ;RX configuration reg =
          Accept broadcasts
```

Vynulujeme případné přerušení

```
  PORTDATA EN0_ISR, -1
;Clear the pending Interrupt /* sichr */
  ENDPORTDATA
```

Nakonec nastavíme vnitřní proměné paket driveru, na stejnou hodnotu jako ukazatel, odkud se budou ukládat přijaté pakety.

```
  MOV     next_packet,#RX_CURR_PG
  RET
```

Packet Driver RTL8019AS
část 2 - vysílání paketů

Vysílání paketů je v tomto driveru řešeno velmi jednoduše. Vzhledem k tomu, že obvod neobsahuje FIFO pro celý vysílaný paket a vysílá paket pomocí DMA kanálu z paměti, je nutno před sestavováním dalšího paketu do paměti čipu RTL počkat na skončení předchozího vysílání. Druhou možností by bylo použít dvou vysílacích bufferů a střídavě z jednoho vysílat a do druhého sestavovat paket. Nicméně ani tak se někdy nevyhneme čekání na dokončení předchozího vysílání. To lze zabezpečit např. pomocí následujícího kódu:

```
;*************************************************************
;***      S E N D   P A C K E T       **
;*************************************************************
;*************************************************************
;** wait for end of previous transmit
;**
tx_wait:
```

Nejprve pro ochranu před zamrznutím nastavíme timeout na dobu danou časem za který je paket buď přenesen, nebo zahozen jako nedoručitelný.

```
  MOV     DPTR,#-40982
;max coll time = 1024*7 Ethernet slot
                   units (51.2us)
tx_wait1:
```

Pak cyklicky čteme hodnotu stavového bitu, udávajícího, zda obvod vysílá. Pokud obvod nevysílá, cyklus ukončíme.

```
  ISAIN   EN_CMD
  JNB     BIT_TRANS,end_tx_wait
  INC     DPTR
  MOV     A,DPH
  ORL     A,DPL
```

Cyklus ukončíme rovněž při vypršení timeoutu.

```
  JNZ     tx_wait1
;;Transmit, with error
;; set errors flag...
end_tx_wait:
  RET
```

Pokud obvod nevysílá, sestavíme do paměti obvodu RTL paket k vysílání, včetně předepsané ethernetovské hlavičky, udávající MAC adresu příjemce, MAC adresu odesílatele a délku paketu. Sestavený paket vyšleme následující procedurou.

```
;*************************************************************
;** send packet
;**
;input:   R6R7      Transmit Byte Count
send_pkt:
```

Nejprve pro jistotu počkáme na ukončení předchozího vysílání

```
; Wait for end of previous transmit
  ACALL   tx_wait
```
Poté zkontrolujeme, jak skončilo předchozí vysílání. Pokud skončilo s chybou je možno nastavit příslušné bity pro softwarové ošetření chyby při vysílání.
```
; Check for recent TX completions in the
interrupt status register
  ISAIN   EN0_ISR   ;Interrupt status reg
  ANL     A,#ENISR_TX_ERR+ENISR_TX
          ; Pending Tx interrupts?
  JZ      tx_idle   ; No, Go on new Tx
; JNB     BITISR_TX_ERR, XmitFrameNoErr
;; Transmit, with error
;; set errors flag...
;XmitFrameNoErr:
```
Vysílalo-li se přečteme podrobnější informace o ukončeném vysílání a případně je uložíme či jinak zpracujeme.
```
  ISAIN   EN0_TSR
          ; Read Transmit status reg
;; save transmit status...
```
Na závěr zpracování předchozího vysílání potvrdíme zpracování přerušení od vysílače.
```
          ; Acknowledge the TX interrupt
  ISAOUT  EN0_ISR, #ENISR_TX_ERR+ENISR_TX
          ; Interrupt status reg
          ; Set Transmitter Interrupt,
          with/no error
```
Před zahájením zkontrolujeme povolenou délku paketu. Nejprve zkontrolujeme, zda paket nepřesahuje 1500 bytů dat + 14 bytů hlavičky. Delší pakety zahodíme.
```
tx_idle:
  MOV     A,R7      ;Is packet too large?
  ADD     A,#LOW(-GIANT-1)
  MOV     A,R6
  ADD     A,#HIGH(-GIANT-1)
  JNC     XmitNoGiant
;; set errors flag...
  RET
```
Následuje kontrola zda paket není kratší než 46+14 bytů.
```
XmitNoGiant:
  MOV     A,R6
          ; Is the frame long enought ?
  JNZ     XmitNoRunt
  MOV     A,R7
  ADD     A,#-RUNT
  JC      XmitNoRunt
```
Je-li kratší, natáhneme jej na minimální povolenou délku. (Paket se automaticky doplní "smetím" z předchozího paketu, není doplňován např. nulami).
```
;;MOV     R6,#HIGH(RUNT)
  ; Stretch frame to minimum allowed
  MOV     R7,#LOW(RUNT)
```
Na závěr sdělíme obvodu RTL délku vysílaného paketu a jeho umístění v paměti. Poté odstartujeme vysílání.
```
XmitNoRunt:
  BEGINPORTDATA
  PORTDATA EN0_TCNTHI+portind,AR6
          ; High byte of tx byte count
  PORTDATA EN0_TCNTLO+portind,AR7
          ; Low  byte of tx byte count
  PORTDATA EN0_TPSR,NE_START_PG
          ; Transmit starting page
          ; First page of TX buffer
  PORTDATA EN_CMD, EN_PAGE0+
          EN_NODMA+EN_TRANS+EN_START
          ; Remote DMA, Transmit a frame,
          ; Start the chip, clear reset
  ENDPORTDATA
  RET
```

Packet Driver RTL8019AS
část 3 - příjem paketů

Příjem paketů je nesporně nejsložitější částí celého driveru. Nejprve ukážeme proceduru, která čte hlavičku přijatého paketu včetně stavových informací.
```
;*********************************************************
;***    R E C E I V E   P A C K E T
;*********************************************************
;*********************************************************
;** read header of receive packet
;**
ReadPacketHeader:
  mov     R2,#high(SIZE_OF_8019_HDR+
                   SIZE_OF_ETH_PKT_HDR)
  mov     R3,#low(SIZE_OF_8019_HDR+
                   SIZE_OF_ETH_PKT_HDR)
                   ;R2R3 Byte Count
; MOV     R4,#...
          ;R4R5 Remote Start Address
  MOV     R5,#0
  MOV     R1,#rcv_hdr   ;R1 store pointer
  MOV     DPTR,#icall_save
  ;pointer to procedure BYTE(@R1) save
  AJMP    DMARead
```
Hlavička přečtená z obvodu RTL vypadá následovně:

rcv_hdr	1 byte
Received frame status (Status přijatého paketu)	
rcv_hdr_nxt_pg	1 byte
Page after this frame (Ukazatel na následující paket)	
rcv_hdr_size	2 byte
Length of this frame (Délka paketu)	
eth_pkt_hdr_dest	6 bytů
Destination Address (MAC Adresa příjemce)	
eth_pkt_hdr_src	6 bytů
Source Address (MAC Adresa odesílatele)	
eth_pkt_hdr_type	2 byty
Length/Type (Délka paketu/Typ paketu)	

Ukažme tedy jak vypadá zpracování přijatého paketu.
```
;*********************************************************
;** receive packet
;**
rcv_pkt:
```
Nejprve zjistíme, zda je co zpracovávat.
```
  ISAIN   EN0_ISR
          ; Get pending interrupts
  JNZ     ISRtestOverrun
  RET               ; Ret if none
```
Otestujeme zda nedošlo k poškození dat v bufferu, vlivem jeho přetečení.
```
ISRtestOverrun:
          ; Was there an overrun ?
  JB      BITISR_OVER,recv_overrun
          ; Go if so
  AJMP    recv_no_overrun
          ; Go if not
```
Buffer přetekl, je nutno ošetřit přetečení.
```
;*********************************************************
;** receive overrun
;**
```
Nejprve si uschováme aktuální stav obvodu.
```
recv_overrun:
  ISAIN   EN_CMD
          ; read Chip's command register
  PUSH    Acc       ; and save it
```
A poté zastavíme obvod.
```
;; Stop the NIC
  ISAOUT  EN_CMD, #EN_PAGE0+
                  EN_NODMA+EN_STOP
; Remote DMA, Stop and reset the chip
```
A vyčkáme na dokončení příjmu či vysílání paketu.
```
;; Wait 1.6ms for the NIC to stop
transmitting or receiving a packet.
National
;; says monitoring the ISR RST bit is
not reliable, so a wait of the maximum
;; packet time (1.2ms) plus some padding
is required.
  ACALL   longpause
```
Poté začneme se "vkřišením" obvodu, nastavením délky přenosu na nulu.
```
;; Reset RBCR[01] back to zero as per
magic incantation.
  BEGINPORTDATA
  PORTDATA EN0_RCNTHI,0
          ; MSB Remote byte count reg
  PORTDATA EN0_RCNTLO,0
          ; LSB Remote byte count reg
  ENDPORTDATA
```
Otestujeme, zda obvod v okamžiku zastavení vysílal, a pokud ano, zjistíme, zda bylo v pořádku dokončeno. Pokud ne tak si to poznamenáme a později zopakujeme přenos.
```
;; check the saved state of the EN_TRANS
bit in the command register
  CLR     flagResend
  POP     Acc
          ; pop Chip's command register
  JNB     BIT_TRANS,rcv_ovr_loopback
;; Transmitter was running, see if it
finished or died
  ISAIN   EN0_ISR
          ; Get pending interrupts
  ANL     A,#ENISR_TX_ERR+ENISR_TX
          ; Did the transmitter finish?
  JNZ     rcv_ovr_loopback
; one will be set if TX finished
; Transmitter did not complete, remember
to resend the packet later.
  SETB    flagResend
rcv_ovr_loopback:
```
Do doby dokončení celé záchrané operace, je nutno obvod zapnout do smyčkového módu, kdy nepřijímá data z linky. Po té je možno obvod opět spustit.
```
;; Have to enter loopback mode and then
restart the NIC before you are
;; allowed to slurp packets up off the
ring
  BEGINPORTDATA
  PORTDATA EN0_TXCR,ENTXCR_LOOP
; set internal loopback mode
  PORTDATA EN_CMD, EN_PAGE0+
          EN_NODMA+EN_START
; Start the chip running again
```
Zkontrolujeme, zda je v bufferu nějaký paket, porovnáním adresy kam bude obvod ukládat data a adresy kde je následující paket očekáván tímto programem
```
;; Verify that there is really a packet
to receive by fetching the current
;; page pointer and comparing it to the
next packet pointer.
  PORTDATA EN_CMD, EN_PAGE1+EN_NODMA
                   ; Set Page 1
  ENDPORTDATA
  ISAIN   EN1_CURPAG
                   ; Current memory page
  MOV     R4,A
  ISAOUT  EN_CMD, #EN_PAGE0+EN_NODMA
                   ; Page 1, Remote
  MOV     A,next_packet
  XRL     A,R4
          ; Check if buffer empty
  JNZ     rcv_ovr_rx_one
          ; O.K. get the NIC header
```
V paměti není žádný paket, přeskoč verifikaci.
```
; NO PACKET IN THE RING AFTER AN OVW
INTERRUPT??? Can this ever happen?
; YES! if overrun happend between a
receive interrupt and the when the
; current page register is read at the
start of recv_frame.
  AJMP    rcv_ovr_empty
          ; Current Page == next_packet
```
Přečteme hlavičku přijatého paketu.
```
rcv_ovr_rx_one:
; MOV     R5,#0
; R4R5 Remote Start Address Register
  PUSH    AR4
  ACALL   ReadPacketHeader
  POP     AR4
```
Začneme kontrolovat přijatý paket ve frontě na jeho poškození.
```
  MOV     A,rcv_hdr
          ; Get the buffer status byte
          ;Received a good packet ?
```
Nejprve zjistíme, zda má paket ve své hlavičce uvedeno zda je správně přijatý.
```
  JNB     BITRSR_RXOK, rcv_ovr_ng
          ; No, received debris
```
Pokud ano, zkontrolujeme ukazatel na další paket ve frontě zda je v povoleném rozsahu.
```
; EVEN if the NIC header status is OK, I
have seen garbaged NIC headers, so
; it doesn't hurt to range check the
next packet pointer here.
  MOV     A,rcv_hdr_nxt_pg
          ; pointer to next packet
  CJNE    A,#RX_START_PG,$+3
          ;; First page of RX Ring
  JC      rcv_ovr_ng
          ;; = NE_STOP_PG
;;invalid pointer, Above the top
```
Ukazatel je v pořádku, proto jej uložíme a zpracujeme přijatý paket.
```
  MOV     next_packet,A
          ;save pointer to next packet
  MOV     curr_recv,R4
          ;save pointer to current packet
  LCALL   ProcessEthPacket
  SJMP    rcv_ovr_ok
```
Pokud v hlavičce není uvedeno správné přijetí paketu, popř. je nesprávný ukazatel na další paket, zahodíme všechny pakety v paměti.
```
rcv_ovr_ng:
;
; HAD TO ADD ERROR RECOVERY HERE. TO
BLINDLY PROCEED AND ASSUME THE NEXT;
PACKET POINTR FROM THE NIC HEADER IS
VALID IS INVITING DISASTER.
;
; Error recovery consists of killing and
restarting the NIC. This drops all
; the packets in the ring, but thats
better than winding up in the weeds!
;
```

```
; Instead copy the last known current page
pointer into the next packet pointer
; which will result in skipping all the
packets from the errored one to where
; the NIC was storing them when we
entered this ISR, but prevents us from
; trying to follow totally bogus next
packet pointers through the card RAM
; space.
  MOV     A,R4      ; Current memory page
;;MOV     next_packet,A
```
Poté ukončíme proceduru přijetí paketu potvrzením přerušení a instrukcí RET.
```
  AJMP    recv_frame_break1
;save next_packet...
```
Přitý paket je zpracován, popř. nebyl přijat žádný paket. Zbývá nastavit mezní registr.
```
rcv_ovr_ok:
rcv_ovr_empty:
  MOV     A,next_packet
          ; Grap the next packet pointer
  DEC     A         ; Back up one page
  CJNE    A,#RX_START_PG,$+3
          ;; Did it wrap?
  JNC     BoundaryOK; >= RX_START_PG
  MOV     A,#NE_STOP_PG-1
          ;; Yes, back to end of ring
BoundaryOK:
  ISAOUTA EN0_BOUNDARY
          ;Boundary Register
```
Smazat příznak přepsání přijímací fronty.
```
  BEGINPORTDATA
; Clear the OVW bit in the ISR register.
  PORTDATA          EN0_ISR, ENISR_OVER
;Interrupt status reg
;Receiver overwrote the ring
```
Vypnou režim smyčky.
```
; Take the NIC out of loopback
  PORTDATA          EN0_TXCR, 0
                    ;Normal Operation
  ENDPORTDATA
```
A pokud bylo zastaveno probíhající vysílání, tak jej obnovit.
```
; Any incomplete transmission to resend?
  JNB     flagResend,recv_frame_break
                    ;No
; Yes, restart the transmission
  ISAOUT  EN_CMD,EN_TRANS+
          EN_NODMA+EN_START
          ;Start the transmitter
```
Poté ukončíme proceduru přijetí paketu potvrzením přerušení a instrukcí RET.
```
  AJMP    recv_frame_break
```
Buffer nepřetekl, zkontrolujeme zda došlo při příjmu k chybám.
```
recv_no_overrun:
  JNB     BITISR_COUNTERS,
          NoReadTallyCounters
```
Pokud byly detekovány chyby, tak přečteme jejich počet.
```
;*************************************************************
;** some errors, read Tally Counters
;**
  PUSH    Acc
  MOV     ADDRPORT,#EN0_COUNTER0
                    ;CNTR0..2
;CNTR0 Frame Alignment Error Tally
Counter Register
;CNTR1 CRC Error Tally Counter Register
;CNTR2 Missed Packet Tally Counter
Register
  ISAREAD
  INC     ADDRPORT
  ISAREAD
  INC     ADDRPORT
  ISAREAD
  ISAOUT  EN0_ISR, #ENISR_COUNTERS
          ;Interrupt status reg
  ;ReSet CNT (Error Tally Counters)
  POP     Acc
; NoReadTallyCounters:
```
Pokud byl přijat paket, ať již vpořádku, či s chybou zpracujeme jej.
```
  JB      BITISR_RX, PacketReceivedOK
  JB      BITISR_RX_ERR, PacketReceived
```
V opačném případě ukončíme proceduru přijetí paketu potvrzením přerušení a instrukcí RET.
```
  AJMP    recv_frame_break
;*************************************************************
;** test frame pointer
;**
```
Podtvrdíme zpracování přerušení od přijatého paketu
```
PacketReceived:
PacketReceivedOK:
  ISAOUT  EN0_ISR, #ENISR_RX_ERR +
                    ENISR_RX
; Clear those request
```
Poté si uschováme ukazatel, pokud je buffer zaplněn přijatými pakety
```
;; Get the rx page (incoming packet
pointer)
  ISAOUT  EN_CMD, #EN_PAGE1+
                    EN_NODMA+EN_START
          ; Switch to page 1 registers
  ISAIN   EN1_CURPAG
          ; Get current page of rcv ring
  PUSH    Acc       ; ans save it
  ISAOUT  EN_CMD, #EN_PAGE0+
                    EN_NODMA+EN_START
          ; Back to page 0 registers
```
A v cyklu zpracujeme postupně celou frontu.
```
; This becomes the loop back point to
read packets from the ring.
; now only loop back and read until
those packets received at the time
; the current page register is read
above have been read.
recv_frame:
;; Read all the frames ??
  MOV     R4,next_packet
  POP     Acc
          ; Get current page of rcv ring
  CJNE    A,AR4,recv_more_frames
```
Došli jsme na konec fronty, potvrdíme přerušení a vrátíme se instrukcí RET
```
  SJMP    recv_frame_break
```
Přečteme hlavičku paketu
```
;; Remove one frame from the ring.
Boundary is always a page behind.
recv_more_frames:
  PUSH    Acc
          ; Save current page of rcv ring
  MOV     curr_recv,R4
; MOV     R5,#0     ; R4R5 Remote Start
Address Register
  ACALL   ReadPacketHeader
```
A otestujeme zda je přijatý paket vpořádku
```
  MOV     A,rcv_hdr
          ; Get the buffer status byte
  XRL     A,#ENRSR_RXOK ; Good packet ?
  JZ      GoodPacket
```
Popř. zda nebyl přijat broadcast/multicast
```
  XRL     A,#(ENRSR_PHY + ENRSR_RXOK) XOR
                    ENRSR_RXOK
          ;Received broadcast/multicast
  JNZ     InvalidNextFramePtr
GoodPacket:
```
Paket vypadá vpořádku, zkontrolujeme ukazatel na hlavičku následujícího paketu
```
; EVEN if the NIC header status is OK, I
have seen garbaged NIC headers, so
; it doesn't hurt to range check the
next packet pointer here.
  MOV     A,rcv_hdr_nxt_pg
          ; get pointer to next packet
  MOV     next_packet,A
                    ; and save it
  CJNE    A,#RX_START_PG,$+3
;; First page of RX Ring
  JC      InvalidNextFramePtr
                    ; < RX_START_PG
  CJNE    A,#NE_STOP_PG,$+3
;; Last page + 1 of RX Ring
```
V případě, že je vše v pořádku paket zpracujeme
```
  JC      ProcessFrame  ; next_packet
```
Pokud není v pořádku, nastavíme ukazatel na konec fronty a ukončíme proceduru.
```
InvalidNextFramePtr:
  POP     Acc
          ; Get current page of rcv ring
recv_frame_break1:
  MOV     next_packet,A
recv_frame_break:
  ISAOUT  EN0_ISR, #ENISR_RDC
          ;Interrupt status reg
          ;remote dma complete
  RET
```
Zpracování paketu
```
;*************************************************************
;** process frame
;**
ProcessFrame:
  LCALL   ProcessEthPacket
```
Paket byl zpracován, vyjmeme jej z fronty
```
  MOV     R4,next_packet
  DEC     R4
  CJNE    R4,#RX_START_PG,$+3
          ;;First page of RX Ring
  JNC     SetBoundaryNxt
          ; >= RX_START_PG
  MOV     R4,#NE_STOP_PG-1
          ;;(Last page + 1)-1 of RX Ring
SetBoundaryNxt:
  ISAOUT  EN0_BOUNDARY,R4
          ;Boundary page of ring buffer
```
A zpracujeme další případný paket
```
  SJMP    recv_frame
```

Packet Driver RTL8019AS

část 4 - Zdrojové texty a literatura

V první řadě bych chtěl poděkovat kolektivu autorů vedených nesporou autoritou v oboru panem Russellem Nelsononem za jeho práce v oblasti paketových driverů. V jeho práci lze najít podnětné nápady, které byly uplatněny i v tomto driveru.
Výsledný tvar driveru ve tvaru zdrojových souborů je možno stáhnout ve formě RAR souboru z domovské stránky celého projektu http://8052.lphard.cz. Všechny uvedené zdrojové texty podléhají licenci GNU. V případě požadavku na šíření pod jinou licencí, např. pro použití v komerčním produktu, kontaktujte autora.

P-Code Interpreter

část 1 - Struktura P-kódu a jeho interpretr

WWW8052 používá v části programu služeb interpretru P-kódu. Interpretr umožňuje dosáhnou dvou věcí:

- Obejit nemožnost vykonávání programu odjinud, než z vnitřní paměti programu, tj. umožnit vykonávat program např. i z vnější sériové I2C paměti.
- Zmenšení kódu

Struktura P-kódu vychází vychází z několika požadavků:

- Adresace příkazů je přímá, bez indexační tabulky, tj. v instrukci je obsažena adresa vstupního bodu výkonné procedury.
- Parametry příkazů je možno zadávat jak v pořadí MSB-LSB tak LSB-MSB
- Parametry je možno zadávat i nepřímo - pomocí odkazu do RAM
- Příkaz obsahuje max. 3 parametry o délce 16ti bitů

Vyjdeme-li z výše uvedených požadavků, docházíme k následující struktuře:

```
AAAAAAAA  adresa procedury MSB
AAAAAAAA       -"-         LSB
__SSSIII  option: (S)prohození LSB/MSB,
(I)nepřímá adresace parametru
11111111  parametr 1 MSB (MSB1)
11111111     -"-     LSB (LSB1)
22222222  parametr 2 MSB (MSB2)
22222222     -"-     LSB (LSB2)
33333333  parametr 3 MSB (MSB3)
33333333     -"-     LSB (LSB3)
```

Celkem by P-kód měl 9 bytů na instrukci, což je poměrně hodně, vzhledem k tomu, že poměrně mnoho příkazů nepoužívá všechny 3 parametry. Často je nevyužita i možnost daná 16ti bitovým parametrem, velká část používaných parametrů je pouze 8mi bitová. Jak z toho ven? Cesta je jednoduchá, strukturu P-kódu doplníme o 6 bitů, udávajících zda je příslušný parametr MSB1 až LSB3 nenulový. Doplněné bity nám umožňují zavést proměnou délku P-kódu vynecháním celých nulových parametrů, či jejich MSB/LSB části. Omezíme-li mírně možnosti P-kódu o možnost prohození LSB/MSB u parametru 2 a adresaci procedury na 13 bitů, docházíme k výsledné struktuře:

```
IIIAAAAA  adresa procedury MSB,
(I)nepřímá adresace parametru
AAAAAAAA       -"-         LSB
abcdefSS  option: (S)prohození LSB/MSB,
parametr MSB1..LSB3 uveden
11111111  parametr 1 MSB (MSB1 - a)
11111111     -"-     LSB (LSB1 - b)
22222222  parametr 2 MSB (MSB2 - c)
22222222     -"-     LSB (LSB2 - d)
33333333  parametr 3 MSB (MSB3 - e)
33333333     -"-     LSB (LSB3 - f)
```

Výsledný P-kód má proměnou délku od 3 do 9 bytů. Jeho generování je možno zabezpečit makrem:

```
;
; pcode EQU
;
blank equ 0
none  EQU 0000h
IND1  EQU 8000h ;par1 INDirect
IND2  EQU 4000h ;par2 INDirect
IND3  EQU 2000h ;par3 INDirect
SWAP1 EQU 8002h ;always Indirect Swap
SWAP3 EQU 2001h
;
DP    MACRO fce, option, par1, par2,
par3
      DW fce + (option and 0E000h)
btk1  SET 0
btk2  SET 0
btk3  SET 0
btk4  SET 0
btk5  SET 0
btk6  SET 0
if (par1 SHR 8)  0    btk1 SET 10000000b
endif
if (par1 AND 0FFh)  0 btk2 SET 01000000b
endif
if (par2 SHR 8)  0    btk3 SET 00100000b
endif
if (par2 AND 0FFh)  0 btk4 SET 00010000b
endif
if (par3 SHR 8)  0    btk5 SET 00001000b
endif
if (par3 AND 0FFh)  0 btk6 SET 00000100b
endif
      DB btk1 or btk2 or btk3 or btk4 or
btk5 or btk6 or (option and 0003h)
if (par1 SHR 8)  0
      DB high(par1)
endif
if (par1 AND 0FFh)  0
      DB low(par1)
endif
if (par2 SHR 8)  0
      DB high(par2)
endif
if (par2 AND 0FFh)  0
      DB low(par2)
endif
if (par3 SHR 8)  0
      DB high(par3)
endif
if (par3 AND 0FFh)  0
      DB low(par3)
endif
ENDM
```

Z úvodní definice parametrů je vidět, že parametry SWAP mají smysl pouze ve spojení s příslušným parametrem IND (pořadí LSB/MSB u literární proměné je možno prohodit už ve fázi překladu zdrojového textu). V úvodní definici adresace procedury P-kódu je uvedeno, že je možno pomocí P-kódu adresovat strojový kód v rozsahu 13ti bitů, ale není to tak zcela pravda. Je zde ještě jedno omezení. Pro ukončení činosti interpretru P-kódu a přechod do režimu strojového kódu je použita zarážka, často používaná i při definici ASCIIZ textů. To znamená, že definici P-kódu ukončíme uvedením binární nuly za konec části v P-kódu, např. pomocí DB 0. Tato binární nula však koliduje s definicí adresy procedury v P-kódu, je-li použitá instrukce bez modifikace INDirect. MSB část adresy proto nesmí být nulová. Vyjdeme-li z těchto dvou omezení, vychází použitelný rozsah adres 100h...1FFFh. Na toto omezení je nutno pamatovat především při linkování a zabezpečit, aby žádná vstupní adresa strojové P-procedury nebyla v rozsahu 0000..00FFh.

Přejděme nyní k předávání parametrů z interpretru P-kódu do vlastních strojových P-procedur. Parametry se předávají pomocí registrů a to přes:

R6(MSB)R7(LSB) v případě 1. parametru
R4(MSB)R5(LSB) v případě 2. parametru
R2(MSB)R3(LSB) v případě 3. parametru

Pokud je v parametrech P-kódu instrukce zadáno méně než 6 byte, jsou nezadané hodnoty parametrů vynulovány. Po případném načtení parametrů do registrů je provedeno vyhodnocení INDirect příznaků. Je-li příslušný příznak uveden, je pomocí LSB části dvoj-registru, tj. registrem R7/R5/R3 naadresována vnitřní RAM procesoru a zní je vždy vyzvednuto 16bitů parametru v pořadí MSB - LSB, jak je vidět z následujího fragmentu kódu:

```
NoOpt2: JNB token.7,noI1
        MOV A,R7
        MOV R0,A
        MOV A,@R0
        MOV R6,A
        INC R0
        MOV A,@R0
        MOV R7,A
noI1:   JNB token.6,noI2
```

Po vyhodnocení INDirect příznaků jsou následně vyhodnoceny příznaky SWAP, prohazující mezi sebou MSB a LSB část dvojregistru. Vyhodnocení objasníme nejlépe fragmentem kódu:

```
noI3:   jnb pcodeOption.1,NoOpt1
        MOV A,R6   ;swap1
        XCH A,R7
        MOV R6,A
NoOpt1: jnb pcodeOption.0,NoOpt0
```

Výše uvedené makro DP je univerzální, automaticky zjišťuje velikost (a přítomnost) parametrů. Nicméně je nutné vzhledem k možnosti rozdělit program do modulů, mít možnost explicitně udat velikost BYTE/WORD parametrů 1 až 3. K tomu složí sada maker DPB__, ..., DP__B, ..., DPW__, kde písmeno B či W na příslušné pozici znamená, že velikost parametru je napevno nastavena na BYTE či WORD, popř. _ znamená, že velikost se zjišťuje automaticky.

P-Code Interpreter

část 2 - P-instrukce pro základní matematické operace a větvení programu

Navázání interpretru P-kódu na jednotlivé P-instrukce lze demonstrovat např. na matematické knihovně. Základní skelet P-instrukce objasníme na následující funkci:

```
;
; *** PCMPBI *** compare STRING (cpu
IDATA mem) with BYTE constant ***
; pcmpbi buffer_addr, byteconst
;       R7,  R5
pcmpbi: SETB zflag
        MOV  A,R7
        MOV  R0,A
        CLR  C
        MOV  A,@R0
        SUBB A,R5
        MOV  cflag,C
        JZ   pcmpbi0
        CLR  zflag
pcmpbi0:RET
```

Vstupní parametry jsou předávány pomocí registrů R7 (parametr 1) a registru R5 (parametr 2), tak jak bylo uvedeno v předchozí kapitole. Tato procedura porovnává dvě čísla, jedno zadané pomocí odkazu přes registr R7 a druhé přímo v registru R5. Výsledky porovnání jsou uloženy do globálních bitových proměných zflag a cflag.

V mnemonice I51 bychom mohli zapsat:

zflag = (@R7 == R5),
cflag = (@R7 < R5).

Podle hodnoty těchto bitových proměných, definovaných v hlavním modulu interpretru P-kódu (Pcode.asm), lze provádět větvení programu.

Ukončení procedury, je pomocí instrukce RET.

Obsah knihovny WWW8051, modul PARITM

instrukce	parametry	popis	operace	příznaky
pcmpbi	(B)a, (B)imm	Compare Byte to Immediate	(BYTE)@a - (BYTE)imm	zflag, cflag
pmovbi	(B)a, (B)imm	Mov Immediate to Byte	(BYTE)@a = (BYTE)imm	
pmovb	(B)a, (B)b	Mov Byte to Byte	(BYTE)@a = (BYTE)@b	
pcmpwi	(B)a, (W)imm	Compare Word to Immediate	(WORD)@a - (WORD)imm	zflag, cflag
pbitwi	(B)a, (W)imm	Test bit Immediate Word	(WORD)@a AND (WORD)imm	zflag
pmovwi	(B)a, (W)imm	Mov Immediate to Word	(WORD)@a = (WORD)imm	
paddwi	(B)a, (W)imm	Add Immediate to Word	(WORD)@a += (WORD)imm	
psubwi	(B)a, (W)imm	Subbtract Immediate Word	(WORD)@a -= (WORD)imm	
pxorwi	(B)a, (W)imm	XOR Immediate Word	(WORD)@a=(WORD)@aXOR (WORD)imm	
pandwi	(B)a, (W)imm	AND Immediate Word	(WORD)@a = (WORD)@a AND (WORD)imm	
pshiftw	(B)a, (B)n	Shift Word n-bit Left/Right	if(n>0){(WORD)@a << n} else{(WORD)@a >>(-n)}	
pcmpn	(B)a, (B)b, (B)lng	Compare two string	for(i=0;i<lng;i++) (B)@a[i]-(B)@b[i]	zflag, cflag
paddn	(B)a, (B)b, (B)lng	Add two string	for(i=0;i<lng;i++) (B)@a[i]+=(B)@b[i]	

(B) popř. **(BYTE)** - délka argumentu, či operace 8 bitů
(W) popř. **(WORD)** - délka argumentu, či operace 16 bitů
a, **b** - adresa dat v paměti procesoru (DATA/IDATA)
@a - data v paměti procesoru na adrese a
@a[i] - data v paměti procesoru na adrese a + i
imm - absolutní konstanta
n - počet bitů
lng - délka stringu

Nahlédnutí do tabulky základních aritmetických operací odhalí poměrně malé možnosti adresace jednotlivých operandů, většinou s absolutní konstantou. Není to však pravda. Nesmíme zapomenout na INDirect příznaky samotného P-kódu.

Např. příkaz

```
DPBB_ pmovwi,
        IND2,data_addr,wordtemp,blank
```

provede (WORD)@a = (WORD)@b

tj. přiřazení

(WORD)data_addr = (WORD)wordtemp

Uplatníme-li příznak INDirect na BYTE příkazy nesmíme zapomenout, že INDirect se rozvíjí jako WORD a BYTE příkaz s tohoto rozvoje použije jeho LSB část. Je proto nutno uvažovat, že se ve skutečnosti bere operand z adresy @(operand+1), popř. načítání INDirect parametrů ještě modifikovat pomocí SWAP.

Uplatníme-li důsledky INDirect na instrukční kód, zjistíme, že např. MOVB je duplicitní instrukce.

Obsah knihovny WWW8051, modul PCODE

instrukce	parametry	popis	operace
pcall	(W)addr	Call to P-Code subroutine	push current P-Code PC to stack, jump to addr
pret		Return from P-Code subroutine	pop P-Code PC from stack
pjump	(W)addr	Jump to P-Code addr	jmp addr
pjumpeq	(W)addr	Jump if zflag to P-Code addr	if(zflag) jmp addr
pjumpne	(W)addr	Jump if not zflag to P-Code addr	if(zflag == 0) jmp addr
pjumpCarry	(W)addr	Jump if cflag to P-Code addr	if(cflag) jmp addr
pjumpnCarry	(W)addr	Jump if not cflag to P-Code addr	if(cflag == 0) jmp addr

addr - adresa

Adresa, uvedená jako parametr P-Code instrukce, musí ukazovat opět do P-kódu, nelze skákat na instrukce strojového kódu. Pokud je potřeba skoku přímo do strojového kódu, je nutno nejprve ukončit sekci P-kódu, pomocí DB 0 a následně použít příslušnou strojovou instrukci jmp, jb, jnb, ... Pokud je potřeba volat strojovou proceduru z P-kódu pomocí CALL, není ukončení potřeba. Volání je možno provést tak, jako by to byla P-kódová instrukce, např. pomocí makra DP strojova_instrukce, parametr1, parametr2, parametr3.

P-Code Interpreter

část 3 - P-instrukce pro základní síťové operace

Obsah knihovny WWW8051, modul PETHER ver. 1.0

instrukce	parametry	popis	operace	příznaky
ptstr2s	(B)a, (W)offset, (W)lng	Compare String to Receive Frame	for(i=0;i<lng;i++) (B)@a[i]-(B)@receive[offset+i])	zflag
ptstr2e	(B)e, (W)offset, (W)lng	Compare eepromString Frame to Receive	for(i=0;i<lng;i++) (B)@e[i]-(B)@receive[offset+i])	zflag
pr2s	(B)a, (W)offset, (W)lng	Copy from Receive Frame to String	for(i=0;i<lng;i++)(B) @a[i]=(B)@receive[offset+i])	
px2s	(B)a, (W)offset, (W)lng	Copy from Xmit Frame to String	for(i=0;i<lng;i++) (B)@a[i]=(B)@receive[offset+i])	
pi2x	(W)offset, (W)imm	Mov Word Immediate to Xmit Frame	(W)@xmit[offset]=(W)imm	
ps2x	(W)offset, (B)a, (W)lng	Mov String to Xmit Frame	for(i=0;i<lng;i++) (B)@xmit[offset+i]=(B)@a[i])	
pe2x	(W)offset, (B)e, (W)lng	Mov eepromString to Xmit Frame	for(i=0;i<lng;i++) (B)@xmit[offset+i]=(B)@e[i])	
pc2x	(W)offset, (B)c, (W)lng	Mov codeString to Xmit Frame	for(i=0;i<lng;i++) (B)@xmit[offset+i]=(B)@c[i])	
pz2x	(W)offset, (W)lng	Mov Zero to Xmit Frame	for(i=0;i<lng;i++) (B)@xmit[offset+i]=0)	
pr2x	(W)xmit_offset, (W)receive_offset, (W)lng	Mov Receive Frame to Xmit Frame	for(i=0;i<lng;i++) (B)@xmit[xmit_offset+i]=@receive[receive_offset+i])	

(B) popř. **(BYTE)** - délka argumentu, či operace 8 bitů
(W) popř. **(WORD)** - délka argumentu, či operace 16 bitů
a, **b** - adresa dat v paměti procesoru (DATA/IDATA)
c - adresa dat v paměti PROGRAMU procesoru (CODE)
e - adresa dat v paměti EEPROM procesoru (eXDATA)
@a - data v paměti procesoru na adrese a
@a[i] - data v paměti procesoru na adrese a + i
@a[offset + i] - data v paměti procesoru na adrese a + offset + i
@c[i] - data v paměti PROGRAMU procesoru na adrese c + i
@e[offset + i] - data v paměti EEPROM procesoru na adrese e + offset + i
imm - absolutní konstanta
lng - délka stringu

Většina P-instrukcí používá DMA kanál Ethernet čipu pro přímé přenosy mezi požadovanou datovou oblastí a čipem. Existují však vyjímky. První z ní je instrukce **pi2x**, která používá proměnou **workreg** k uložení konstanty, zadané v parametru instrukce. Druhou vyjímkou je instrukce **pr2x** která, vzhledem k neexistenci oddělených DMA kanálů pro přijímací a vysílací frame, musí používat přenos přes pomocnou datovou oblast. Proto tato instrukce kopíruje data po částech, pomocí vyrovnávací paměti **ebuf**.

instrukce	parametry	popis	operace	příznaky
xmit_frame	(W)lng	Transmit Xmit Frame	Transmit lng bytes to Ethernet	
psendchar	(B)char	Write char to Xmit Frame	@xmit[tcpWritePointer++]=char	
bflush		Flush ebuf to Xmit Frame		

Stejně jako u instrukce **pr2x**, nejsou u instrukce **psendchar** data přímo zapisovány do Ethernet čipu, ale pomocí vyrovnávací paměti **ebuf**. Tato paměť je ovládána přes dvě globálních proměné. První z nich **tcpWritePointer**, určující offset, odkud budou zapisována data do xmit frame. Tento offset se automaticky průběžně zvyšuje, tak jak dochází ke kopírování vyrovnávací paměti **ebuf** do vysílaného frame. Druhou proměnou je **unwrited**, udávající offset ve vyrovnávací paměti **ebuf**. Skutečný offset zapisovaných dat je ve skutečnosti **tcpWritePointer + unwrited**. Před prvním voláním **psendchar** je nutné obě proměné inicializovat, **tcpWritePointer** na hodnotu potřebného offsetu a **unwrited** na **0**. Zápis dat přes **psendchar** je nutno ukončit voláním **bflush**.

instrukce	parametry	popis	operace	příznaky
in_checksum		Calculate Frame checksum		

Instrukce **in_checksum** počítá sumu bloku, uloženého v paměti Ethernet čipu. Blok je vymezen absolutní adresou v paměti čipu, předávanou přes globální proměnou **chkaddr** a svoji délkou, předávanou přes globální proměnou **chklen**. Výsledná a zároveň inicializační hodnota sumy je předávána přes globální proměnou **chkaddr**.
Modul pether je součástí knihovny **kernel.lib**, obsažené ve vývojovém systému

Obsluha sériového kanálu

Obsluha sériového kanálu u WWW8052 je podřízena třem požadavkům:

Protokolu TCP
Minimálnímu obsazení paměti dat
Spolupráci s MS Windows

Rozeberme nyní jednotlivé požadavky podrobněji:

Podpora protokolu TCP vyžaduje na straně WWW8052 alespoň minimální podporu TCP stacku, tj. mimo jiné i možnost zopakovat vysílání vyslaných, ale nepotvrzených dat. Vzhledem k zoufalému nedostatku paměti RAM je nutné zvolit, namísto přesunu dat z vyrovnávací paměti sériového kanálu do TCP stacku, složitější postup. Data jsou z vyrovnávací paměti pouze zkopírována do TCP paketu. Jejich vymazání z vyrovnávací paměti je provedeno až po potvrzení přenesení dat příjemcem.

Obvykle se vyrovnávací paměť sériového kanálu realizuje pomocí kruhových bufferů s dvojicí ukazatelů. Jeden z ukazatelů slouží jako ukazatel zapisovaných dat, druhý složí k jako ukazatel pro čtení. Nicméně, každý uspořený byte je dobrý, takže WWW8052 používá lineární buffery. Jejich ovládání je následující:
Buffer vysílače používá jediný ukazatel (**s1point**) složící k zápisu dat do bufferu, ukazující pozici prvního volného místa ve vysílacím bufferu. Pro přesun dat z bufferu do vysílače je použit ukazatel napevno nastavený na počátek bufferu. Po přesunu znaku z bufferu do sériového kanálu je celý buffer posunut o jeden znak směrem k jeho počátku a tím je vyslaný znak z bufferu odstraněn.

Buffer přijímače používá, vzhledem k podpoře TCP, dvojici ukazatelů. První (**r1point**) ukazuje na první volnou pozici v přijímacím bufferu a slouží jako ukazatel zápisu pro přenos dat z přijímače do bufferu. Druhý (**r2point**) ukazuje na první nepřečtený znak v bufferu. Slouží pro přenos dat z bufferu přijímače do aplikace. Při opakování přenosu dat je prostě možno tento ukazatel posunout zpět. Pokud je potvrzen přenos n-znaků tak je celý buffer posunut o n-znaků ke svému počátku a o stejnou hodnotu jsou posunuty k počátku i oba ukazatele.

Spolupráci s MS Windows je poměrně jednoduchá, nicméně náročná na paměť. Dimenzování vysílacího bufferu Windows neovlivňují, takže jeho velikost závisí jen na požadavku aplikace běžící na WWW8052. Silně však ovlivňují dimenzování bufferu přijímače, a to především svojí obsluhou HW vyrovnávacích pamětí na čipu sériového vysílače v PC. Reakce na žádost o pozastavení vysílání, ať již přes HW handshake pomocí RTS/CTS či přes SW handshake pomocí XON/XOFF, je závislá na zaplnění bufferů v sériovém kanálu PC. Je proto nutno počítat s přeběhem. To znamená, že vyrovnávací pamět v WWW8052 musí, i přes signalizaci stůj, jsem plná, být schopna přijmout a zpracovat další blok dat. Velikost bloku dat závisí na nastavení Windows a je nutno počítat s typickým přeběhem cca 16 byte. O tuto hodnotu je nutno zvětšit přijímací buffer. Z paměti WWW8052 tak typicky vyrovnávací paměť vysílače ukrojí 16+1 byte a vyrovnávací paměť přijímače 32+2 byte.

Obsluha sériového kanálu na fyzické úrovni je v WWW8052 řešena přes přerušení. Přerušovací procedura **IRIT**, mimo přenosy mezi buffery a vlastním přijímačem/vysílačem, nastavuje trojici stavových bitů (**rxint**, **txint**, **srun**). Stavový bit **rxint** signalizuje, že od jeho posledního vymazání, došlo k příjmu nejméně jednoho znaku ze sériového kanálu. Stavový bit **txint** obdobně signalizuje odeslání nejméně jednoho znaku. Stavový bit **srun** emuluje hardwarovou signalizaci "vysílač vysílá předchozí zapsaný znak". K řízení

toku dat slouží dva bitové přepínače, nastavované při konfiguraci sériového kanálu. První z nich (**rtscts**) povoluje HW řízení toku dat pomocí dvojice řídicích signálů RTS/CTS. Druhý z nich (**xonxoff**) povoluje SW řízení toku dat pomocí protokolu XON/XOFF.

Je-li použit protokol XON/XOFF je pro řízení stavového automatu řídícího příjem a vysílání použita další čtveřice stavových bitů. Stavový bit **flagtxoff** odpovídá signálu CTS, použitému při HW řízení toku dat. Je-li tento bit nastaven, je zakázáno vysílání dat. Bit **flagtxoff** je je nulován popř. nastaven přijetím znaku XON popř. XOFF, nezapisovaných při aktivovaném protokolu XON/XOFF do bufferu přijímače. Přijímač používá obdobný stavový bit **flagxoff**, určující zda byl naposledy vyslán řídicí znak XON či XOFF a tím povoleno, či zakázáno vysílání protistrany. Vzhledem k nemožnosti okamžitého vyslání znaku XON/XOFF a nutnosti čekání na vyprázdnění vysílače je nutno použít pro řízení příjmu dvojici pomocných bitů **xonrq** a **xoffrq** signalizujících obsluze vysílače, že má místo dat z bufferu vyslat příslušný řídicí znak.

Zpracování Ethernet Paketu

Hlavní smyčka

Hlavní smyčka ukázkové aplikace WWWW8052 zabezpečuje obsluhu několika procesů:
Každý průchod smyčkou resetuje Watchdog obvodu 89S8252
Při průchodu smyčkou jsou otestovány podmínky řízení toku dat (Xon/Xoff, CTS) a zahájeno případné vysílání.
Je zkontrolován obvod RTL8019 a přečtena případná hlavička přijatého paketu.

```
;
;#####################################################
;  Main Loop                                        
;#####################################################
;
mainloop:
       mov     WMCON,#RESTARTWATCHDOG
;restart Watchdog
        lcall   sstat
;scan RTS/CTS,...
        mov     stateFlg,#0
;clear rx_eth_bit, flagARP..flagTCP
        LCALL   rcv_pkt
       jnb     rx_eth_bit,NoRxEth
```

V případně přijetí paketu je zkontrolován obsah paketu na přítomnost zpracovatelných typů a známý paket (ICMP / TCP / UDP / ARP) je zpracován.

```
       acall   ProcessEthPacket
        jnb     flagICMP,$+6
        lcall   ProcessICMP
        jnb     flagTCP,$+6
        lcall   ProcessTCP
        jnb     flagUDP,$+6
        lcall   ProcessUDP
        jnb     flagARP,$+6
        lcall   ProcessARP
```

Není-li přijat žádný paket, popř. je přijat nezpracovávatelný paket, je otestováno uplynutí 1 milisekundy.

```
NoRxEth:jnb     bit1ms,mainloop
        clr     bit1ms
;~1 ms loop
fastloop:
```

Vyšli případné požadavky na zjištění MAC adresy Gateway či "telnet" protistanice

```
        jnb     flagArpGwRq,$+6
        lcall   ProcessArpGwRq
        jnb     flagArpPoiRq,$+6
        lcall   ProcessArpPoiRq
```

Je-li zjištěna MAC adresa "telnet" protistanice a povoleno případné spojení s protistanicí, zahaj TCP vyjednávání vysláním SYN paketu

```
        jnb     flagSynRq,$+6
        lcall   ProcessSynRq
```

Není-li otevřeno spojení s "telnet" protistanicí a je-li povoleno zahaj vyjednávání o spojení

```
        mov     a,tcpState23
        jnz     est23ch         ;if Connect
Established, don't make new
        jnb     rxint, no23ch   ;if Not
Data to send, don't make connection
        jnb     flagIPactive,no23ch
         ;if flash_ip_point = 0.0.0.0,
         ;don't make connection
        lcall   makeconnect23
        sjmp    no23ch
```

V případě otevřeného spojení s "telnet" protistanicí zkontroluj zda není větší množství nevyslaných dat v bufferu a popřípadě je odešli.

```
est23ch:
        jb      flag23WaitForAck,no23ch
         ;if All previous send data is Acked
         ;and sizeof(unsend data in Rx
         buffer) >= MAXCHAR/2 char
         ;then send output packet
        mov     a,r1point
       cjne    a,#rser+(MAXCHAR/2),$+3
        jc      no23ch
         ;jump if sizeof(unsend data in Rx
         buffer)
```

Zkontroluj uplynutí 200 ms

```
        djnz    slowtimer,mainloop
        mov     slowtimer,#slowtiming
;~50 ms loop
slowloop:
        clr     tcpState80stateSyn
        mov     a,Retry23
        jz      no23timer
        dec     Timeout23
        mov     a,Timeout23
        jnz     no23timer
;~200 ms loop
```

Zopakuj vysílání případnéch nepotvrzených dat, popř. po vypršení počtu opakování spojení uzavři.

```
        mov     Timeout23,#ethtiming
        dec     Retry23
        mov     a,Retry23
        jz      kill23
        lcall   RetryTelnet
        ajmp    no23
kill23:
        lcall   KillTelnet
        ajmp    no23
```

Jsou-li v bufferu jakákoliv nevyslaná data tak je odešli.

```
no23timer:
       jnb     tcpState23stateEstablished,
                        no23
       jbc     flag23WaitForAck,no23
        lcall   TelnetOut
;send all queued
;data, don't wait for limit MAXCHAR/2
no23:
        ajmp    mainloop
```

Vyhodnocení přijatého Ethernet paketu

Po přijetí paketu je v něm hledáno několik "magických" čísel, specifikujících o jaký typ paketu se jedná. Nejprve je v hlavičce paketu hledána identifikace dle RFC894 (A Standard for the Transmission of IP Datagrams over Ethernet Networks) specifikující, že se jedná o přenos IP paketu. Druhou identifikací, hledanou v hlavičce paketu je identifikace ARP paketu dle RFC826 (An Ethernet Address Resolution Protocol) specifikující, že se jedná o přenos ARP paketu.

```
;
;*****************************************************
ProcessEthPacket:
;;if (rx_eth_pkt.pktType == 0x0800) goto
IPPacket; //IP;;else if (rx_eth_pkt.pktType
== 0x0806) goto ArpPacket; //ARP
;;else goto UnknownEthType;
       mov     a,eth_pkt_hdr_type
        xrl     a,#08h
                 ;high ARP and IP ID
        jnz     UnknownEthType
       mov     a,eth_pkt_hdr_type+1
        jz      IPPacket
        xrl     a,#06h          ;low ARP ID
        jz      ArpPacket
UnknownEthType:
        ret                     ;Unknown ID
of Ethernet packet
;*****************************************************
;Type of ethernet packet = 0806h ARP
ArpPacket:
        setb    flagARP
        ret
```

Z přijatého IP paketu dle RFC791 (Internet Protocol) je do pomocného pole v RAM procesoru přenesen blok 12 byte.

Version	IHL	Type of Service	Total Length
Identification		Flags	Fragment Offset
Time to Live	Protocol		Header Checksum
Source Address			
Destination Address			

Destination Address je zkontrolována, zda obsahuje IP adresu WWW8052. Paket určený pro jinou adresu je ignorován.

```
;*****************************************************
;Type of ethernet packet = 0800h IP
NoMyIP:
       mov     WMCON,#WATCHDOG
;disable EEPROM DPTR'
        ret
         ;IP with IP_DSTADDR != flash_my_ip
IPPacket:
;;if (rx_eth_pkt.pkt.ip.ipheader.DstAddr ==
                        flash_my_ip) {
       mov     R7,#dwordtemp
;buffer_addr = dwordtemp iftemp
       mov     R5,#IP_TTL
;source_packet_offset
        mov     R3,#12
;length
       lcall   short_pr2s
;read IP_TTL IP_PROTO IP_CKSUM
;IP_SRCADDR IP_DSTADDR to temp var
       mov     dptr,#flash_my_ip
       mov     r0, #dwordtemp+
                 (IP_DSTADDR-IP_TTL)
       mov     r1, #IP_ADDR_LEN
       mov     WMCON,#WATCHDOG OR EEMEN
                 ;enable EEPROM
myiploop:movx   a,@dptr
        xrl     a,@r0
        jnz     NoMyIP
        inc     dptr
        inc     r0
       djnz    r1,myiploop
```

Následuje otestování Source Address zda se odesílatel paketu nachází na stejné síti jako WWW8052, či je nutno s ním komunikovat přes bránu (gateway).
Vyjdeme-li ze vztahů:
MY_NETWORK = (my_ip AND network_mask)
SRC_NETWORK = (src_ip AND network_mask)
a test
MY_NETWORK == SRC_NETWORK
nahradíme jeho možnou strojovou reprezentací
(MY_NETWORK XOR SRC_NETWORK) == 0
dojdeme několika úpravami ke vztahu
((my_ip XOR src_ip) AND network_mask) == 0
použitém v WWW8052

```
;;if (((rx_eth_pkt.pkt.ip.ipheader.SrcAddr
flash_my_ip) flash_ip_mask) != 0) {
       mov     dptr,#flash_my_ip
       mov     r0, #dwordtemp+
                 (IP_SRCADDR-IP_TTL)
       mov     r1, #IP_ADDR_LEN
gwloop1:movx    a,@dptr
        xrl     a,@r0
        mov     @r0,a
        inc     dptr
        inc     r0
       djnz    r1,gwloop1
       mov     dptr,#flash_ip_mask
       mov     r0, #dwordtemp+
                 (IP_SRCADDR-IP_TTL)
       mov     r1, #IP_ADDR_LEN
gwloop2:movx    a,@dptr
        anl     a,@r0
        jnz     gwloopbreak
        inc     dptr
        inc     r0
       djnz    r1,gwloop2
gwloopbreak:
       mov     WMCON,#WATCHDOG
;disable EEPROM DPTR'
        jz      noneedgw
```

I když lze v 99% předpokládat, že požadavek z cizí sítě přijde přes bránu a tím pádem má nastavenu správnou MAC adresu odesílatele, nelze se na to spolehnout. V přijatém paketu z cizí sítě proto vždy vyměníme MAC adresu odesílatele za MAC adresu brány. Tato MAC adresa je v případné odpovědi použita jako adresa příjemce. Výměnu MAC adresy však můžeme provést pouze známe-li ji. Otestujeme proto zda byla MAC adresa brány pomocí ARP dotazu již zjištěna. Je-li tento test negativní,

nastavíme požadavek na vyslání ARP dotazu a přijatý paket zahodíme jako nezpracovatelný.

```
;;  if(mac_gateway == 0) {
       mov     r0, #mac_gateway
       mov     r1, #ETHER_ADDR_LEN
gwloop3:mov     a,@r0
       jnz     noneedarpgw
        inc     r0
       djnz    r1,gwloop3
;;   MakeArpGwRq(); return;
       setb    flagArpGwRq
;Arp Gateway Request
        ret
;;  } else {
;;   EthRcvHdr.pktSrc = mac_gateway;
noneedarpgw:
       mov     r0, #mac_gateway
       mov     r1, #eth_pkt_hdr_src
       mov     r2, #ETHER_ADDR_LEN
gwloop4:mov     a,@r0
        mov     @r1,a
        inc     r0
        inc     r1
       djnz    r2,gwloop4
;;  }
;;}
```

V paketu s ověřenými IP adresami jsou hledány čísla protokolů dle RFC1700 (Assigned Numbers) a při nalezení čísla protokolu, podporovaného WWWW8052 je nastaven příznak odpovídajícího protokolu sdělující např. hlavní smyčce programu, že byl přijat zpracovatelný paket.

```
noneedgw:
;;  switch (rx_eth_pkt.pkt.ip.Proto) {
;;    case IPT_ICMP: goto protoICMP;
       mov     R0,#dwordtemp+
                 (IP_PROTO-IP_TTL)
        mov     a,@R0
       xrl     a,#IPT_ICMP
;proto ICMP
        jz      protoICMP
;;    case IPT_TCP:  goto protoTCP;
       xrl     a,#IPT_ICMP xor IPT_TCP
;proto TCP
        jz      protoTCP
;;    case IPT_UDP:  goto protoUDP;
       xrl     a,#IPT_TCP xor IPT_UDP
;proto UDP
        jz      protoUDP
        ret
protoICMP:
       setb    flagICMP
        ret
protoTCP:
       setb    flagTCP
        ret
protoUDP:
       setb    flagUDP
        ret
;;  }
;;}
;
```

WWW8052 - http server

Základní charakteristiky jednoduchého www serveru, obsaženého ve vzorovém příkladu:

Podpora základního souborového systému, adresáře jsou ignorovány, ze jména souboru včetně přípony je spočítána 16-ti bitová suma a ta je použita k vyhledávání v souborovém systému.
Souborový systém může být rozložen na 3 "média" externí I²C EEPROM, flashROM procesoru, EEPROM procesoru (prohledávání souborového systému probíhá v uvedeném pořadí, proto lze přidáním externí I²C EEPROM překrýt soubory v procesoru).
Podporována je metoda GET.
Jsou podporovány jednoduché CGI skripty.

CGI skripty je možno do html kódu stránky vložit jednou z následujících možností:

'mac.cgi'
CGI skript je ohraničen z obou stran znakem **'** a musí obsahovat příponu **.cgi**
Použití: jméno CGI skriptu včetně znaků **'** je nahraženo jeho výstupem.

'mac.cgi?🛠
CGI skript je ohraničen ze začátku znakem **'** a na konci znakem **?**, následovaným jedním znakem parametru (zde zobrazeným jako znak 🛠, může být i speciální či netištitelný znak) a musí obsahovat příponu **.cgi**
Použití: jméno CGI skriptu včetně znaku **'** a **?** a parametru je nahraženo jeho výstupem.
Načtený parametr skriptu se předává přes globální proměnnou **state80**

?mac.cgi'text1{text2}
CGI skript je ohraničen ze začátku znakem **?** a na konci znakem **'** a musí obsahovat příponu **.cgi** Skript je následován dvojicí textů **text1** a **text2**. Text1 či text2 může být i prázdný, nicméně musí být uvedeny alespoň ohraničující složené závorky.
Použití: jméno CGI skriptu včetně znaku **?** a **'** je nahraženo jeho případným výstupem, nicméně hlavním účelem CGI skriptu je nastavení příznaku **zflag**.
V závislosti na nastanení příznaku **zflag** je vypsán text1 či text2 (bez ohraničujících znaků **{}**)

?mac.cgi?🛠text1{text2}
CGI skript je ohraničen z obou stran znakem **?** a musí obsahovat příponu **.cgi**
Za ohraničujícím znakem je jeden znak parametru (zde zobrazeným jako znak 🛠, může být i speciální či netištitelný znak).
Skript je následován dvojicí textů **text1** a **text2**. Text1 či text2 může být i prázdný, nicméně musí být uvedeny alespoň ohraničující složené závorky.
Použití: jméno CGI skriptu včetně znaků **?** je nahraženo jeho případným výstupem, nicméně hlavním účelem CGI skriptu je nastavení příznaku **zflag**.
Načtený parametr skriptu se předává přes globální proměnnou **state80**
V závislosti na nastanení příznaku **zflag** je vypsán text1 či text2 (bez ohraničujících znaků **{}**)

Zdrojové texty http serveru jsou v souboru **http.a51**. Příklady html stránek a použití CGI skriptů jsou uvedeny v následující kapitole. Uvedené příklady obsaženy v souboru **htmlpage.a51**. Oba uvedené zdrojové soubory jsou součástí GNU vývojového systému.

WWW8052 - Setup

Konfiguraci serveru WWW8052 je možno zobrazit a měnit pomocí CGI skriptů. V popisované konfiguraci je v serveru WWW8052 uloženo několik html stránek a příslušných skriptů, část z nich uložena v paměti EEPROM, část v paměti flashROM.

index.html

Zobrazení konfigurace umožňuje default stránka serveru, alias **index.html**.

Zdrojová html stránka, uložená v serveru WWW8052
(Pozn. Netištitelné znaky 🛠 reprezentují binární parametr příslušné CGI procedury)

```
<html><meta http-equiv="Cache-Control"
content="no-cache">
<title>89C8252 WWW server</title>
<body><center><H1>AT89C8252 WWW server</H1>
Welcome in server WWW8052<br>
running on the AT89S8252 single-chip
Microcontroller and RTL8019AS Ethernet
Controller.<br>
MAC: 'mac.cgi' IP: 'ip.cgi?🛠 MASK:
'ip.cgi?🛠 GW: 'ip.cgi?🛠<br>
COM: 'bd.cgi?🛠00,N,8,1 Flow Control:
XON/XOFF <B>'?xon.cgi?🛠off{on}</B> RTS/
CTS <B>'?rts.cgi?🛠off{on}</B> POINT:
'ip.cgi?🛠
</center></body></html>
```

Expandovaná html stránka, odeslaná serverem WWW8052

```
<html><meta http-equiv="Cache-Control"
content="no-cache">
<title>89C8252 WWW server</title>
<body><center><H1>AT89C8252 WWW server</
H1>
Welcome in server WWW8052<br>
running on the AT89S8252 single-chip
Microcontroller and RTL8019AS Ethernet
Controller.<br>
MAC: 00-00-E8-EE-09-F5 IP: 192.168.0.77
MASK: 255.255.255.0 GW:
192.168.0.253<br>
COM: 38400,N,8,1 Flow Control: XON/XOFF
<B>off</B> RTS/CTS <B>on</B> POINT:
0.0.0.0
</center></body></html>
```

A takto stránku vidí uživatel

config.html

Změnu konfigurace umožňuje stránka serveru **config.html**.

Zdrojová html stránka, uložená v serveru WWW8052
(Pozn. Netištitelné znaky 🛠 reprezentují binární parametr příslušné CGI procedury)

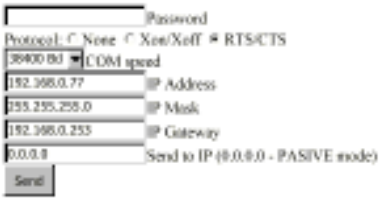
```
<html><meta http-equiv="Cache-Control"
content="no-cache"><head>
<title>89C8252 WWW config</title></head>
<body><form action="/setup.html"
method="GET"><input type="password"
name="W">Password<br>
Protocol:<input type="radio" name="P"
value=0'fl.cgi?0>None
 <input type="radio" name="P"
value=2'fl.cgi?2>Xon/Xoff
 <input type="radio" name="P"
value=1'fl.cgi?1>RTS/CTS<br>
<select name="B" size="1">
 <option value=7'bdc.cgi?7>600 Bd</
option>
 <option value=6'bdc.cgi?6>1200 Bd</
option>
 <option value=5'bdc.cgi?5>2400 Bd</
option>
 <option value=4'bdc.cgi?4>4800 Bd</
option>
 <option value=3'bdc.cgi?3>9600 Bd</
option>
 <option value=2'bdc.cgi?2>19200 Bd</
option>
 <option value=1'bdc.cgi?1>38400 Bd</
option>
</select>COM speed<br>
<input type="text" name="I"
value="'ip.cgi?🛠">IP Address<br>
<input type="text" name="M"
value="'ip.cgi?🛠">IP Mask<br>
<input type="text" name="G"
value="'ip.cgi?🛠">IP Gateway<br>
<input type="text" name="T"
value="'ip.cgi?🛠">Send to IP (0.0.0.0 -
PASIVE mode)<br>
<input type="submit" value="Send"></
form></body></html>
```

Expandovaná html stránka, odeslaná serverem WWW8052

```
<html><meta http-equiv="Cache-Control"
content="no-cache"><head>
<title>89C8252 WWW config</title></head>
```

```
<body><form action="/setup.html"
method="GET"><input type="password"
name="W">Password<br>
Protocol:<input type="radio" name="P"
value=0>None
 <input type="radio" name="P"
value=2>Xon/Xoff
 <input type="radio" name="P" value=1
CHECKED>RTS/CTS<br>
<select name="B" size="1">
 <option value=7>600 Bd</option>
 <option value=6>1200 Bd</option>
 <option value=5>2400 Bd</option>
 <option value=4>4800 Bd</option>
 <option value=3>9600 Bd</option>
 <option value=2>19200 Bd</option>
 <option value=1 SELECTED>38400 Bd</
option>
</select>COM speed<br>
<input type="text" name="I"
value="192.168.0.77">IP Address<br>
<input type="text" name="M"
value="255.255.255.0">IP Mask<br>
<input type="text" name="G"
value="192.168.0.253">IP Gateway<br>
<input type="text" name="T"
value="0.0.0.0">Send to IP (0.0.0.0 -
PASIVE mode)<br>
<input type="submit" value="Send">></
form></body></html>
```

A takto stránku vidí uživatel



setup.html

Výběr konfigurace mezi vygenerovaným a default profilem umožňuje stránka **setup.html** serveru.

Zdrojová html stránka, uložená v serveru WWW8052
(Pozn. Netištitelné znaky 🛠 reprezentují binární parametr příslušné CGI procedury)

```
<html><meta http-equiv="Cache-Control"
content="no-cache">
<title>89C8252 WWW setup</title><body>
<center><H1>AT89C8252 WWW server</
H1><br>
'ipsetup.cgi'MAC: 'mac.cgi'<br>
<b>Current config:</b><br>
IP: 'ip.cgi?🛠 MASK: 'ip.cgi?🛠 GW:
'ip.cgi?🛠 POINT: 'ip.cgi?🛠<br>
COM: 'bd.cgi?🛠00,N,8,1 Flow Control:
XON/XOFF <B>'?xon.cgi?🛠off{on}</B> RTS/
CTS <B>'?rts.cgi?🛠off{on}</B><br>
<b>New config:</b><br>
IP: 'ip.cgi?🛠 MASK: 'ip.cgi?🛠 GW:
'ip.cgi?🛠 POINT: 'ip.cgi?🛠<br>
COM: 'bd.cgi?🛠00,N,8,1 Flow Control:
XON/XOFF <B>'?🛠xon.cgi?off{on}</B> RTS/
CTS <B>'?rts.cgi?🛠off{on}</B><br>
<b>Default config:</b><br>
IP: 'ip.cgi?🛠 MASK: 'ip.cgi?🛠 GW:
'ip.cgi?🛠 POINT: 'ip.cgi?🛠<br>
COM: 'bd.cgi?🛠00,N,8,1 Flow Control:
XON/XOFF <B>'?xon.cgi?🛠off{on}</B> RTS/
CTS <B>'?rts.cgi?🛠off{on}</B>
<form action="setup.html"
method="GET"><b>Entry Password for
change:</b> <input type="password"
name="W"><br>
<input type="submit" name="E"
value="New->Current"><input
type="submit" name="E" value="Default-
>Current"><br>
<a href="/"><b>Exit</b></a></form></
center></body></html>
```

Expandovaná html stránka, odeslaná serverem WWW8052

```
<html><meta http-equiv="Cache-Control"
content="no-cache">
<title>89C8252 WWW setup</
title><body><center><H1>AT89C8252 WWW
server</H1><br>
MAC: 00-00-E8-EE-09-F5<br>
<b>Current config:</b><br>
IP: 192.168.0.77 MASK: 255.255.255.0 GW:
192.168.0.253 POINT: 0.0.0.0<br>
COM: 38400,N,8,1 Flow Control: XON/XOFF
<B>off</B> RTS/CTS <B>on</B><br>
<b>New config:</b><br>
IP: 192.168.0.77 MASK: 255.255.255.0 GW:
192.168.0.253 POINT: 0.0.0.0<br>
COM: 38400,N,8,1 Flow Control: XON/XOFF
<B>off</B> RTS/CTS <B>on</B><br>
<b>Default config:</b><br>
IP: 192.168.0.77 MASK: 255.255.255.0 GW:
192.168.0.253 POINT: 0.0.0.0<br>
COM: 38400,N,8,1 Flow Control: XON/XOFF
<B>off</B> RTS/CTS <B>on</B>
<form action="setup.html"
method="GET"><b>Entry Password for
change:</b> <input type="password"
name="W"><br>
<input type="submit" name="E"
value="New->Current"><input
type="submit" name="E" value="Default-
>Current"><br>
<a href="/"><b>Exit</b></a></form></
center></body></html>
```

A takto stránku vidí uživatel

A heslo potřebné pro ovládání ? Platné je uloženo v modulu **htmlpage.a51** v definici **password:**. (Pro netrpělivé - ve vzorovém konfiguraci je nastaveno na **1234**)

debug.html

Ve vzorové konfiguraci je uložena ještě jedna html stránka **debug.html**, pro fandy hexdumpů, jejíž výstupem je:

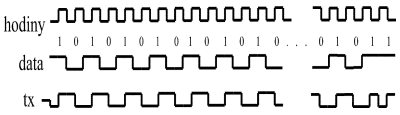
```
00:00 0F 10 00 00 00 00 00 00 46 01 38 07 04 00 00
10:00 00 00 00 00 14 16 73 AE 00 08 40 1A E1 E0 D0
20:11 15 00 00 0C 20 D0 02 04 14 13 63 6D 01 4F 82
30:01 00 00 E8 EE 09 F5 00 A0 C9 92 6A 45 08 00 22
40:03 7A 01 00 00 00 00 00 00 00 00 00 00 00 00 00
50:00 00 00 00 00 00 00 00 00 00 73 DB 67 40 62 63
60:6D C0 A8 00 0B 04 0E 00 A0 C9 92 6A 45 50 19 00
70:28 41 F6 4F 4D 4F 00 79 00 3B 06 10 00 0F A4 9D
80:0C 06 37 53 0E CA 17 08 88 08 94 15 15 A0 14 1D
90:CF A0 1D 8D 1F 00 CF 91 1F 80 00 68 74 6D 6C 20
A0:48 54 54 50 2F 31 2E 31 0D 0A 41 63 63 65 70 74
B0:30 20 37 34 20 0D 0A 42 30 3A 33 30 30 20 33 30
C0:20 32 30 20 33 33 20 33 30 20 32 30 20 33 32 20
D0:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
E0:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
F0:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00



48 54 54 50 2F 31 2E 31 0D 0A 41 63 63 65 70 74
H  T  T  P  /  1  .  1  @  6  A  c  c  e  p  t
3A 20 69 6D 61 67 65 2F 67 69 66 2C 20 69 6D 61
:     i  m  a  g  e  /  g  i  f  ,     i  m  a
67 65 2F 78 2D 78 62 69 74 6D 61 70 2C 20 69 6D
g  e  /  x  -  x  b  i  t  m  a  p  ,     i  m
61 67 65 2F 6A 70 65 67 2C 20 69 6D 61 67 65 2F
a  g  e  /  j  p  e  g  ,     i  m  a  g  e  /



54 54 50 2F 31 2E 31 0D 0A 41 63 63 65 70 74 3A
T  T  P  /  1  .  1        A  c  c  e  p  t  :
20 69 6D 61 67 65 2F 67 69 66 2C 20 69 6D 61 67
   i  m  a  g  e  /  g  i  f  ,     i  m  a  g
65 2F 78 2D 78 62 69 74 6D 61 70 2C 20 69 6D 61
e  /  x  -  x  b  i  t  m  a  p  ,     i  m  a
67 65 2F 6A 70 65 67 2C 20 69 6D 61 67 65 2F 70
g  e  /  j  p  e  g  ,     i  m  a  g  e  /  p
```

HW principy Ethernetu

Počátky Ethernetu lze vystopovat do doby 1968-1972, doby kdy vznikala síť ALOHA na Havaji. Na této radiové síti byly ověřeny principy později uplatněné u Ethernetu. Jedním z hlavních principů bylo sdílení jednoho média více stanicemi na principu časového multiplexeru s náhodným časováním. Stanice, která potřebovala odeslat paket, jej vyslala a čekala na potvrzení. Pokud nedošlo ve stanoveném čase, např. že došlo ke kolizi z důvodů vysílání několika stanic najednou, vysílání se po uplynutí náhodného času opakovalo. Na tyto práce navázal v letech 1972-1977 projekt firmy Xerox v jeho výzkumném středisku PARC. Tento projekt vtiskl Ethernetu jeho další hlavní princip a tím je sledování nosné. Stanice, pokud chce vysílat, monitoruje před vysíláním stanovenou dobu přenosové médium a pokud je volné, zahájí vysílání. Oba hlavní principy jsou i součástí názvu patentu vymezujícího první definici Ethernetu - "Carrier-Sense Media Access with Collision Detection" - CSMA/CD. První experimentální síť sloužila ke spojení počítačů ALTO a prvních laserových tiskáren na světě, nazývaných EARS. Její přenosová frekvence byla odvozena ze systémových hodin procesorů ALTO a byla 2,94 Mbps.

Tyto počátky daly Ethernetu do vínku relativní spolehlivost založenou na několika robusních vlastnostech. První vlastností je vysílání synchronizace, na začátku každého bloku. Ethernet je sdílené médium, se sériovou komunikací. Komunikuje se po blocích, nesoucích na začátku každého bloku 6ti bytovou adresu příjemce následovanou 6ti bytovou adresou odesílatele, vlastním datovým blokem a na závěr 4 bytovou cyklickou sumou. Před každým blokem je vysílána synchronizační posloupnost složení ze série 31 dvojic bitů 1,0 následovaných značkou začátku bloku, reprezentovanou dvojicí bitů 1,1. Těchto, celkem 64 bitů, slouží pouze k synchronizaci a k automatickému nastavení citlivosti přijímačů. Ethernet karty je při vysílání vkládají automaticky, a při příjmu naopak vypouštějí. (Pozn. nemusí vždy platit, že přijímač obdrží vždy všech 64 bitů, některé počáteční bity mohou být "ztraceny", vlivem pozdního zapnutí obvodů vysílače, či špatného počátečního nastavení citlivosti.)

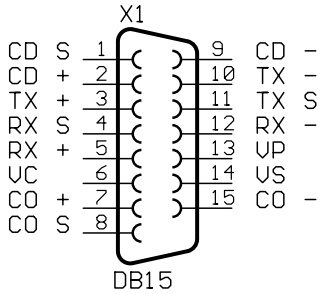
Druhou vlastností je, že každý vysílaný bit si zároveň nese sebou synchronizační bit, vyplývající z použité modulace. Ethernet používá modulaci nazývanou Manchester II. Tato modulace je založena na rozdělení bitového intervalu na dvě poloviny, v první je vysílána inverzní hodnota bitu a ve druhé je vysílán bit přímo. Je možno říci, že každý bit se nahradí ve vysílači vlastně dvojicí bitů s poloviční dobou trvání. 0 dvojicí 10 a 1 dvojicí 01. Např. výše zmíněná synchronizační posloupnost bude vypadat takto:



Hodinový signál, zobrazený na na prvním řádku slouží pouze k synchronizaci vysílače a při příjmu je jej nutno pomocí fázového závěsu obnovit z přijímaných dat. Na druhém řádku jsou uvedena data, tak jak je posílá jádro kontroleru Ethernetu do modulátoru. Na třetím řádku jsou zobrazena data, tak jak je posílá modulátor do linky.

Podrobnější pohled na vysílaná data odhalí další vlastnost, danou použitou modulací, a to konstantní stejnosměrnou úroveň během vysílání. Tato vlastnost je uplatněna hned ve dvou bodech. Prvním bodem je jednoduchá realizace galvanického odělení obvodů připojených k lince od obvodů vlastního počítače. Je realizováno pomocí důsledné symetrizace všech signálů, které odstraní stejnosměrnou složku a následného oddělení pomocí transformátorů. Druhým bodem je, pokud se neuplatní symetrizace, naopak jednoduchá možnost vyhodnocování střední hodnoty této stejnosměrné úrovně. Při vysílání se monitoruje střední hodnota a pokud dojde k odchylce, způsobené především současným vysíláním dvou a více stanic, je signalizována kolize. Této možnosti detekce kolize používají např. obvody MAU pro koaxiální kabel.

AUI konektor



AUI konektor je tvořen konektorem DB15 (2 řady, lichoběžníkový konektor Cannon, s 8+7 kontakty). Jeho osazení je popsáno v následující tabulce:

Vývod	IEEE název	Pár	Označení	Směr
3	DO + (Data Out +)	Transmit	TX+	DTE->MAU
10	DO - (Data Out -)		TX-	DTE->MAU
11	DO S (Data Out Shield)	(Vysílání)	TXS	DTE->MAU
5	DI + (Data In +)	Receive	RX+	DTE<-MAU
12	DI - (Data In -)		RX-	DTE<-MAU
4	DI S (Data In Shield)	(Příjem)	RXS	DTE->MAU
7	CO + (Control Out +)	Optional		DTE->MAU
15	CO - (Control Out -)			DTE->MAU
8	CO S (Control Out Shield)	(Volitelný)		DTE->MAU
2	CI + (Control In +)	Collision	CD+	DTE<-MAU
9	CI - (Control In -)		CD-	DTE<-MAU
1	CI S (Control In Shield)	(Detekce kolize)	CDS	DTE->MAU
6	VC (Voltage Common)	Power		DTE->MAU
13	VP (Voltage Plus)			DTE->MAU
14	VS (Voltage Shield)	(Napájení)		DTE->MAU

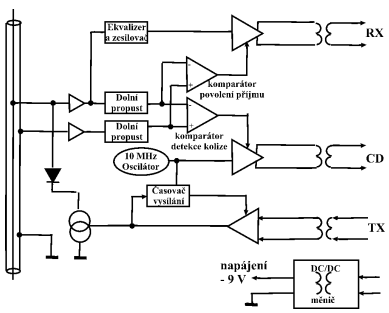
Kde směr DTE->MAU znamená směr ze zařízení směrem k médiu.

Jak probíhá vysílání dat např. po koaxiálním kabelu? Vysílaná data prochází, stejně jako další signály (přijímaná data, signalizace kolize a napájení) skrz obvody galvanického oddělení do obvodů MAU. Obvody Media Attachment Unit - jednotky pro připojení média mohou být realizovány buď přímo na kartě, nebo jako samostatná "krabička" připojovaná přes standartní rozhraní realizované konektorem AUI.

CTI - Obvody připojení koaxiálního kabelu

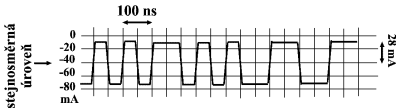
Hlavní součástí MAU pro koaxiální kabel je obvykle obvod DP8392C nebo jeho klon. Jak tento obvod, nazývaný CTI - Coaxial Transceiver Interface vlastně funguje a z čeho se skládá? V jeho zapojení lze vystopovat čtyři funkční celky:

a) Vysílač
b) Přijímač
c) Obvody detekce kolize
d) Časovač (omezovač) vysílání



a) Obvody vysílače převádějí vstupní symetrický signál na signál asymetrický, založený na modulovaném proudovém zdroji. Tento zdroj budí, pokud je povoleno vysílání, koaxiální kabel proudem -41mA (typicky, mezní hranice jsou mezi -37 a -45mA). Na uvedené stejnosměrné úrovni je namodulován vlastní užitečný signál s úrovní+-28mA. Zároveň jsou vysílanému signálu "zaobleny" hrany, tak aby se minimalizovalo rušení vyššími harmonickými. Hrany signálu mají dobu náběhu/sestupu upravenu na 25ns(+-5ns)

b) Přijímač je složitější. Obsahuje na vstupu



vysokoimpedanční nízkokapacitní zezilovač následovaný ekvalizérem, vyrovnávajícím nerovnoměrnou kmitočtovou charakteristiku koaxiálního kabelu. Na výstupu z ekvalizéru, obnovujícímu vlastně původní obdélníkový tvar signálu, vstupujícího do vysílače protistanice, je zapojen obvod šumové brány, potlačujícímu signál na výstupu, pokud není na koaxiálním kabelu předepsaná střední stejnosměrná úroveň.

c) Obvody detekce kolize jsou založeny na sledování střední stejnosměrné úrovně na koaxiálním kabelu. Vstupní signál je vyhlazen dolní propustí a veden na dva komparátory. První, jak již bylo uvedeno, monitoruje přítomnost stejnosměrné složky a odblokovává výstupní obvody přijímače. Druhý detekuje posuv stejnosměrné úrovně pod hranici cca -1,5V (typicky -1,53V, mezní hranice jsou -1,45 a -1,58V, [detailní výpočty]). Vyjdeme-li ze stejnosměrné zakončovací impedance 50Ohmů/2 (zakončovací odpory jsou na obou koncích) vychází nám, že skrz kabel musí procházet stejnosměrný proud 60mA. Výstup komparátoru, signalizující zapnutí minimálně dvou vysílačů najednou, je převeden pomocí oscilátoru 10MHz na střídavý signál a veden ven z obvodu do obvodů rozhraní AUI. Rychlost s jakou tento obvod zareaguje na kolizi je jedním z hlavních parametrů ovlivňujících zpoždění, které je nutno brát v úvahu při návrhu rozsáhlejších sítí. Typické zpoždění je 7 až 8 bitů (tj. 700 až 800ns).

d) Časovač vysílání je vlastně nouzovou brzdou, sloužící k přerušení vysílání a zablokování vysílače, pokud by chtěl vysílat déle než 20ms.

Z uvedených funkčních principů lze konstatovat, že buzení koaxiálního kabelu je proudové, přijímače naopak mají napěťový vstup. Správná funkce je proto, nejen z důvodů odstanění odrazů na koncích kabelu, podmíněna ukončením obou konců kabelu zakončovacími odpory 50Ohmů. Střední stejnosměrná úroveň na kabelu v době bezchybného vysílání je cca 1Volt. Užitečný datový signál má amplitudu cca 1,4Voltu špička-špička.

Obvody galvanického oddělení jsou realizovány obvykle trojicí miniaturních transformátorků v jednom pouzdře DIL16. Další nezbytnou součástí galvanického oddělení je měnič napětí oddělující napájení obvodů MAU od ostatních obvodů. Je rovněž většinou vyroben jako černá skříňka v pouzdře DIL24. Všechny části galvanického oddělení musí být schopny odolat u "tenkého" Ethernetu (klasický kabel RG58) 500Vst. po dobu jedné minuty a 2000Vst. u "tlustého" Ethernetu.

V obvodu CTI jsou mimo tyto hlavní části, potřebné ke komunikaci, většinou realizovány i některé další funkce, sloužící k diagnostickým a kontrolním funkcím. Jednou z těchto funkcí je funkce SQE/Heartbeat, popsaná podrobněji v kapitole o TP MAU. Standartní nastavení SQE/Heartbeat je, není-li předepsáno jinak, VYPNUTO. Použití MAU se zapnutým SQE testem ve spojení s repeatrem bez inteligence vede většinou k problémům.

Komunikace po TP kabelu, ...

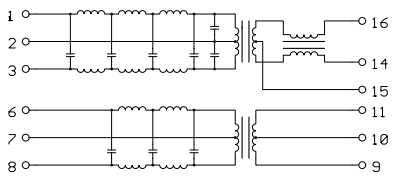
Signál na TP kabelu a jeho tvarování

Hlavní úlohou obvodu, nazývaného TPEX - Twisted-Pair Ethernet Transceiver je funkce tvarování signálu. Původní signál, generovaný obvody Manchester modulátoru je obdélník. Podíváme-li se na jeho spektrum pomocí Fourierovy transformace, zjistíme, že se skládá z řady lichých harmonických s nezanedbatelnou úrovní:
f(x)=sin(x) + 1/3 sin(3x) + 1/5 sin(5x) + ...
převedeme-li koeficienty na vyjádření v dB je úroveň 3. harmonické -4,77 dB a 5. harmonické -7 dB. Tyto hodnoty potlačení harmonických jsou naprosto nedostačující, vzhledem k tomu, že nenesou žádnou informaci a způsobují jen nežádoucí rušení. Jsou dvě možnosti jak je odstranit:

Potlačení harmonických signálů pomocí fitrace.
Buzení vedení tvarem signálu blízkým sinusovce.

1) Potlačení harmonických signálů pomocí fitrace

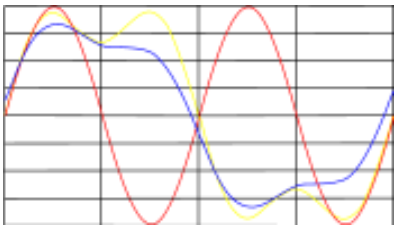
Používá buzení TP kabelu přes hybridní filtr / transformátor, např. obvod 20F001N. Na následujícím obrázku je výstupní filtr zobrazen v horní části schematu. Buzen je symetrickým obdélníkovým signálem, přivedeným na vývody 1 a 3. TP vedení je připojeno na vývody 14 a 16.



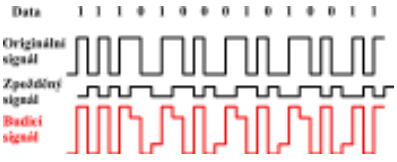
Tento obvod má frekvenci zlomu (cut-off) 17 Mhz. Útlum obvodu je uveden v následující tabulce:

20MHz	7dB
25MHz	19dB
30MHz	32dB
40MHz	35dB

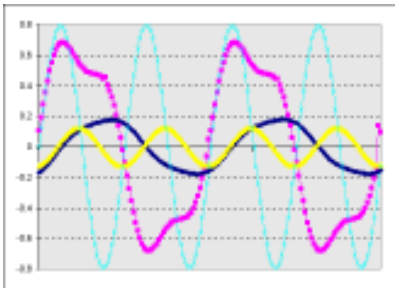
Na vyšších frekvencích je možno počítat s útlumem cca 25 dB. Přidáním tohoto filtru je vyřešena část problému. Proč ale jen část? Signál zpracovávaný tímto filtrem má totiž dvě hlavní základní frekvence, v závislosti na vysílaných datech. Vysílá li se datová sekvence 00 či 11 je modulační frekvence 10 MHz. Situace se však mění vysílá-li se sekvence 01 či 10, tato sekvence má modulační frekvenci poloviční tj. 5 MHz! Projde-li takovýto signál výše uvedeným filtrem, zůstane třetí harmonická téměř nedotčena a na výstupu dostaneme průběh zobrazený na následujícím obrázku žlutou



(signál po průchodu filtrem: **10 MHz** / **5 MHz** / **korigovaný signál 5 MHz**)
Tento signál není příliš vhodný k dalšímu zpracování. Vzhledem k obtížné realizaci filtru propouštějícího 10 MHz a majícího dostatečný útlum pro 15 MHz je nutno jít cestou korekce tvaru signálu vstupujícího do filtru. Tvar se koriguje na tvar zobrazený, po průchodu filtrem, na obrázku modrou barvou. V čem vlastně spočívá úprava tvaru signálu před filtrací? Od signálu jdoucího z Manchester kodéru se odečítá cca 20% signálu z kodéru zpožděných o 50 ns. Vznikne budicí signál zobrazený na následujícím obrázku.



Další příčinou, proč je nutno korigovat vysílaný signál, jsou vlastnosti samotného TP kabelu. Vzhledem k tomu, že má silně kapacitní charakter a je buzen z impedance cca 100 ohmů, nutné k impedančnímu přizpůsobení, projevuje se zde " setrvačnost" způsobená nabíjení a vybíjení kabelu. Po průchodu kabelem dostáváme signál zobrazený na následujícím obrázku.



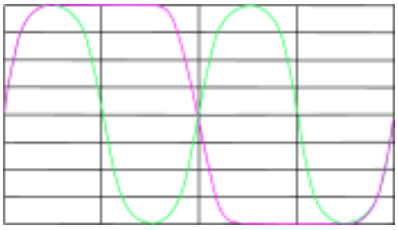
výstupní signál z filtru
10 MHz (data 1111 či 0000) / **5 MHz** (data 0101)
signál na konci kabelu
10MHz / **5 MHz**

2) Buzení vedení tvarem signálu blízkým sinusovce.

Prvním náznakem tvarování signálu je metoda využitá při komunikaci po koaxiálním kabelu. Hrany signálu mají dobu náběhu/sestupu upravenu na 25 ns. Obdélníkový signál je vlastně převeden na lichoběžníkový, skládající se z řady lichých harmonických:
f(x)=sin(x) - 1/9 sin(3x) + 1/25 sin(5x) - 1/49 sin(7x) + ...
převedeme-li koeficienty na vyjádření v dB je úroveň 3. harmonické -9,5 dB a 5. harmonické -14 dB. Tyto hodnoty potlačení harmonických jsou dvojnásobné oproti obdélníkovému signálu, nicméně jsou nepostačující.
Chceme-li budit TP kabel pouze přes hybridní transformátor, např. obvod Valor ST7011, Pulse PE-65745, YCL 16PT-41, ... je nutno jako budicí signál použít obvod blízký sinusovce. Obvody filtrace či tvarovače jsou přímo součástí integrovaných výstupních budičů. Z jednočipových řadičů Ethernetu používá tento způsob buzení kabelu např. obvod CS8900. Jak jsou obvody filtrace realizovány? Pomineme-li poloanalogová řešení je možno použít dva hlavní principy:

Generaci signálu pomocí tabulky a navazujícího D/A převodníku, tento princip je uplatněn např. v obvodu LXT901/907, kde je použit 5-ti bitový D/A převodník pracující na 70 MHz.

Tvarování signálu pomocí FIR filtru, tento princip je uplatněn např. u obvodu MTD907, kde je použit 16-ti bodový FIR filtr pracující na 160 MHz. Ukázkou činnosti takovéhoto filtru může být např. signál zobrazený na následujícím obrázku.

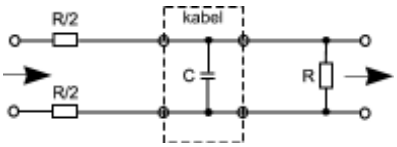


Zobrazený signál byl získán průchodem obdélníkového **10**/**5** MHz signálu Blackmanovo FIR filtrem. Nicméně i přes vyhovující spektrum výstupního signálu, se vzhledem k nutnosti kompenzovat vlastnosti TP kabelu, používá korekce tvaru uvedená v závěru odstavce 1.

TP kabel a vliv jeho kapacity

Průběh napětí na TP kabelu v závislosti na jeho kapacitě

Vyjdeme-li z velmi zjednodušeného modelu kabelu a jeho budičů dostaneme následující náhradní zapojení:



Odpory o hodnotě poloviny vlnové impedance reprezentují seriové budicí/přizpůsobovací odpory mezi výtupem vlastního integrovaného budiče a hybridním filtrem/transformátorem. Vliv náradní impedance filtru a transformátoru je zahrnut do budicího signálu a proto není v tomto náhradním schematu uvažován. Náhradní impedance kabelu je pro další výklad silně zjednodušena a pro další výpočty je uvažována pouze kapacita kabelu. Na výstupu je kabel zakončen zatěžovacím odporem, rovným vlnové impedanci kabelu. Tento zatěžovací odpor je součástí vstupních obvodů přijímače

Jaké je jsou vlastnosti kabelů používaných pro TP kabeláž?

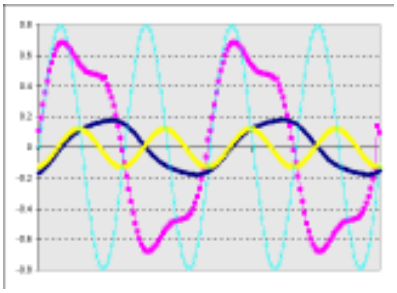
Vybereme-li z katalogu jednoho z předních světových výrobců, firmy Belden dva typické představitele, každého z opačných konců nejlepší/nejhorší dostáváme:

Typ	Odpor	Kapacita	
1700A (DataTwist 350)	78,7 ohm/km	45,9 pF/m	rozšířená Cat.5 do 350 MHz
1247A (DataTwist Three)	88,6 ohm/km	72,2 pF/m	Cat.3

oba kabely mají nominální impedanci 100 ohm

Na následujících obrázcích jsou použity pro označení signálu barvy:
vstupní signál **10 MHz** (data 1111 či 0000) / **5 MHz** (data 0101), výstupní signál **10MHz** / **5 MHz**

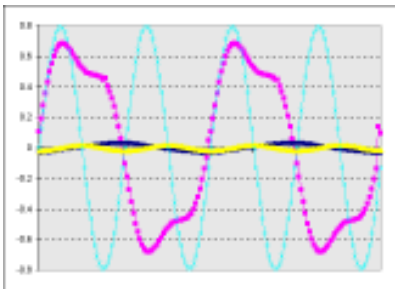
Průběh na napětí kabelu při kapacitě **1 nF**, což odpovídá 21 m kabelu Belden typ 1700 či 13 m kabelu Belden typ 1247



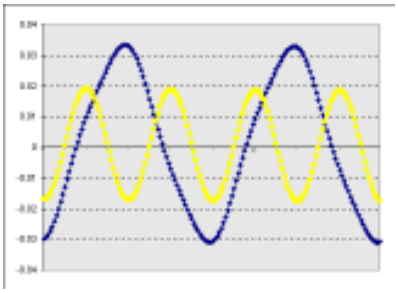
Z tohoto průběhu je patrné proč se používá k buzení kabelu v případě 5 MHz, daný tvar signálu, zabezpečující ve své první části stejný náběh signálu na výstupu kabelu jako u signálu 10 MHz a ve své druhé části omezující vliv RC článku budič-kabel a jeho vliv na amplitudu signálů s různým kmitočtem.

Další obrázky ukazují stejný signál při různých kapacitách kabelu

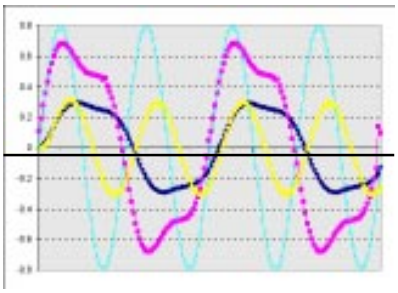
Průběh na napětí kabelu při kapacitě **7 nF**, což odpovídá 150 m kabelu Belden typ 1700 či 90 m kabelu Belden typ 1247



Výstupní signál v detailu



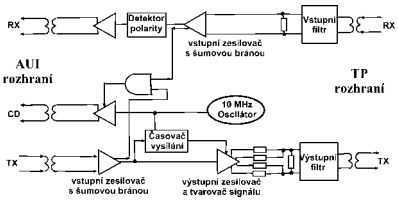
Průběh na napětí kabelu při kapacitě **300 pF**, což odpovídá 6 m kabelu Belden typ 1700 či 4 m kabelu Belden typ 1247



TPEX - Obvody připojení TP kabelu

Hlavní součásti TP MAU

Hlavní součástí MAU pro TP kabel je obvykle obvod Am79C98 nebo jeho klon. Jak tento obvod, nazývaný TPEX - Twisted-Pair Ethernet Transceiver vlastně funguje a z čeho se skládá? Jeho zapojení je celkem jednoduché:



1) Na straně vysílače je vysílaný signál, přicházející z AUI interface zesílen. Po zesílení je veden do výstupního budiče, který zajišťuje na svém výstupu dvě dvojice symetrických signálů, jeden pouze zesílený a druhý zpožděný o 50 ns. Po sečtení externími odpory vznikne průběh uvedený na obrázku 3, který je veden do výstupního hybridního transformátoru / filtru. Mimo dobu vysílání je výstupní budič TP kabelu uzavřen pomocí šumové brány

2) Na straně přijímače prochází signál z výstupu vstupního hybridního transformátoru / filtru vstupním zesilovačem a přes obvod šumové brány je veden do budičů AUI interface. Obvod "Detektor polarity" je pomocným obvodem a obrací fázi signálu o 180 stupňů v případě, že je špatně zapojena kabeláž (otočena polarita páru).

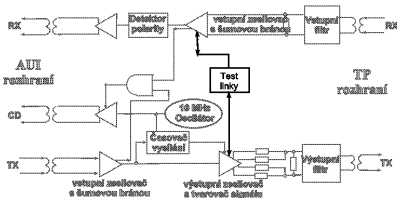
3) Mimo těchto dvou, řekněme "tvarovačů signálu" TPEX obsahuje další obvod a to obvod detekce kolize. Kolize je na TP kabeláži detekována odlišně, od způsobu detekce na koaxiálním kabelu. Nelze zde, vzhledem k nesdílenému vedení mezi několika vysílači, detekovat kolizi pomocí změn stejnosměrné úrovně. Kolize se detekuje přímo z vlastnosti, že pokud jeden vysílá tak ostatní mlčí, tj. přijímací část je pomocí šumové brány vypnuta. Zjistí-li proto obvod TPEX současnou aktivaci vysílací i přijímací části vygeneruje na výstupu CD AUI interface signál kolize.

Pomocné funkce TP MAU

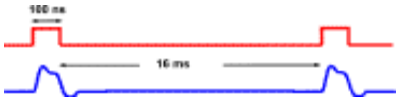
Obvody, uvedené v předchozím odstavci, stačí pro řádnou komunikační funkci, nestačí však pro kontrolní účely. Představte si, že připojíte nějaké měřicí zařízení, které má vysílat jednou za hodinu. Čekat u něj, až proběhne vysílání, aby bylo možno zkontrolovat, zda je připojovací kabeláž vpořádku, jak sami uznáte není to pravé. Proto byla komunikace na TP a tím i příslušné obvody, doplněny o několik kontrolních funkcí. Jednou z nich, je zpětná smyčka (loopback) předávající data ze vstupu pro vysílaná data AUI(TX) zpět na výstup přijímaných dat AUI(RX). Na schematických obrázcích popisovaný obvod používá tuto funkci jen v diagnostickém režimu, nicméně existuje poměrně hodně obvodů, kde je tato funkce implicitní. Znamená to tedy, že pokud vysíláme, tak se vysílaná data vracejí přes výstup přijímače. Pokud ovšem dojde ke kolizi, je loopback vypnut a na výstupu přijímače jsou skutečná přijímaná data. Celý obvod se tedy chová stejně, jako by byl nebyla použita TP kabeláž, ale kabeláž založená na koaxiálnímu kabelu.

1) Linkový test

Linkový test patří mezi hlavní testovací funkce. Jeho schematické umístění v blokovém zapojení je vidět na následujícím obrázku:



Tento obvod kontroluje neporušenost linky a její polaritu. Provádí to pomocí jednoduché metody, není-li delší dobu, obvykle cca 16 ms co vysílat, obvod vyšle testovací linkový puls, zobrazený na následujícím obrázku.

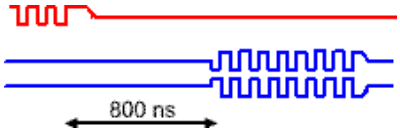


ideální (teoretický) průběh
upravený (korigovaný) průběh
po průchodu filtrem

Obvody přijímače vyhodnocují, pomocí časového okna, udávajícího minimální (cca 4 ms) a maximální (cca 60 ms) dobu, ve které můžou linkové pulsy přijít od předchozí aktivity protistanice (tj. vysílání dat nebo předchozího linkového pulsu), přítomnost linkových pulsů. Je-li tato nepřítomnost delší než je hranice (cca 64 ms), obvod nahlásí ztrátu spojení. Tato ztráta spojení je navenek uživateli obvykle signalizována zhasnutím LED "Link" na zařízení. Přijímač tyto linkové pulsy neposílá dále na svůj výstup. Vzhledem ke známé polaritě a tvaru linkových pulsů, jsou tyto pulsy použity rovněž pro vyhodnocení, zda nedošlo na trase vysílač-přijímač k prohození vodičů v páru a tím k otočení fáze signálu. Tuto chybu polarity pulsu vyhodnotí obvody detekce polarity přijímaného signálu a korigují ji. Linkový test patří k základním testům a můžeme-li volit má-li být zapnutý, je jeho standartní nastavení ZAPNUTO. Modifikovaným linkovým pulsem se v dnešní době testují možnosti zařízení, především jeho schopnost komunikovat na 100Mbitové rychlosti a plným duplexem. Např. signalizace plného duplexního režimu je realizována jako vyslání 16-ti linkových pulsů ve standartním časování následovaných jedním "rychlejším" pulsem s odstupem od předcházejícího cca 5 ms.

2) SQE (Heartbeat) Test

Signal Quality Test, nazývaný též Heartbeat (tep srdce) patří mezi pomocné testovací funkce. Jeho účelem je testovat signalizaci kolize na rozhraní AUI. Je-li tato funkce zapnuta, tak je na výstupu signalizace kolize (CD) signalizováno mimo kolizí ukončení vysílání paketu. Prakticky je tento self-test prováděn tak, že obvod počká po skončení vysílání 800 ns (tj. 8 bitů) a pak vyšle na výstup signalizace kolize CD 8 bitů. Celá situace je znázorněna na následujícím obrázku:



TX (AUI) **vysílaný signál**
CD +/- (AUI) **signál kolize**

SQE test patří k pomocným testů a jeho standartní nastavení je, není-li předepsáno jinak, např. při připojení na nějakou síťovou kartu vysloveně jej vyžadující, VYPNUTO. Použití MAU se zapnutým SQE testem ve spojení s repeatrem bez inteligence vede většinou k problémům.

Zapojení konektorů a kabelů TP

Typické propojení počítače s aktivním prvkem je tvořeno ze tří částí.

propojovacím "patch" kabelem mezi síťovou kartou v počítači a zásuvkou.
strukturovanou kabeláží spojujícím zásuvky s příslušnými panely v rozvaděči
propojovacím "patch" kabelem mezi panelem a aktivním prvkem (hubem, switchem, ...)

Pozn. *V dalším textu je aktivní prvek označen názvem **hub** (koncentrátor), i když se ve skutečnosti může jednat o switch (přepínač) či jiný obdobný prvek.*

Strukturovaná kabeláž, používaná pro rozvody Ethernetu technologií TP, dnes používá nejčastěji konektorů RJ45. Tedy z pohledu uživatele naprosto stejných konektorů jaké jsou na TP síťových kartách. Od nich se liší v části "skryté" uživateli pod panelem a to částí určenou pro připojení kabelů, spojujících jednotlivé konektory v zásuvkách s příslušnými konektory v patch panelech. Zde není technické provedení jednotné a závisí na výrobci příslušného konektoru.

Dnes se nejčastěji používají zářezové bloky systému S110, popř. Krone, doplněné více nebo méně dokonalou mechanickou fixací přívodního kabelu.

Problémy s fixací kabelů byly jedním z důvodů který dal vzniknout řadě dalších konektorovacích systémů. Za velmi zdařilý, alespoň dle mého názoru, lze považovat např. systém PanJack firmy Panduit.

Strukturovaná kabeláž používá čtyřpárové kroucené "twistované" kabely. Pevně instalovaná kabeláž používá kabelů z drátů, propojovací kabely jsou zhotovovány z lanek. Průřez kabelů je typicky 24 AWG (drát, lanko 7/32). Podle příslušného provedení kabelu je nutno vybírat i příslušné provedení konektorů. Jednotlivé páry v kabelu jsou označeny barevně.

1 Modrá (Blue)
2 Oranžová (Orange)
3 Zelená (Green)
4 Hnědá (Brown)

Vždy jeden z drátů v páru má příslušnou barvu a druhý do páru je buď bílý, nebo různě proužkovaný v kombinaci bílá/příslušná barva. Připojení jednotlivých drátů ke kontaktům konektoru je standartizováno. Pro "jednoduchost" se zde používají dva barevné standarty T568A a T568B mající mezi sebou vzájemně prohozený zelený a oranžový pár, tj. páry používané u 10Base-T/100Base-T ke komunikaci. Oba barevné standarty jsou zobrazeny na nákresech na této stránce.

Standart 10BaseT / 100BaseT používá pro komunikaci pouze dva páry, pár 2 (oranžová) a 3 (zelená). Zbývající páry 1 (modrá) a 4 (hnědá) jsou ke komunikaci nepoužity a je možno je např. při požadavku na velmi ohebný, lehký a skladný propojovací kabel (např. pro připojení notebooku do sítě) zcela vynechat. Standartně je konektor RJ45 osazený na síťových kartách zapojen dle zapojení označovaného zkratkou **MDI**:

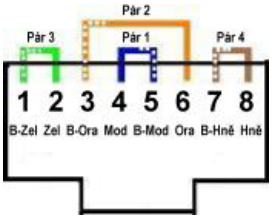
Pin	Jméno	Směr
1	TD+	karta -> hub
2	TD-	karta -> hub
3	RX+	hub -> karta
4	N.C.	nezapojeno
5	N.C.	nezapojeno
6	RX-	hub -> karta
7	N.C.	nezapojeno
8	N.C.	nezapojeno

Kabely, propojujících počítač s hubem, nebo ze zásuvkou rozvodu LAN jsou zapojeny 1:1, tj. spolu jsou spojeny stejně očíslované vývody. Nicméně při eventuální výrobě kabeláže je nutno dbát na dodržení vysokofrekvenčních vlastností a tím i párování jednotlivých signálů tj. 1/2, 3/6, 4/5 a 7/8.

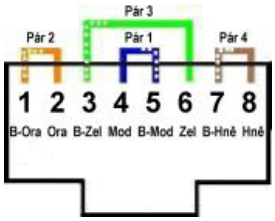
Pokud potřebujeme propojit pouze 2 počítače nepotřebujeme hub či jiný další aktivní prvek, ale pouze křížový propojovací kabel. Stejný kabel je nutno používat i pro propojení dvou hubů či jiných aktivních prvků v případě, že použitý prvek nemá možnost prohození vývodů TD a RX. Vývody pro propojení hubu s nadřazeným hubem jsou obvykle řešeny pomocí přepínače či parallelně zapojeného "kříženého" konektoru.
Překřížené zapojení, prohazující mezi sebou signály TD a RX, je označováno zkratkou **MDI-X**. V případě parallelně zapojeného konektoru je možno zapojit pouze jeden z konektorů, druhý musí zůstat nezapojen !

T568A	T568B
1 - Bílá / Zelená	1 - Bílá / Oranžová
2 - Zelená	2 - Oranžová
3 - Bílá / Oranžová	3 - Bílá / Zelená
4 - Modrá	4 - Modrá
5 - Bílá / Modrá	5 - Bílá / Modrá
6 - Oranžová	6 - Zelená
7 - Bílá / Hnědá	7 - Bílá / Hnědá
8 - Hnědá	8 - Hnědá

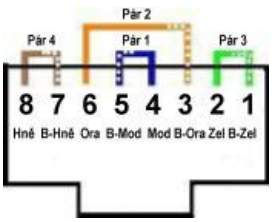
pohled zepředu na zásuvku



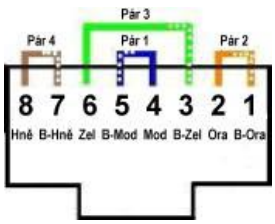
pohled zepředu na zásuvku



pohled zepředu na zástrčku



pohled zepředu na zástrčku



Zapojení standartních, nepřekřížených kabelů

1 - 1
2 - 2
3 - 3
4 - 4
5 - 5
6 - 6
7 - 7
8 - 8

1 - 1 2 - 2 3 - 3 4 nezapojeno 5 nezapojeno 6 - 6 7 nezapojeno 8 nezapojeno	Minimalizovaná varianta, NEDOPORUČOVÁNO, pouze pro speciální účely, především s ohledem na rozměry a hmotnost kabelu (připojení notebooků a pod.)

Zapojení překřížených kabelů

1 - 3
2 - 6
3 - 1
4 - 8
5 - 7
6 - 2
7 - 5
8 - 4

1 - 3 2 - 6 3 - 1 4 - 4 5 - 5 6 - 2 7 - 7 8 - 8	Někdy používaná varianta, křížící pouze páry TD a RX a zbývající páry 1 a 4 propojující 1:1
1 - 3 2 - 6 3 - 1 4 nezapojeno 5 nezapojeno 6 - 2 7 nezapojeno 8 nezapojeno	Minimalizovaná varianta, křížící pouze páry TD a RX

Na obrázcích zobrazené konektory, demonstrující zapojení kabelu jsou zapojeny dle T568B. Pro T568A je nutno zaměnit oranžová a zelené páry.

Označení kříženého kabelu písmenem X (od MDI-X či X-over), zobrazené na obrázku, na koncovce kabelu není standartizováno a je pouze autorovou zvyklostí umožňující na první pohled odlišit křížený kabel. Jak je tedy možno jednoduše odlišit od sebe křížený a standartní kabel ? Vezmeme oba konektoru a dáme si je stejným směrem, najdeme krajní pár označený zelenou či oranžovou barvou a zkontrolujeme zda je na druhém konektoru má krajní pár stejné barevné označení. Pokud je na obou konektorech krajní pár zelený, jde o standartní kabel dle T568A. Pokud je na obou konektorech krajní pár oranžový, jde o standartní kabel dle T568B (u kabelů častěji používaná varianta, autor ji používá i v rozvodech). Pokud je na jednom konektoru krajní pár zelený a na druhém oranžový, jde o křížený kabel.

FL MAU - Obvody připojení optického kabelu

Komunikace po optickém kabelu potřebuje minimálně tři prvky:

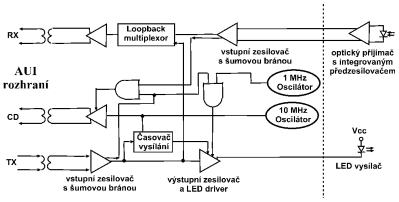
optický vysílač a jeho budič
optický kabel
optický přijímač s navazujícími tvarovači signálu

Optický přijímač a vysílač s příslušnou elektronikou je obvykle tvořen modulem MAU (Media Attachment Unit - jednotka pro připojení média). Tyto moduly mohou být jak externí (jsou znázorněny na obrázku), připojované přes AUI interface, tak interní, ve firemním provedení určeném přímo pro zasunutí do hubu či přepínače. Jako vysílací prvek se v těchto obvodech používá LED dioda. Přijímacím prvkem obvykle bývá PIN fotodioda. Jak vysílací, tak přijímací dioda mají speciální pouzdření, jehož součástí je příslušný konektor pro připojení optického kabelu. Těchto konektorů se používá několik druhů. Dva typičtí představitelé jsou na obrázcích MAU jednotek. Levá používá ST konektory, aretované stejným způsobem jako BNC konektory, tj. bajonetovým otočným kroužkem. Pravá používá SC konektory, které jsou hranaté a aretované mechanismem ovládaným posuvnou vnější částí konektoru. SC konektory je možno speciální sponkou spojovat do dvojic. (Na obrázku jsou do SC konektorů zasunuty gumové ochrané zátky, chránící citlivou optickou část před znečištěním.)

Optický kabel je obvykle skleněný a tvořen jedním či několika páry optických vláken. Každé vlákno má na sobě několik vrstev izolace a umožňuje, pokud nejsou použity speciální technologie, jednosměrnou komunikaci. K plnému připojení do ethernetové sítě jsou tedy, obdobně jako u TP kabelů, potřebná dvě vlákna, jedno pro vysílací a druhé pro přijímaci směr. V současnosti se používají tři hlavní rozměry optických vláken: 9m/125m (SM), 50m/125m, 62.5m/125m (MM). Co vlastně tyto rozměry znamenají? Optický kabel je tvořen ze dvou vrstev s odlišným indexem lomu světla. Jedné vnitřní, nazývané jádro (core), a jedné vnější, nazývané plášť (cladding). Uvedené rozměry jsou rozměr jádra / rozměr pláště. Optická komunikace prochází pouze jádrem, úlohou pláště je vytvářet na rozhraní jádro/plášť "odraznou" vrstvu "držící" světelný paprsek uvnitř jádra. Druhý pojem, používaný u optických kabelů, je zda je kabel jednovidový (singlemode - SM), nebo multividový (multimode - MM). Tento pojem souvisí s počtem módů světla, které je daný kabel schopný přenášet. SM kabely, mají vzhledem k tomu, že přenáší pouze jeden mód světla daleko lepší parametry co se týká přenosových vlasností (šířka pásma, zkreslení impulsu, útlum). Proto se používají pro dálková vedení. MM kabely se používají především v sítích LAN pro vzdálenosti do jednotek kilometrů. Hlavní důvod, proč se používají MM kabely je nižší cena prvků a technologií potřebných k jejich spojování. Každý kabel je nutno minimálně navázat na vysílací a přijímací prvek.SM kabel má jádro o průměru pouhých 8 až 9 m a proto je nutno např. konektory vybírat na přesné tolerance. U MM kabelu 62.5/125m není tak kritické, zda je vnitřní průměr otvoru pro optické vlákno 126 či 128m.

Pro vysílání se obvykle používá viditelná vlnová délka 850nm (červená) pro 10 Mbit komunikaci, pro 100 Mbit komunikaci se používá 1300nm vzhledem k lepším vlasnostem optických vláken na této vlnové délce.

Hlavní součástí MAU pro optický kabel je obvykle obvod HFBR-4663 či ML4668. Jak tento obvod vlastně funguje a z čeho se skládá? Jeho zapojení je velmi podobné obvodu používanému pro TP kabely:



1) Na straně vysílače je vysílaný signál, přicházející z AUI interface zesílen. Po zesílení je veden do výstupního budiče, který je tvořen řízeným zdrojem proudu. Tento zdroj je z důvodů omezení rušení tvořen zdrojem konstantního proudu (nastavitelným, obvykle cca 50 mA) přepínaného buď na LED výstup nebo Vcc. Mimo dobu vysílání je výstupní budič FO kabelu pomocí šumové brány přepnut do režimu vysílání IDLE signálu který má obdobnou funkci jako linkový test u TP obvodů. Proti němu je však jednodušší a je tvořen 1 MHz signálem. Tento signál umožňuje protistanici monitorovat kvalitu linky a seřídit si citlivost vstupního zesilovače.

2) Na straně přijímače prochází signál z výstupu integrovaného optického příjímače vstupním zesilovačem s řízeným ziskem a přes obvod šumové brány, která mimo jiné potlačuje signály s frekvencí menší než cca 2.5 MHZ a tím i IDLE signál, je veden do budičů AUI interface. Multiplexor, zapojený před výstupní budič AUI interface, je určen k realizaci zpětné smyčky (loopback) předávající data ze vstupu pro vysílaná data AUI(TX) zpět na výstup přijímaných dat AUI(RX). Na schematických obrázcích popisovaný obvod používá tuto funkci implicitně. Znamená to tedy, že pokud vysíláme, tak se vysílaná data vracejí přes výstup přijímače. Pokud ovšem dojde ke kolizi, je loopback vypnut a na výstupu přijímače jsou skutečná přijímaná data. Celý obvod se tedy chová stejně, jako by byl nebyla použita optická kabeláž, ale kabeláž založená na koaxiálnímu kabelu.

3) Mimo těchto dvou, řekněme "tvarovačů signálu" obsahuje obvod detektor kolize. Kolize se detekuje přímo z vlasnosti, že pokud jeden vysílá tak ostatní mlčí, tj. přijímací část je pomocí šumové brány vypnuta. Zjistí-li proto obvod TPEX současnou aktivaci vysílací i přijímací části vygeneruje na výstupu CD AUI interface signál kolize.